Suivi visuel d'objets dans un réseau de caméras intelligentes embarquées

Aziz Dziri

N° d'ordre : D. U : 2610
E D S P I C : 715

UNIVERSITÉ BLAISE PASCAL - CLERMONT-FERRAND II

ÉCOLE DOCTORALE SPI
SCIENCES POUR L'INGÉNIEUR

T H È S E

Présentée par

Aziz DZIRI

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

de l'Université Blaise Pascal - Clermont-Ferrand
Spécialité : VISION POUR LA ROBOTIQUE

Suivi visuel d'objets dans un réseau de caméras intelligentes embarquées

Thèse préparée au CEA-LIST, dans le laboratoire
Adéquation Algorithme Architecture (L3A)

Soutenue publiquement le 30 octobre 2015 devant le jury :

M.	Alain Mérigot	Université Paris−Saclay	Président
M.	Andrea Cavallaro	Queen Mary University of London	Rapporteur
Mme.	Catherine Achard	Université Pierre et Marie Curie	Rapporteur
M.	Quoc-Cuong Pham	CEA-LIST	Examinateur
M.	Thomas Dombek	CEA-LIST	Invité
M.	Roland Chapuis	Université Blaise Pascal	Directeur de thèse
M.	Marc Duranton	CEA-LIST	Encadrant

Résumé :
Le suivi d'objets est de plus en plus utilisé dans les applications de vision par ordinateur. Compte tenu des exigences des applications en termes de performance, du temps de traitement, de la consommation d'énergie et de la facilité du déploiement des systèmes de suivi, l'utilisation des architectures embarquées de calcul devient primordiale. Dans cette thèse, nous avons conçu un système de suivi d'objets pouvant fonctionner en temps réel sur une caméra intelligente de faible coût et de faible consommation équipée d'un processeur embarqué ayant une architecture légère en ressources de calcul. Le système a été étendu pour le suivi d'objets dans un réseau de caméras avec des champs de vision non-recouvrant.

La chaîne algorithmique est composée d'un étage de détection basé sur la soustraction de fond et d'un étage de suivi utilisant un algorithme probabiliste Gaussian Mixture Probability Hypothesis Density (GMPHD). La méthode de soustraction de fond que nous avons proposée combine le résultat fournie par la méthode Zipfian Sigma-Delta avec l'information du gradient de l'image d'entrée dans le but d'assurer une bonne détection avec une faible complexité. Le résultat de soustraction est traité par un algorithme d'analyse des composantes connectées afin d'extraire les caractéristiques des objets en mouvement. Les caractéristiques constituent les observations d'une version améliorée du filtre GMPHD. En effet, le filtre GMPHD original ne traite pas les occultations se produisant entre les objets. Nous avons donc intégré deux modules dans le filtre GMPHD pour la gestion des occultations. Quand aucune occultation n'est détectée, les caractéristiques de mouvement des objets sont utilisées pour le suivi. Dans le cas d'une occultation, les caractéristiques d'apparence des objets, représentées par des histogrammes en niveau de gris sont sauvegardées et utilisées pour la ré-identification à la fin de l'occultation. Par la suite, la chaîne de suivi développée a été optimisée et implémentée sur une caméra intelligente embarquée composée de la carte Raspberry Pi version 1 et du module caméra RaspiCam. Les résultats obtenus montrent une qualité de suivi proche des méthodes de l'état de l'art et une cadence d'images de $15 - 30$ fps sur la caméra intelligente selon la résolution des images.

Dans la deuxième partie de la thèse, nous avons conçu un système distribué de suivi multi-objet pour un réseau de caméras avec des champs non recouvrants. Le système prend en considération que chaque caméra exécute un filtre GMPHD. Le système est basé sur une approche probabiliste qui modélise la correspondance entre les objets par une probabilité d'apparence et une probabilité spatio-temporelle. L'apparence d'un objet est représentée par un vecteur de m éléments qui peut être considéré comme un histogramme. La caractéristique spatio-temporelle est représentée par le temps de transition des objets et la probabilité de transition d'un objet d'une région d'entrée-sortie à une autre. Le temps de transition est modélisé par une loi normale dont la moyenne et la variance sont supposées être connues. L'aspect distribué de l'approche proposée assure un suivi avec peu de communication entre les noeuds du réseau. L'approche a été testée en simulation et sa complexité a été analysée. Les résultats obtenus sont prometteurs pour le fonctionnement de l'approche dans un réseau de caméras intelligentes réel.

Mots clés : Suivi visuel d'objets, Soustraction de fond, GMPHD, Occultations, Caméra intelligente, Réseau de caméras.

Visual multi-object tracking in a network of embedded smart cameras

Abstract :
Multi-object tracking constitutes a major step in several computer vision applications. The requirements of these applications in terms of performance, processing time, energy consumption and the ease of deployment of a visual tracking system, make the use of low power embedded platforms essential. In this thesis, we designed a multi-object tracking system that achieves real time processing on a low cost and a low power embedded smart camera. The tracking pipeline was extended to work in a network of cameras with non-overlapping field of views.

The tracking pipeline is composed of a detection module based on a background subtraction method and on a tracker using the probabilistic Gaussian Mixture Probability Hypothesis Density (GMPHD) filter. The background subtraction, we developed, is a combination of the segmentation resulted from the Zipfian Sigma-Delta method with the gradient of the input image. This combination allows reliable detection with low computing complexity. The output of the background subtraction is processed using a connected components analysis algorithm to extract the features of moving objects. The features are used as input to an improved version of GMPHD filter. Indeed, the original GMPHD do not manage occlusion problems. We integrated two new modules in GMPHD filter to handle occlusions between objects. If there are no occlusions, the motion feature of objects is used for tracking. When an occlusion is detected, the appearance features of the objects are saved to be used for re-identification at the end of the occlusion. The proposed tracking pipeline was optimized and implemented on an embedded smart camera composed of the Raspberry Pi version 1 board and the camera module RaspiCam. The results show that besides the low complexity of the pipeline, the tracking quality of our method is close to the stat of the art methods. A frame rate of $15 - 30$ was achieved on the smart camera depending on the image resolution.

In the second part of the thesis, we designed a distributed approach for multi-object tracking in a network of non-overlapping cameras. The approach was developed based on the fact that each camera in the network runs a GMPHD filter as a tracker. Our approach is based on a probabilistic formulation that models the correspondences between objects as an appearance probability and space-time probability. The appearance of an object is represented by a vector of m dimension, which can be considered as a histogram. The space-time features are represented by the transition time between two input-output regions in the network and the transition probability from a region to another. Transition time is modeled as a Gaussian distribution with known mean and covariance. The distributed aspect of the proposed approach allows a tracking over the network with few communications between the cameras. Several simulations were performed to validate the approach. The obtained results are promising for the use of this approach in a real network of smart cameras.

Keywords : Visuel multi-objet tracking, Background subtraction, GMPHD, Occulusions, Smart camera, Network of cameras.

Remerciements

Je souhaite remercier les rapporteurs pour le temps qu'ils ont accordé à la lecture de ma thèse : merci au docteur Catherine Achard pour son rapport détaillé, ses remarques constructives et pour avoir relevé toutes les typographies et fautes dans le document, je lui en suis très reconnaissant. Thanks to the professor Andrea Cavallaro for reviewing my thesis and coming to the defense.

Je remercie le professeur Alain Mérigot d'avoir présidé le jury de thèse. Merci au docteur Quoc-Cuong Pham d'avoir accepté d'examiner ma thèse. Merci à Thomas Dombek qui a accepté de faire partie du jury de thèse.

Un grand merci à mes encadrants Roland Chapuis et Marc Duranton pour leurs conseils, leur soutien et l'intérêt qu'ils ont porté au sujet tout au long de ces trois années de thèse. Je les remercie pour leur ouverture d'esprit et la liberté qu'ils m'ont accordée dans les choix. Roland, merci pour tes précieux conseils et pour ta disponibilité même avec la grande distance géographique. Tes qualités de pédagogue et ta rigueur m'ont permis de comprendre des problèmes complexes avec des mots et des exemples simples, les échanges avec toi étaient toujours productifs. Marc, merci pour ton soutien et tes conseils. Ton esprit d'analyse m'a toujours forcé à avoir du recul sur le sujet et de se poser les bonnes questions.

Merci au CEA de m'avoir accueillie dans sa structure. Cette thèse a commencé au laboratoire LCE. Merci au chef du laboratoire Raphael David pour l'intérêt qu'il a porté au sujet et pour ses retours suites aux points de thèse auxquels il a participé. Les neufs derniers mois de cette thèse ont été effectués au laboratoire L3A. Merci au chef du laboratoire Thomas Dombek pour ses conseils qui m'ont été très précieux.

Je remercie aussi Stéphane Chevobbe pour ses conseils et sa relecture de quelques parties du document de thèse. Cela m'a été très bénéfique.

Je remercie Suresh Pajaniradja pour son aide au montage de la démonstration de suivi mono-caméra et pour tous ses conseils concernant la photographie et la plongée sous-marine. Merci à Mehdi Darouich pour les quelques échanges qu'on a eu sur ma thèse. Ils m'ont été d'une grande utilité.

Merci à Annie Straboni, la secrétaire du département pour sa bonne humeur et pour son aide sur la partie administrative tout au long des trois années de thèse.

Un grand merci également à mes parents grâce à qui j'ai pu atteindre mes objectifs. Merci pour leur soutien et leurs conseils durant toutes ces années passées. Merci à mon frère Massinissa pour son soutien.

Un grand merci aux doctorants avec qui j'ai partagé ces trois années de thèse : Alexandre Aminot avec qui les discussions sont toujours agréables et avec qui j'ai beaucoup appris sur le plan professionnel ainsi que sur le plan personnel, Karl-Eduard Berger pour sa bonne humeur et pour les discussions philosophiques sur la vie qu'on a eu, et Safae Dehmani avec qui j'ai partagé des moments de travail à la bibliothèque pour finaliser cette thèse.

Je remercie mes co-bureau (derniers et anciens) : Michal Szczepanski, Dzila Adjemian, Fady Azar Abi Fadel, Gregory Vaumourin, Fabrice Mayran de Chamisso, Alexandre Aminot, Julien Collet, Benjamin Chomarat et Marvin Faix pour leur sympathie, bonne humeur et pour les discussions intéressantes qu'on a eu l'occasion d'échanger. Je remercie particulièrement Michal et Dzila de m'avoir supporté pendant la phase de rédaction.

Merci à Katerina T. qui m'a supporté et a cru en moi pendant les périodes difficiles de la thèse.

Merci à l'équipe de WWO, Luca, Andrea, Alain, Alexandre, Vincent, Karl et Daniela avec qui le sport est devenu une partie de la thèse.

Merci à tous les collègues pour qui la bonne humeur était au quotidien : Erwan, Caaliph, Charly, Baptiste A, Baptise C, Mehdi, Stéphane, Karim, Laurent, Vincent, Alexandre G, Boukary, Jean-Lucien ...

Merci à toutes les personnes du DACLE avec qui j'ai eu des échanges durant ces trois années de thèse.

Pour finir, merci à toutes les personnes que j'ai malencontreusement oublié de remercier.

Table des matières

Introduction

Contexte et Motivation

Le suivi d'objets par vision constitue un maillon essentiel dans un grand nombre d'applications comme la vidéo surveillance, le contrôle du trafic routier, l'aide à la conduite automobile ou encore l'analyse du comportement et l'aide aux personnes âgées. Selon le contexte, un seul ou plusieurs objets sont considérés et une seule ou plusieurs caméras sont déployées. Le suivi d'objets avec une seule caméra présente plusieurs challenges liés aux limitations des capteurs de vision (faible cadence d'image, basse résolution, faible dynamique par pixel, distorsions des couleurs, bruit, champ de vision limité, *etc*), aux objets (objets non rigides, nombre d'objets variant au fil de temps, occultations entre objets, petites tailles d'objets, *etc*), aux exigences des scénarios applicatifs (fonctionnement en temps réel, haute fiabilité du système, *etc*) et à l'environnement (variation d'éclairage, occultations causées par l'environnement, *etc*). Différentes approches ont été développées pour répondre à ces challenges et pour assurer une bonne qualité de suivi mono-caméra. Cependant, certains challenges restent difficiles à résoudre avec une seule caméra, notamment, les problèmes d'occultations quand le nombre d'objets est élevé, comme les scènes de foule par exemple, ou le problème de surveillance d'une large région. Pour cette raison, des systèmes de suivi à base de réseaux de caméras ont été élaborés. Le déploiement de plusieurs caméras introduit de nouveaux challenges tels que la synchronisation et la calibration de caméras ainsi que la fusion de données. Les contraintes principales dans les réseaux de caméras sont liées au coût des noeuds du réseau, à la consommation énergétique, aux besoins en bande passante et à l'intersection entre les champs de vision des caméras.

D'un côté, plusieurs méthodes de suivi d'objets assurant de bonnes performances ont été élaborées pour fonctionner sur des machines de calcul de hautes performances, comme les ordinateurs personnels ou les serveurs de calcul, dont le coût et la consommation en énergie sont élevés. D'un autre côté, différentes architectures embarquées de vision ont été élaborées pour être déployées en réseau afin d'assurer une basse consommation et de diminuer le coût global du système, en particulier, quand le nombre des noeuds du réseau est grand. Cependant, peu de travaux se sont intéressés au développement de systèmes fonctionnant en temps réel sur ces architectures embarquées à cause de la difficulté du problème. En effet, les algorithmes de vision sont complexes et les architectures embarquées ont de faibles ressources de calcul et de mémoire. Dans ce travail de thèse, l'objectif est de concevoir un système de suivi multi-objet pour un réseau de caméras avec des champs de vision non recouvrants, qui peut fonctionner sur des noeuds embarqués de faible coût et de basse consommation. En effet, chaque noeud du réseau est une caméra intelligente ayant une architecture légère à ressources de calcul limitées qui effectue un traitement proche du capteur. Chaque caméra exécute un système de suivi multi-objet en temps réel qui a pour but de détecter et de suivre des personnes. L'architecture du réseau est une

architecture distribuée dans laquelle chaque caméra communique avec ces proches voisines, ce qui permet d'envisager la surveillance sur de larges régions.

Contributions

La thèse est divisée en deux parties : le suivi mono-caméra et le suivi multi-caméra. La première partie consiste à concevoir une chaîne de suivi multi-objet pour une architecture de vision embarquée. La chaîne est composée de la détection d'objets à base de la soustraction de fond et du suivi en utilisant le filtre *Gaussian Mixture Probability Hypothesis Density* (GMPHD). La deuxième partie consiste en l'extension de l'approche pour assurer le suivi dans un réseau de caméras avec des champs de vision non recouvrants. Tous les développements et les choix algorithmiques effectués dans la thèse prennent en considération les contraintes liées à l'architecture embarquée : faible empreinte mémoire et faible nombre d'opérations. Les contributions apportées dans cette thèse sont les suivantes :

1. Après une analyse des algorithmes de soustraction de fond issus de l'état de l'art, nous avons complété la méthode Zipfian Sigma Delta en lui ajoutant l'information du gradient de l'image. Cette modification permet d'améliorer la qualité de segmentation des objets en mouvement tout en gardant une faible complexité calculatoire.
2. Après l'analyse des méthodes de suivi multi-cible, le filtre GMPHD a été retenu comme l'algorithme assurant le meilleur compromis entre la complexité calculatoire et la qualité de suivi. Nous avons réalisé une analyse sur la dégradation en performance de suivi et sur le gain en vitesse d'exécution et en mémoire pour une implémentation embarquée du filtre GMPHD où certaines opérations sont approximées et où le codage en virgule fixe est utilisé. L'expérimentation est réalisée sur des données de simulation. Ce travail a permis de montrer que, même ce filtre de l'état de l'art qui est vu comme une méthode complexe, peut fonctionner sur des processeurs embarqués. Les résultats de ce travail ont été publiés dans la conférence Fusion [Dziri et al., 2014].
3. L'utilisation du filtre GMPHD pour le suivi d'objets dans les systèmes visuels a montré des limitations dues aux occultations fréquentes qui se produisent entre objets. Nous avons développé des modules de détection et de gestion d'occultations afin d'améliorer les performances de suivi, spécialement dans le cas de suivi de personnes. Ces modules ont permis de résoudre les occultations sans introduire une grande complexité du calcul. Les résultats de ce travail ont été publiés dans la conférence VISAPP [Dziri et al., 2015a].
4. La chaîne de suivi multi-objet, composée de la méthode de soustraction de fond et du suivi à base du filtre GMPHD amélioré, a été optimisée pour fonctionner en temps réel sur une caméra intelligente embarquée de faible coût. La caméra intelligente est composée de la plate-forme Raspberry Pi version 1 et du module caméra RaspiCam. Le suivi multi-objet s'exécute en temps réel sur la caméra intelligente. Les résultats obtenus ont été acceptés pour publication dans la conférence ICDSC [Dziri et al., 2015b].
5. L'approche de suivi multi-objet dans un réseau de caméras avec des champs de vision non recouvrants a été développée en tenant compte de la nature probabiliste

du filtre GMPHD. L'approche est basée sur un traitement distribué dans le but de réduire le nombre de communications entre les noeuds et de diminuer les besoins en bande passante. L'approche proposée a été validée par des simulations et les résultats obtenus sont prometteurs pour son fonctionnement dans un réseau réel de caméras intelligentes.

Structure du document

Le document est structuré comme suit :

— Chapitre 1 : L'objectif du chapitre est de donner une vue globale sur ce qui se fait dans le domaine de suivi d'objets par vision et de montrer les différents axes de recherches qui correspondent à la problématique de la thèse. Un état de l'art et une classification des méthodes et des architectures embarquées utilisées dans les systèmes de suivi visuel sont présentés. Le chapitre est structuré en trois sections : suivi mono-caméra, suivi multi-caméra et architectures embarquées de vision.

— Chapitre 2 : L'objectif du chapitre est de fournir une vue globale du système proposé pour le suivi multi-objet dans un réseau de caméras avec des champs non recouvrants. Tout d'abord, le scénario d'étude et les contraintes du système sont définies. L'analyse de l'état de l'art est réalisée dans le but de comparer les familles des méthodes et des architectures de calcul suivant le scénario d'étude et les contraintes définies du système. Finalement, la vue globale du système est fournie en se basant sur le résultat de l'analyse.

— Chapitre 3 : L'objectif du chapitre est de présenter la méthode de détection d'objets en mouvement basée sur la soustraction de fond. Tout d'abord, les méthodes de soustraction de fond de l'état de l'art sont analysées en se basant sur leur performance de segmentation et leur temps d'exécution. La méthode Zipfian Sigma-Delta retenue comme une bonne candidate pour les systèmes embarqués est détaillée. Dans le but d'améliorer les performances de cette méthode, une fusion hiérarchique du résultat de segmentation avec l'information du gradient de l'image d'entrée est présentée. Finalement, les résultats de détection comparant la méthode proposée à d'autres méthodes sont présentés et discutés.

— Chapitre 4 : Ce chapitre a deux objectifs. Le premier est de sélectionner, parmi les algorithmes de suivi retenus dans le chapitre 2, celui assurant un bon compromis entre la qualité du suivi et la complexité de calcul, en se basant sur les travaux de recherche existants. La solution retenue est le filtre GMPHD. Le deuxième objectif est l'étude du filtre GMPHD en réalisant deux expérimentations. La première expérimentation a pour but de quantifier, sur des données de simulation, la dégradation de la qualité de suivi et du gain en vitesse d'exécution quand des versions approximées du filtre GMPHD sont utilisées. La deuxième expérimentation consiste à étudier le filtre dans un contexte de suivi visuel d'objets, en utilisant des bases de données, dans le but de comprendre ces limitations. Finalement, les résultats des deux expérimentations sont présentés et discutés.

— Chapitre 5 : La limitation principale du filtre GMPHD dans un système visuel de suivi d'objets est le problème des occultations. L'objectif de ce chapitre est l'amé-

lioration du filtre GMPHD en concevant une méthode de gestion des occultations se produisant entre objets. Les travaux de recherche s'intéressant à la gestion des occultations dans le filtre GMPHD sont présentés et discutés. Par la suite, la méthode proposée est détaillée. Les résultats de la comparaison du filtre GMPHD amélioré au filtre GMPHD original, sur des bases de données d'images, sont présentés et discutés.

— Chapitre 6 : La chaîne mono-caméra de suivi multi-objet, composée du détecteur développé dans le chapitre 3 et du filtre GMPHD amélioré, est optimisée et implémentée sur une caméra intelligente embarquée. L'objectif du chapitre est d'étudier la qualité de suivi et le temps d'exécution de la chaîne. La qualité de suivi est mesurée sur des bases de données d'images permettant une comparaison à des méthodes de l'état de l'art. Le temps d'exécution est mesuré sur la caméra embarquée composée de la carte Raspberry Pi version 1 et de la caméra RaspiCam. Les résultats obtenus montrent que le traitement en temps réel est atteint sur la caméra embarquée pour une qualité de suivi proche de celle des méthodes de l'état de l'art.

— Chapitre 7 : L'approche probabiliste de suivi multi-objet dans un réseau de caméras avec des champs de vision non recouvrants est présentée. Tout d'abord, les méthodes de la littérature utilisant un formalisme probabiliste pour modéliser le problème de suivi inter-caméra sont analysées. Par la suite, l'approche proposée est détaillée en expliquant son principe et la manière dont la probabilité de correspondance inter-caméra est calculée. Par la suite, le scénario de simulation utilisé pour la validation de l'approche est détaillé en présentant la topologie du réseau et les différents paramètres contrôlables dans le simulateur. Les résultats obtenus en simulation démontrent la faisabilité de l'approche, ce qui est prometteur pour le passage à un réseau de caméras réel. Afin de se positionner dans un contexte embarqué contraint en bande passante, en mémoire et en calcul, la complexité de l'approche est analysée.

— Conclusions et Perspectives : L'objectif du chapitre est de conclure le travail de cette thèse et de présenter les différentes perspectives du travail réalisé.

Chapitre 1

État de l'art

Sommaire

1.1 Introduction

Le suivi visuel d'objets est un domaine de recherche très actif. Il présente plusieurs challenges et traite toute la chaîne allant des architectures matérielles aux applications, en passant par les algorithmes de détection, de classification et de suivi d'objets, les bases de données et les métriques d'évaluation ainsi que les réseaux de caméras. Suite à ces nombreux challenges, il existe plusieurs états de l'art dans la littérature [Hu et al., 2004, Valera and Velastin, 2005, Yilmaz et al., 2006, Kim et al., 2010, Maggio and Cavallaro, 2011, Kristan et al., 2013, Vezzani et al., 2013, Kristan et al., 2013, Wang, 2013, Song et al., 2013, Luo et al., 2014]. Chacun de ces états de l'art se focalise sur une ou plusieurs classes des problèmes rencontrés dans la conception d'un système de suivi d'objets. L'objectif de ces états de l'art est de classifier et d'évaluer les méthodes utilisées dans les systèmes de suivi d'objets.

Dans ce chapitre, nous nous intéressons aux trois axes qui définissent la problématique de la thèse : le suivi d'objets en utilisant une seule caméra, le suivi d'objets dans un réseau de caméras et les architectures de vision embarquées susceptibles d'être utilisées dans un système de suivi d'objets. La première section du chapitre présente les différentes briques constituant une chaîne générique de suivi d'objets dans le cas d'une seule caméra. Nous nous intéressons particulièrement aux algorithmes constituant les deux briques de base

(la détection et le suivi) et aux différentes caractéristiques utilisées pour la description d'objets. Dans la deuxième section, les méthodes utilisées pour le suivi d'objets dans un réseau de caméras sont étudiées pour les deux configurations : caméras avec des champs de vision recouvrants et caméras avec des champs de vision non recouvrants. Dans la troisième section, les architectures embarquées de vision sont présentées en s'intéressant à leurs caractéristiques. Nous finissons le chapitre par une conclusion.

1.2 Suivi mono-caméra

Une chaîne générique de suivi visuel d'objets mono-caméra est composée de deux étapes principales : détection et suivi [Yilmaz et al., 2006]. L'étape de détection permet de localiser les objets d'intérêt dans l'image et d'extraire leurs caractéristiques. L'étape de suivi permet de générer les trajectoires des objets, au fil du temps. Les deux étapes peuvent être réalisées séparément ou conjointement. Dans la réalisation séparée, l'algorithme de détection s'exécute sur chaque image de la séquence vidéo. Dans ce cas, l'algorithme de suivi a pour rôle de relier les résultats de détection obtenus sur deux images successives, afin de construire la trajectoire de l'objet. Dans la réalisation conjointe des deux étapes, l'algorithme de détection s'exécute seulement quand un objet apparaît pour la première fois dans l'image. Le résultat de détection permet d'initialiser l'algorithme de suivi, qui lui permet de suivre et de relocaliser l'objet d'intérêt dans l'image. Les chercheurs de la communauté de la vidéo surveillance considèrent une étape supplémentaire, entre les deux étapes précédentes : la classification [Hu et al., 2004, Enzweiler and Gavrila, 2009]. Dans ce contexte, la détection est considérée comme un pré-traitement qui permet de localiser tous les objets en mouvement, dans la scène. L'étape de classification a pour but de déterminer la classe de chaque objet détecté (humain, voiture, autre objet, *etc*). Cependant, quand aucun pré-traitement n'est effectué, la classification peut être considérée comme une méthode de détection, ce qui est le cas dans ce chapitre. Dans ce chapitre, nous considérons le cas d'une chaîne générique composée de deux étapes : détection et suivi.

1.2.1 Détection

Les algorithmes de détection peuvent être classifiés en deux catégories principales : détection de mouvement et détection par apprentissage.

1.2.1.1 Detection de mouvement

L'objectif de cette méthode est de localiser dans l'image tous les objets en mouvement. Elle peut s'effectuer avec deux méthodes différentes : la soustraction de fond et le flux optique.

Soustraction de fond

La soustraction de fond a pour objectif de segmenter les objets en mouvement qui sont présents dans la scène. Elle est utilisable pour le cas de caméras fixes. Afin de réaliser la segmentation de mouvement, un modèle de l'arrière plan de la scène est nécessaire.

Chaque image de la séquence vidéo est comparée au modèle de l'arrière plan. Les pixels de l'image présentant une différence significative par rapport au modèle sont considérés comme appartenant aux objets en mouvement. La sortie d'un algorithme de soustraction de fond est une image binaire.

Suivant le modèle utilisé pour décrire l'arrière plan, un grand nombre de méthodes ont été développées. La plus basique consiste à prendre l'image de l'arrière plan comme modèle [Benezeth et al., 2008]. Les pixels des objets en mouvement sont détectés en appliquant un seuillage sur la différence absolue entre l'image de l'arrière plan et l'image courante. La mise à jour du modèle s'effectue avec un filtre linéaire auto régressif d'ordre 1, comme l'indique l'équation (1.1).

$$BG_k(x,y) = (1-\alpha)BG_{k-1}(x,y) + \alpha I_k(x,y) \tag{1.1}$$

$BG_k(x,y)$ est la valeur du pixel ayant les coordonnées x, y dans l'image de l'arrière plan à l'instant k. $I_k(x,y)$ est la valeur du pixel ayant les coordonnées x, y dans l'image courante à l'instant k. α est un paramètre compris entre 0 et 1, qui permet de contrôler la vitesse de mise à jour du modèle. Dans le cas où $\alpha = 1$, nous retrouvons le cas spécial de différence entre deux images successives. Pour $\alpha = 0$, le modèle d'arrière plan reste constant et ne se met pas à jour.

Certaines méthodes utilisent l'image de l'arrière plan comme modèle, mais la mise à jour du modèle ne s'effectue pas comme décrit dans l'équation (1.1). Parmi ces méthodes, nous trouvons le médian adaptatif [McFarlane and Schofield, 1995] qui incrémente ou décrémente la valeur d'un pixel du modèle suivant le signe de la différence entre le modèle et l'image courante, le médian temporel [Cucchiara et al., 2001] qui prend pour chaque pixel la valeur médiane sur les N dernière images de la séquence vidéo et le sigma-delta [Manzanera and Richefeu, 2007] qui a été élaboré pour les calculateurs légers. Dans [Casares et al., 2010, Casares and Velipasalar, 2010], un pixel de l'arrière plan est mis à jour avec l'équation (1.1) en adaptant le paramètre α suivant la stabilité du pixel. La stabilité est déterminée en comptant le nombre de fois que le pixel a basculé de l'état appartenant à l'arrière plan à l'état appartenant à un objet ou l'inverse, au cours des N dernières images.

Les méthodes représentant l'arrière plan par une image sont sensibles aux variations d'éclairage. Afin de remédier à ce problème, des modèles utilisant une distribution statistique par pixel ont été développés. Dans [Wren et al., 1997], chaque pixel du modèle est représenté par une densité de probabilité Gaussienne définie par la couleur moyenne du pixel et une covariance liée à cette couleur. La comparaison de l'image courante au modèle s'effectue avec une distance en log de vraisemblance (*log-likelihood*) ou en utilisant la distance de Mahalanobis [Benezeth et al., 2008]. La moyenne et la variance sont mises à jour à chaque nouvelle image, en utilisant le même principe décrit par l'équation (1.1). Afin de rapprocher au mieux la vraie densité de l'arrière plan, [Stauffer and Grimson, 1999] ont utilisé un mélange de Gaussiennes au lieu d'une seule Gaussienne. Différentes variantes de la modélisation en mélange de Gaussiennes ont été présentées dans [KaewTraKulPong and Bowden, 2002, Zivkovic, 2004,

El Baf et al., 2008b, Zhao et al., 2012]. La modélisation des pixels de l'arrière plan avec une distribution statistique connue permet une représentation compressée de l'information statistique. Cependant, cette compression engendre des pertes d'information quand la distribution utilisée ne correspond pas à la distribution réelle.

Afin de surpasser cette limitation, la modélisation de l'arrière plan avec des distributions non-paramétriques a été utilisée. [Elgammal et al., 2000] utilisent l'estimation par noyau de la densité de probabilité pour chaque pixel des modèles (*Kernel density estimation*), ce qui revient à déduire la densité de probabilité associée à chaque pixel à partir d'un ensemble d'échantillons. VuMeter [Goyat et al., 2006] est une méthode soustraction de fond basée sur une estimation discrète de la distribution de probabilité de chaque pixel du modèle. Dans [Godbehere et al., 2012], la densité de probabilité de chaque pixel est modélisée par un histogramme. La comparaison de l'image courante au modèle de l'arrière plan s'effectue en utilisant l'inférence Bayésienne.

Certains chercheurs ont traité le problème différemment en proposant d'autres méthodes. Dans [Oliver et al., 2000], les auteurs utilisent une modélisation à base de vecteurs propres, en appliquant l'analyse en composantes principales sur l'ensemble des images d'arrière plan. Dans [Barnich and Van Droogenbroeck, 2011] la méthode proposée est appelée ViBe et consiste à modéliser chaque pixel par N échantillons pris des images précédentes. Un pixel de l'image courante est classifié comme appartenant à un objet en mouvement, si le nombre d'échantillons du modèle se trouvant à l'intérieur d'un cercle de rayon R centrée sur le pixel, est supérieur à un nombre défini. La mise à jour de chaque pixel du modèle se fait en remplaçant un échantillon sélectionné aléatoirement par le pixel de l'image courante. En s'intéressant au problème de détection d'objet, plutôt qu'à la segmentation de mouvement, [Javed et al., 2002] ont proposé une méthode qui utilise deux modèles d'arrière plan. Chaque pixel du modèle est représenté par un mélange de Gaussiennes. Un des modèles est construit à partir de l'amplitude et l'orientation du gradient. L'autre modèle est basé sur l'information couleur. Chaque modèle est appliqué séparément pour la segmentation au niveau pixel. Les résultats obtenus sont fusionnés au niveau région pour décider si la région est un objet pertinent.

La figure 1.1, présente quelques exemples de soustraction de fond issus de [Benezeth et al., 2008]. La qualité, la complexité et le besoin en mémoire de la soustraction de fond sont directement liés à la méthode utilisée. Des comparaisons des méthodes sont effectuées dans [Benezeth et al., 2008, Barnich and Van Droogenbroeck, 2011, Sobral and Vacavant, 2014]. Des implémentations de quelques méthodes ont été réalisées par Sobral [Sobral, 2013] [1]. La principale limite de la soustraction de fond est qu'elle n'est utilisable que quand la caméra est fixe.

Afin de transformer le résultat de la soustraction de fond en détection, les algorithmes d'analyse des composantes connectées sont utilisés. Un filtrage est souvent appliqué avant l'analyse des composantes connectées dans le but de réduire le bruit de segmentation. Les

1. https ://github.com/andrewssobral/bgslibrary

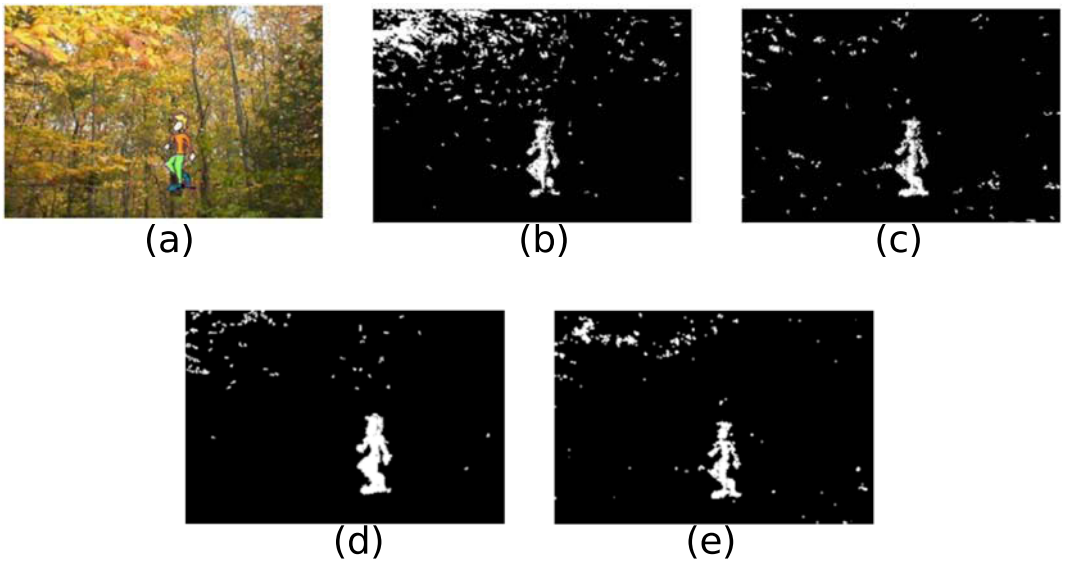

FIGURE 1.1 – Exemples de soustraction de fond [Benezeth et al., 2008]. (a) : image originale, (b) : méthode basique, (c) : une seule Gaussienne, (d) : mélange de Gaussiennes, (e) : estimation par noyau.

algorithmes d'analyse des composantes connectées permettent d'extraire, à partir d'une image binaire, les caractéristiques liées aux régions formées par les pixels connectés. Les caractéristiques décrivant une région peuvent être la boîte englobant la région, son centre de gravité, sa surface, l'histogramme, *etc.*

Flux optique

Contrairement à la soustraction de fond, le flux optique ne se restreint pas aux caméras fixes. Le flux optique est, par définition, le motif du mouvement apparent des objets, des surfaces et des contours dans une scène visuelle, causé par le mouvement relatif entre un observateur (oeil ou caméra) et la scène [Burton and Radford, 1978]. Autrement dit, le flux optique a pour but d'estimer le vecteur de vitesse de chaque pixel (flux optique dense) ou bloc de pixels, en utilisant l'information spatio-temporelle fournie par une séquence vidéo. Le flux optique est utilisé dans un grand nombre d'applications : reconstruction 3D de la scène, segmentation de mouvement, compensation de mouvement ou calcul de la disparité stéréo [Beauchemin and Barron, 1995]. Cependant, son utilisation dans les systèmes de suivi d'objets reste minime comparée à la soustraction de fond. L'hypothèse de base permettant le calcul du flux optique est que l'intensité lumineuse de l'image reste constante, sous l'effet de mouvement, entre les instants t et $t + \Delta t$ [Beauchemin and Barron, 1995, Fleet and Weiss, 2006], comme indiqué dans l'équation (1.2).

$$I(x, y, t) \simeq I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{1.2}$$

$I(x, y, t)$ est la valeur de l'intensité du pixel ayant les coordonnées (x, y) à l'instant t. Δx et Δy sont, respectivement, les déplacements suivant les axes x et y de l'image.

En approximant le deuxième terme de l'équation (1.2) par sa série de Taylor d'ordre 1, nous obtenons l'équation (1.3).

$$I_x V_x + I_y V_y \simeq -I_t \tag{1.3}$$

Avec $I_x = \frac{\delta I}{\delta x}$, $I_y = \frac{\delta I}{\delta y}$, $I_t = \frac{\delta I}{\delta t}$, $V_x = \frac{\delta x}{\delta t}$ et $V_y = \frac{\delta y}{\delta t}$. V_x et V_y représentent les vitesses de déplacement suivant les axes x et y de l'image.

L'équation (1.3) possède deux inconnues, ce qui nécessite une contrainte supplémentaire pour la détermination du flux optique. Selon la contrainte additionnelle, plusieurs méthodes de calcul du flux optique ont été développées. Dans [Beauchemin and Barron, 1995], les auteurs ont classifié les méthodes en cinq catégories : méthodes différentielles, méthodes fréquentielles ou énergétiques, méthodes basées sur la corrélation, méthodes *multiple motion* et méthodes utilisant le raffinement temporel. Dans cet état de l'art, seules les méthodes les plus utilisées dans la littérature sont présentées : les méthodes différentielles et celles basées sur la corrélation. Des états de l'art sur le flux optique peuvent être trouvés dans [Beauchemin and Barron, 1995, Fleet and Weiss, 2006]. Dans [Liu et al., 1998], plusieurs algorithmes de flux optique ont été comparés suivant la qualité des résultats et le temps d'exécution.

Les méthodes différentielles déterminent le flux optique en utilisant l'information spatio-temporelle. Elles utilisent l'équation (1.3) et introduisent une contrainte de lissage sur le gradient ou le laplacien du champ de vitesse. Les méthodes différentielles peuvent être globales ou locales selon la contrainte supplémentaire. Dans les méthodes globales, la contrainte est imposée sur toute l'image. En effet, dans [Horn and Schunck, 1981], la contrainte supplémentaire est imposée sur l'amplitude du gradient du flux optique, et est appliquée à tous les pixels de l'image. Le flux optique est déterminé en minimisant l'équation (1.4).

$$\int\int (I_x V_x + I_y V_y + I_t)^2 + \alpha^2(||\nabla V_x||^2 + ||\nabla V_y||^2) dx dy \tag{1.4}$$

∇g est le gradient de $g(x, y) = (\frac{\partial g(x,y)}{\partial x}, \frac{\partial g(x,y)}{\partial y})$

Les auteurs de [Lucas et al., 1981] ont développé la méthode Lucas-Kanade de calcul du flux optique. Cette méthode est l'une des méthodes les plus utilisées dans la littérature. C'est une méthode différentielle locale qui impose une contrainte sur le voisinage du pixel courant. En effet, les auteurs considèrent que la vitesse d'un voisinage spatial, défini par une fenêtre, est constante. En appliquant cette contrainte à l'équation (1.3), un système de plusieurs équations avec deux inconnues est obtenu. La solution de ce système est déterminée avec la méthode des moindres carrés, ce qui revient à minimiser l'équation (1.5).

$$\sum_{q \in \Omega} W(q)^2 (I_x(q) V_x + I_y(q) V_y + I_t(q))^2 \tag{1.5}$$

Ω est la fenêtre spatiale, q est un pixel qui appartient à la fenêtre spatiale et $W(q)$ est une fonction de pondération.

Les méthodes se basant sur la corrélation déterminent le flux optique en utilisant la correspondance de modèles. Dans ce type de méthodes, un bloc donné dans l'image précédente est recherché dans l'image courante afin de déterminer son vecteur vitesse. La mise en correspondance entre blocs peut être mesurée avec des métriques telles que : la somme des différences absolues (équation (1.6)), la somme de différences quadratiques (équation (1.7)) ou le coefficient de corrélation (équation (1.8)).

$$SAD = \sum_{u=-h}^{h} \sum_{v=-w}^{w} w(u,v)|I(u,v,t) - I(x+u, y+v, t+\Delta t)| \tag{1.6}$$

$$SSD = \sum_{u=-h}^{h} \sum_{v=-w}^{w} w(u,v)(I(u,v,t) - I(x+u, y+v, t+\Delta t))^2 \tag{1.7}$$

$w(.,.)$ est une fonction de pondération, h est la hauteur du bloc et w est la largeur du bloc.

$$corr = \int_u \int_v I(u,v,t) I(x+u, y+v, t+\Delta t) du dv \tag{1.8}$$

L'utilisation des sommes de différences quadratiques ou absolues implique de chercher le bloc dont la distance de correspondance est minimale. L'utilisation du coefficient de corrélation consiste à trouver le bloc assurant la corrélation maximale. La recherche exhaustive consiste à parcourir toute l'image avec différentes échelles à la recherche du bloc d'intérêt. Cette méthode assure de bons résultats, cependant, son coût calculatoire est très élevé. Afin de remédier au problème, plusieurs méthodes de recherche ont été élaborées. La complexité calculatoire peut être réduite en définissant un déplacement maximal de blocs, à partir duquel, la correspondance n'est pas calculée. De plus, des stratégies de recherche comme l'algorithme 2DLOG (two dimensional logarithmic) [Jain and Jain, 1981], l'algorithme de recherche en trois étapes [Kwatra et al., 1987] ou la recherche en quatre étapes [Po and Ma, 1996], permettent de réduire le temps du calcul. Une comparaison des stratégies de recherches est réalisée dans [Jakubowski and Pastuszak, 2013].

La localisation des objets en mouvement à partir du résultat de flux optique s'effectue en appliquant un seuillage sur l'amplitude du vecteur de vitesse obtenu pour chaque pixel ou bloc de pixels, comme indiqué dans la figure 1.2. Cette procédure permet de fournir une image binaire en sortie. Un filtrage suivi d'une analyse des composantes connectées fournit les caractéristiques des objets, comme pour la soustraction de fond. Les systèmes de suivi d'objets décrits dans [Yamamoto et al., 1995, Shin et al., 2005, Aslani and Mahdavi-Nasab, 2013] utilisent le flux optique pour la détection des objets en mouvement.

1.2.1.2 Méthodes d'apprentissage

La détection par apprentissage consiste à détecter des classes d'objets dans l'image : humain, voiture, oiseau, arrière plan, *etc.* Cette détection s'effectue en deux étapes : apprentissage et classification. Durant la première étape, l'algorithme apprend les caractéristiques décrivant les classes des objets d'intérêt et construit un classifieur permettant de

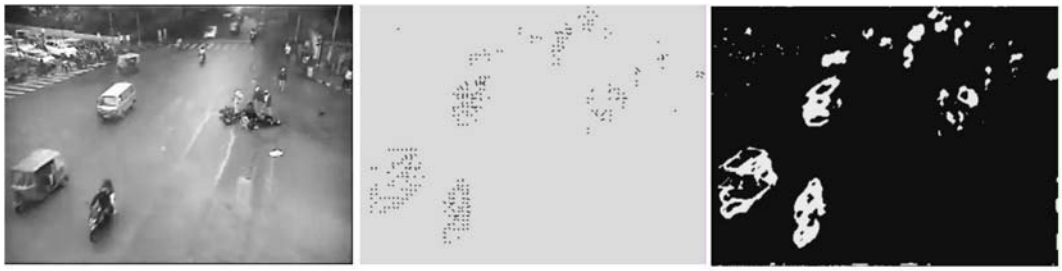

FIGURE 1.2 – Exemples de flux optique [Aslani and Mahdavi-Nasab, 2013]. (a) : image originale, (b) : flux optique, (c) : résultat du seuillage.

reconnaître ces classes dans l'image. La deuxième étape consiste à parcourir l'image d'entrée avec une fenêtre glissante à différentes échelles. Pour chaque position de la fenêtre, les caractéristiques décrivant la région de l'image sont extraites. Le classifieur prend en entrée les caractéristiques et fournit en sortie la classe qui correspond au contenu de la fenêtre. Plusieurs problématiques sont liées à la détection par apprentissage : le choix des caractéristiques, la sélection des données d'apprentissage, la construction du classifieur et les stratégies du parcours d'image.

Plusieurs types de caractéristiques ont été explorés dans la littérature pour décrire des objets. Les réponses des ondelettes de Haar ont été utilisées comme descripteur de personnes dans [Papageorgiou and Poggio, 2000] et de visages dans [Viola and Jones, 2004]. D'autres approches utilisent les points d'intérêt comme caractéristiques. *Speeded Up Robust Features* (SURF) [Bay et al., 2006] est utilisé dans [Li and Zhang, 2013] pour la description de visages et de voitures. Une comparaison des descripteurs SURF, *Binary Robust Invariant Scalable Keypoints* (BRISK) [Leutenegger et al., 2011] et *fast Retina Keypoint* (FREAK) [Alahi et al., 2012] a été réalisée, dans [Schaeffer, 2013], pour le contexte de détection de personnes. D'autres approches sont basées sur la description de personnes avec l'information de gradient [Dalal and Triggs, 2005, Sabzmeydani and Mori, 2007, Lin and Davis, 2008]. Le descripteur le plus connu dans ces approches est les histogrammes d'orientation de gradients (HOG) [Dalal and Triggs, 2005]. L'information de la texture déterminée par Local Binary Patterns (LBP) est utilisée comme descripteur de visage dans [Ahonen et al., 2006]. Dans [Wojek and Schiele, 2008, Gavrila and Munder, 2007, Wang et al., 2009b, Harzallah et al., 2009, Schwartz et al., 2009, Tang et al., 2012], le descripteur est construit en combinant différentes caractéristiques. D'autres approches [Zhu et al., 2010, Felzenszwalb et al., 2010b] décrivent un objet avec la combinaison des caractéristiques locales correspondant à ses différentes parties, par exemple, les différentes parties d'un humain sont le torse, les bras, la tête et les jambes.

Les classifieurs les plus rencontrés dans la littérature sont les machines à vecteurs de support (SVM) [Papageorgiou and Poggio, 2000, Dalal and Triggs, 2005, Felzenszwalb et al., 2010b, Zhu et al., 2010] et l'Adaboost [Viola and Jones, 2004, Sabzmeydani and Mori, 2007, Li and Zhang, 2013]. Dans le cas des SVM, l'apprentissage

d'un classifieur permettant la discrimination entre deux classes est réalisé en déterminant la frontière de séparation maximisant la distance entre cette frontière et les échantillons d'apprentissage les plus proches. L'apprentissage par l'Adaboost s'effectue d'une manière itérative. À chaque itération l'Adaboost détermine un classifieur faible qui raffine le classifieur global. Durant la première itération, les échantillons d'apprentissage contribuent tous avec le même poids dans la conception du classifieur faible. À l'itération suivante, les poids des échantillons changent de manière à se concentrer sur les échantillons mal classifiés précédemment, pour la détermination du nouveau classifieur faible.

Les autres classifieurs que SVM et Adaboost sont moins utilisés dans les systèmes de suivi d'objets. Nous citons les réseaux de neurones [Szarvas et al., 2005, Szegedy et al., 2013] et les *Random Forest* [Tang et al., 2012, Gall et al., 2012]. Un état de l'art détaillé sur les méthodes de détection d'objets par apprentissage est présenté dans [Zhang et al., 2013]. Des comparaisons des méthodes de détection de personnes sont présentées dans [Enzweiler and Gavrila, 2009, Dollar et al., 2012].

Afin de rendre la détection robuste au changement de la taille de l'objet (changement d'échelle), l'image doit être parcourue avec une fenêtre glissante à différentes échelles. Le changement d'échelle dans une image est réalisé en construisant la pyramide de Gaussiennes. Étant donnée que la fenêtre glissante est appliquée à chaque niveau de la pyramide, le nombre total de fenêtres résultantes est élevé. Pour une image de résolution $n \times n$, le nombre de fenêtres est proportionnel à n^2, ce qui signifie que pour une image de taille 320×240 le nombre de fenêtres dépasse un milliard [Lampert et al., 2008]. Un parcours exhaustif ne permet donc pas d'atteindre le temps réel même sur les récentes machines de calcul effectuant des traitements séquentiels [Dollár et al., 2010]. Afin de remédier à ce problème, des approches permettant un balayage efficace de l'image ont été développées dans [Lampert et al., 2008, An et al., 2009] et des implémentations sur des machines de calcul parallèle comme les *Graphic Processor Unit* (GPU) ont été réalisées dans [Prisacariu and Reid, 2009, Benenson et al., 2012].

1.2.2 Descripteurs

Dans le suivi d'objets, un objet est décrit par la forme et les caractéristiques qui le représentent. Le choix des deux paramètres est important pour la construction d'un descripteur pertinent, dans un contexte donné. Les descripteurs des objets sont l'entrée de l'algorithme de suivi.

Dans un système de vision, un objet peut être représenté par différentes formes, comme indiqué dans la figure 1.3. La forme la plus simple est le point [Wang et al., 2006]. Les formes géométriques paramétrables comme le rectangle [Yang et al., 2005a, Maggio et al., 2008, Yoon et al., 2015] et l'ellipse [Kuo et al., 2010b, Zuriarrain et al., 2013, Poiesi and Cavallaro, 2015] sont souvent utilisées dans la littérature à cause de la simplicité de leur modèle (forme rigide). Des formes géométriques non paramétrables comme le contour et la silhouette de l'objet permettent une localisation plus précise des objets non rigides dans l'image [Haritaoglu et al., 1999, Yilmaz et al., 2004, Sun et al., 2011]. Un objet peut aussi être représenté par un ensemble de points d'inté-

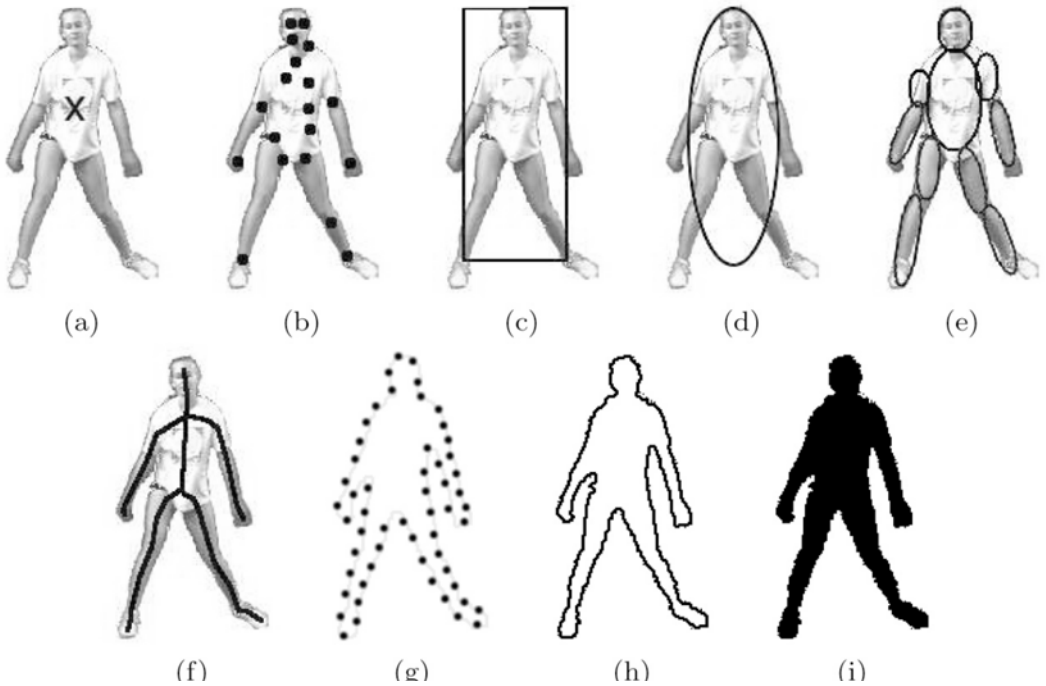

FIGURE 1.3 – Formes représentant un objet dans un système de suivi [Yilmaz et al., 2006]. (a) : un point, (b) : points d'intérêt, (c) : rectangle, (d) : ellipse, (e) : multiple blocs, (f) : squelette, (g)-(h) contour, (i) : Silhouette.

rêt [Shi and Tomasi, 1994, Coifman et al., 1998, Shin et al., 2005], des multiples blocs ou un squelette [Balan and Black, 2006, Andriluka et al., 2009]. Selon le type de la forme utilisée, l'algorithme de suivi est différent.

Pour décrire un objet, plusieurs caractéristiques peuvent être utilisées, dont celles décrites dans la section 1.2.1.2. Les caractéristiques descriptives peuvent être regroupées en deux catégories : apparence (modélisant l'apparence visuelle de l'objet) et mouvement (modélisant le déplacement de l'objet). L'apparence visuelle d'un objet peut être modélisée avec la couleur ou l'intensité [Comaniciu et al., 2000, Nummiaro et al., 2002, Zivkovic and Krose, 2004], le gradient [Yang et al., 2005a, Kuo et al., 2010b] ou la texture [Takala and Pietikainen, 2007, Dash et al., 2014]. Le codage de ces caractéristiques en descripteurs peut se faire de différentes manières. La manière la plus simple est l'utilisation de *template*. Un *template* est un bloc de pixels dont les valeurs sont celles de la caractéristique. Cependant, des codages plus efficaces comme l'histogramme global, multiple histogrammes locaux (histogramme par région de l'objet), *bag-of-words*, des distributions statistiques (Gaussienne, mélange de Gaussiennes, *etc*) ou le descripteur de covariance sont utilisés.

L'utilisation du mouvement comme une caractéristique [Cox and Hingorani, 1996, Wang et al., 2006, Poiesi and Cavallaro, 2015] consiste à décrire l'objet avec l'information spatio-temporelle. En effet, les coordonnées de la position, la vitesse, l'accélération ainsi que la taille (largeur et hauteur) des objets sont utilisées. Le vecteur contenant ces coordonnées construit le descripteurs de l'objet et est appelé vecteur d'état. Dans ce cas,

le modèle décrivant le mouvement des objets doit être connu.

Les différentes caractéristiques sont dans la plupart des cas fusionnées afin d'améliorer la qualité de suivi. Des états de l'art sur les caractéristiques et les descripteurs utilisés dans un système de suivi sont présentés dans [Yilmaz et al., 2006, Vezzani et al., 2013, Li et al., 2013, Luo et al., 2014]. Le choix des caractéristiques pertinentes pour un système de suivi multi-objet est relié au scénario de suivi, notamment à la qualité du suivi attendue et les architectures de calcul utilisées.

1.2.3 Suivi

Le suivi a pour but de générer les trajectoires des objets. Chaque objet possède une identité unique qui lui est assignée par l'algorithme de suivi. La correspondance entre les positions d'un objet dans des images successives constitue la trajectoire. Dans le cas de suivi par détection [Wang et al., 2006, Zuriarrain et al., 2013, Poiesi and Cavallaro, 2015], l'algorithme de suivi relie les résultats de détections issues de deux images successives. Dans ce cas, la détection s'effectue sur toutes les images de la séquence. Cependant, certains algorithmes de suivi nécessitent la détection seulement à l'initialisation du suivi quand un objet apparaît pour la première fois. Après initialisation, la détection et le suivi s'effectuent conjointement [Comaniciu et al., 2000, Pham et al., 2007]. Les algorithmes de suivi sont groupés en trois catégories (figure 1.4), suivant la forme des objets.

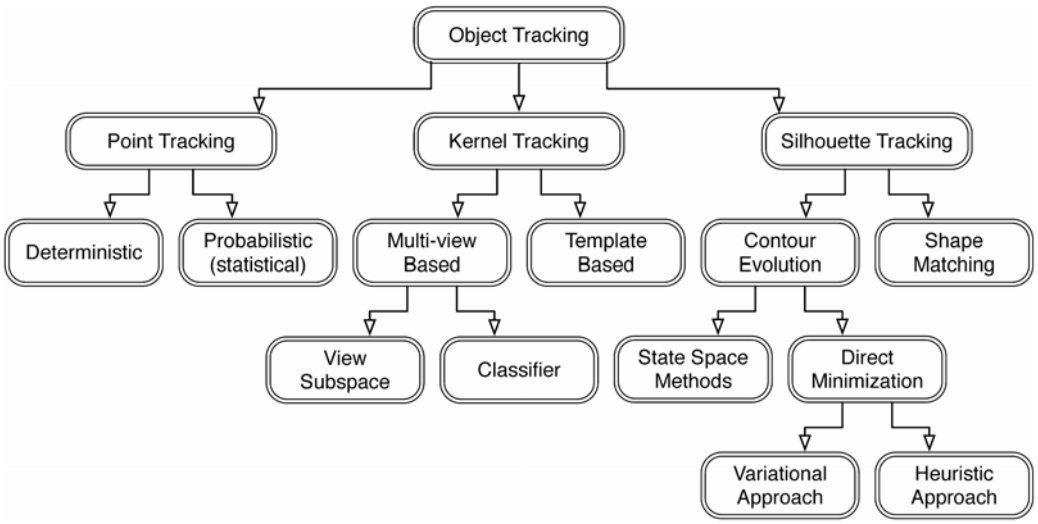

FIGURE 1.4 – Classification des algorithmes de suivi [Yilmaz et al., 2006].

Le filtre de Bayes classique [Arulampalam et al., 2002] est une méthode probabiliste utilisée pour le suivi d'un seul objet représenté avec un point. Le filtre de Bayes est approximé avec un filtre de Kalman pour les système linéaires Gaussiens et avec un filtre à particules dans le cas général. Suivre plusieurs objets, représentés avec des points, en utilisant le filtre de Bayes nécessite une méthode d'association. Les méthodes d'association ont pour but de relier les observations fournies par le détecteur aux pistes (trajectoires des objets). Les méthodes d'association peuvent être des méthodes déterministes comme la théorie des graphes [Shafique and Shah, 2005, Taj et al., 2007] ou statistique comme *Joint*

Probabilistic Data Association (JPDA) [Bar-Shalom and Li, 1995, Jaward et al., 2006] et *Multiple Hypothesis Tracker* (MHT) [Cox and Hingorani, 1996, Blackman, 2004].

Dans la théorie des graphes, les observations générées par le détecteur au cours d'un intervalle de temps (fenêtre temporelle) représentent les sommets (noeuds) du graphe. Une arête représente le coût associé au déplacement d'un objet entre deux noeuds. Les trajectoires des objets sont définies par l'ensemble des chemins entre noeuds qui minimise ou maximise une fonction de coût. Les méthodes existantes diffèrent selon la taille de la fenêtre temporelle, la fonction de coût et la méthode d'optimisation utilisée pour déterminer la solution. Un cas spécial de la théorie de graphes est la méthode *Global Nearest Neighbor* (GNN) [Blackman and Popoli, 1999]. GNN permet de déterminer la solution optimale pour le cas où la fenêtre temporelle est réduite à deux instants successifs. En se basant sur une matrice de coût reliant les observations aux pistes, l'algorithme de Munkres [Bourgeois and Lassalle, 1971] ou l'algorithme Auction [Blackman and Popoli, 1999] permettent de déterminer la solution minimisant le coût global des associations. Dans le cas de GNN, un filtre de Kalman est utilisé pour la prédiction et la mise à jour des pistes et la distance de Mahalanobis est utilisée comme fonction de coût. L'aspect déterministe de la théorie des graphes assure un nombre limité de paramètres à régler, ce qui facilite son utilisation quelque soit le scénario de suivi. La détermination de la solution optimale est d'une complexité NP quand plus de deux ensembles d'observations successifs sont considérés. Des solutions sous-optimales imposant des contraintes sur les caractéristiques des objets sont souvent utilisées dans le but de casser la complexité exponentielle en une complexité polynomiale. GNN intègre l'aspect probabiliste qui est dû à l'utilisation du filtrage de Kalman.

Les méthodes d'association statistiques intègrent l'aspect probabiliste dans la résolution du problème en modélisant les faux positifs, les faux négatifs et les interactions entre objets. JPDA met à jour l'état d'un objet en considérant toutes les observations reçues à l'instant courent. Durant la mise à jour de l'état d'une piste, chaque observation est pondérée par un poids correspondant à la probabilité de l'association de l'observation à la piste. Ce mécanisme assure une bonne robustesse aux faux positifs. JPDA considère que le nombre d'objets est connu et fixe durant tout le suivi. La méthode MHT, quant à elle, formule des hypothèses d'associations et les propage au fil du temps. La décision sur l'association optimale s'effectue d'une manière différée en se basant sur le score des hypothèses. Le nombre d'hypothèses explose de manière exponentielle, d'où l'inconvénient de la méthode. Comme dans la théorie des graphes, l'utilisation des observations d'une fenêtre temporelle assure une robustesse aux occultations.

La résolution du problème de suivi multi-objet sans recours aux méthodes d'association est effectuée en étendant le filtre de Bayes classique aux cas multi-objet. Dans ce cas, l'estimation du nombre d'objets et de leurs positions s'effectue conjointement en modélisant les observations et les pistes avec des ensembles aléatoires finis *Random finit Set* (RFS). Un RFS est un ensemble dont les élément sont des variables ou des vecteurs aléatoires et le nombre d'éléments est une variable aléatoire. Un RFS est défini par une densité de probabilité discrète modélisant le nombre d'éléments dans l'ensemble et une densité de probabilité conjointe sur les éléments de l'en-

semble (densité multi-objet). Cette extension a donné naissance aux filtres *Probability Hypothesis Density* (PHD) [Mahler, 2003, Maggio et al., 2008] et *Cardinalized PHD* (CPHD) [Mahler, 2007, Lamard et al., 2013]. Le principe de PHD et CPHD est d'approximer la densité multi-objet par son moment statistique d'ordre 1. Le filtre PHD suppose que le nombre d'objets suit une loi de Poisson, et seul le moment statistique d'ordre 1 de cette loi est propagé au fil du temps. Dans CPHD, le nombre d'objets est représenté par une densité discrète qui ne se restreint pas à la loi de Poisson. En effet, c'est toute la densité qui est propagée.

Le grand inconvénient des méthodes probabilistes est les hypothèses sur les distributions statistiques des données. Le nombre de paramètres à régler est souvent élevé comparé aux méthodes déterministes ce qui peut être considéré comme avantage et inconvénient. En effet, il peut être un inconvénient quand l'information a priori du scénario étudié n'est pas connue. Une estimation de cette information peut être fastidieuse pour l'utilisateur. L'avantage peut être perçu dans l'adaptabilité des méthodes probabilistes à un nouveau scénario en introduisant l'information a priori de ce dernier. Par exemple, le changement d'un détecteur dans le système de suivi est facilement pris en compte en introduisant son information statistique (faux positifs, faux négatifs, *etc*) dans la méthode..

Pour les objets représentés avec des formes géométriques paramétrables, le suivi s'effectue avec la correspondance des blocs (*Template Matching*) [Porikli and Tuzel, 2005], l'algorithme *Continuously Adaptive Mean-Shift* (CAMSHIFT) [Hidayatullah and Konik, 2011], ou en utilisant le filtre à particules [Arulampalam et al., 2002, Zuriarrain et al., 2013].

Le *Template Matching* consiste à déterminer pour chaque objet, la région de l'image ayant l'apparence la plus proche du *template* représentant l'objet, au sens d'une distance donnée. Cette méthode est efficace pour le suivi d'un seul objet. Cependant, pour plusieurs objets une méthode d'association est nécessaire.

CAMSHIFT est un algorithme de suivi où la détection est seulement nécessaire pour l'initialisation. Chaque objet dans CAMSHIFT est représenté par l'histogramme couleur du rectangle le délimitant. Le principe de l'algorithme est de localiser, de manière itérative, le maximum local d'une densité de probabilité en faisant converger le centre de masse, la hauteur et la largeur du rectangle délimitant l'objet vers la moyenne des pixels contenus dans le rectangle. L'avantage de cette méthode est sa simplicité à localiser un objet. Son inconvénient est la perte de l'objet dont le déplacement entre deux images successives est supérieur à sa taille [Zhang et al., 2009a].

Les filtres à particules peuvent exploiter des caractéristiques d'objets dont le bruit est non Gaussian et dont l'évolution est non-linéaire, ce qui permet un large choix de caractéristiques contrairement au filtre de Kalman. Le principe de fonctionnement est de représenter la densité de probabilité d'un objet avec un ensemble de particules. Les particules sont alors propagées et corrigées au fil du temps. Comme la densité de probabilité est souvent basée sur l'information spatiale et l'information d'apparence, la dimension des particules est élevées. La complexité de ces filtres est exponentielle avec la dimension de la particule et le nombre de particules associées à un objet.

La représentation des objets avec leur silhouette ou contour nécessite un algorithme de suivi se basant, respectivement, sur la correspondance de silhouettes entre deux images

successives (*Shape matching*) [Sato and Aggarwal, 2004] ou en déterminant l'évolution de contour (*active snake tracking*) [Kass et al., 1988, Araki et al., 2000].

L'information fournie par l'algorithme de suivi peut être utilisée pour améliorer la détection, c'est le cas dans [Kalal et al., 2012, Piao and Berns, 2014].

Les problèmes les plus courants qui influencent les performances d'un système de suivi multi-objet sont les problèmes d'occultations. L'occultation se produit à cause de la projection de la scène 3D sur le plan 2D de l'image. En effet, une occultation partielle ou totale se produit lorsque plusieurs objets sont alignés avec le centre de la caméra. Le système de suivi doit gérer les occultions afin d'assurer des trajectoires cohérentes en évitant que les identités assignées aux objets ne changent au fil du temps. Les problèmes d'occultations peuvent être gérés en fusionnant les informations issues de plusieurs caméras ayant des champs de vision recouvrants [Fleuret et al., 2008] ou avec des méthodes spécifiques pour le cas mono-caméra.

Dans le suivi mono-caméra, la résolution des occultations s'effectue durant la phase de détection [Wu et al., 2008, Gao et al., 2011] ou durant la phase de suivi [Yang et al., 2005b, Senior et al., 2006]. Il y a deux types de méthodes : *Merge-Split* (MG) et *Straight-Through* (ST). Les méthodes MS fusionnent les objets quand l'occultation se produit et re-identifient les objets à la fin de l'occultation (figure 1.5-(a)). Les méthodes ST ont pour but d'assigner chaque pixel appartenant à la région occultée à l'un des objets (figure 1.5-(b)). Une étude des méthodes existantes de résolution d'occultations est présentée dans [Gabriel et al., 2003].

1.3 Suivi multi-caméra

L'utilisation de plusieurs caméras ou de réseau de caméras, dans les systèmes de suivi multi-objet, a deux objectifs principaux : élargir la zone surveillée ou rendre les systèmes de suivi plus robustes aux occultations. Le premier objectif est atteint avec une topologie impliquant des champs de vision non recouvrants. La résolution d'occultations, quant à elle, est traitée avec des caméras ayant des champs de vision qui se recouvrent. Selon les objectifs applicatifs et la topologie du réseau de caméras, plusieurs challenges interviennent : estimation de la topologie, calibration, synchronisation, fusion de données et re-identification. Dans cette section, nous nous intéressons aux algorithmes de fusion et de re-identification utilisés dans les réseaux de caméras avec des champs recouvrants et des champs non recouvrants. La nécessité de calibration et de synchronisation d'un réseau de caméras ajoute une difficulté supplémentaire au déploiement du réseau, car pour l'ajout d'une nouvelle caméra il faut effectuer sa calibration et la synchroniser avec le reste des caméras. Les problèmes de calibrations et de synchronisations ne sont pas traités dans cette section, les lecteurs sont invités à consulter [Radke, 2010, Wang, 2013] pour plus de détails.

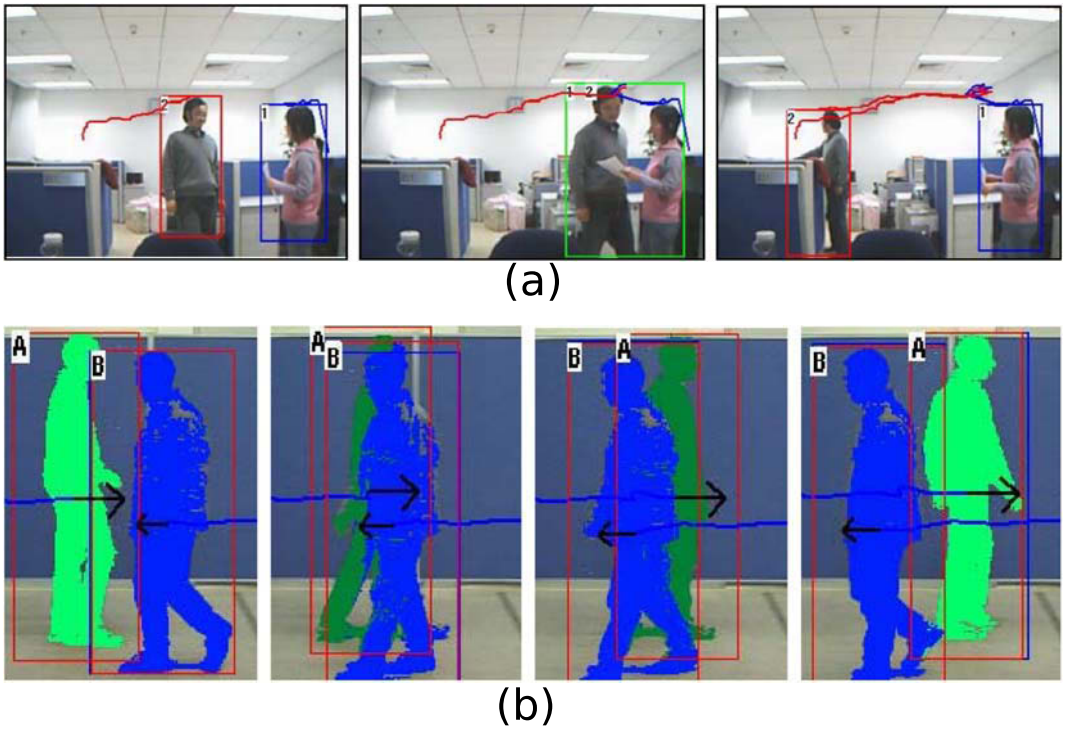

FIGURE 1.5 – Résolution d'occultations. (a) : *Merge-Split* [Yang et al., 2005b], (b) : *Straight-Through* [Cucchiara et al., 2004].

1.3.1 Fusion de données dans des caméras avec des champs recouvrants

Les caméras avec des champs de vision recouvrants permettent de résoudre les occultations objet-objet et objet-environnement en récupérant des informations sur la même scène, de différents points de vue. Cette configuration est très sensible à la calibration et à la synchronisation des caméras. En effet, pour pouvoir fusionner les informations, il faut un repère spatio-temporel commun pour toutes les caméras. La calibration permet de définir pour chaque caméra une relation entre le plan image et le monde réel qui, dans la plupart des cas, est représenté par la vue de dessus de la scène. La synchronisation quant à elle permet d'éviter la fusion d'informations capturées à différents instants. La fusion peut être centralisée ou distribuée selon la présence ou non d'un centre de fusion. La fusion est centralisée si toutes les caméras communiquent avec le centre du fusion. Dans la fusion distribuée chaque caméra communique avec ses proches voisines avant de prendre la décision. La fusion de l'information issue de plusieurs caméras peut s'effectuer à différents niveaux : bas niveau et haut niveau.

La fusion bas niveau est réalisée directement sur les résultats de segmentation des objets [Sternig et al., 2011], en particulier sur les résultats de soustraction de fond [Fleuret et al., 2008, Khan and Shah, 2006, Khan and Shah, 2009, Alahi et al., 2011, Possegger et al., 2013, Golbabaee et al., 2014]. Les résultats de soustraction du fond sont communiqués au centre du fusion. La fusion se base sur les contraintes géométriques fournies par la calibration spatiale des caméras. Dans la fusion centralisée, la taille des

messages véhiculés dans le réseau est importante car l'information transmise par chaque caméra est constituée de l'image entière. Cette approche est désavantageuse à cause de la bande passante de communication requise et de la puissance de calcul nécessaire dans le centre de fusion. Cependant, l'utilisation de l'information bas niveau minimise la perte de données. La fusion permet alors de fournir une meilleure qualité de détection. En effet, les objets sont détectés même si des occultations se présentent dans certains champs de vision de caméras.

La fusion haut niveau consiste à fusionner les résultats de détection ou les résultats de suivi fournis par chaque caméra. Dans ce cas, seulement les informations haut niveau (vecteurs d'état ou vecteur de mesure) comme les positions et les covariances des objets dans le repère réel sont nécessaires. La bande passante requise par le système est moins importante comparée à la fusion bas niveau. La fusion haut niveau peut être centralisée ou distribuée.

Dans la fusion centralisée, une association entre les pistes fournies par les caméras du réseau est réalisée dans le repère commun, pour savoir quelles sont les pistes générées par le même objet. L'association se fait avec la correspondance des caractéristiques [Wang et al., 2009a, Zhang et al., 2009b, Pham et al., 2011], ou en utilisant les contraintes géométriques comme l'homographie [Hu et al., 2006]. La fusion dans [Wang et al., 2009a] consiste en la pondération des vecteurs d'états avec l'inverse de leurs matrices de covariance. Dans [Hu et al., 2006], les résultats de fusion sont les intersections des pistes fournies par les contraintes géométriques.

La fusion distribuée est basée sur le filtre de *Consensus* [Olfati-Saber, 2005, Olfati-Saber, 2007, Kamal et al., 2012]. L'utilisation du filtre *Consensus* dans la fusion avec un Kalman permet d'approximer le filtre de Kalman centralisé avec plusieurs micro-Kalman [Olfati-Saber, 2005]. Dans la fusion distribuée avec le filtre *Consensus*, pour chaque objet, la caméra transmet les termes *Consensus* consistant en la matrice et le vecteur d'information, à toutes les caméras observant l'objet. Afin de déterminer quelle détection est associée à quel objet, une méthode d'association est utilisée par caméra. Lorsqu'une caméra reçoit les messages des caméras observant le même objet, une fusion des termes de consensus est effectuée. Les résultats de fusion sont utilisés pour estimer l'état de l'objet. La dernière étape consiste à propager le vecteur d'état et la matrice de covariance en utilisant le modèle d'état. Les travaux utilisant les filtres de *Consensus* pour le suivi d'objets dans un réseau de caméras sont [Soto et al., 2009, Song et al., 2010, Song et al., 2011, Kamal et al., 2013]. Dans [Taj and Cavallaro, 2011], les méthodes de suivi d'objets dans les réseaux de caméras sont classifiées suivant la configuration du réseau (décentralisée et distribué) et sont comparées selon les coûts de communication et de calcul ainsi que l'efficacité énergétique.

1.3.2 Ré-identification des objets dans des caméras avec des champs non recouvrants

Le champ de vision d'une seule caméra est limité. Couvrir une large région avec des caméras ayant des champs de vision recouvrants n'est parfois pas adéquat à cause du grand nombre de caméras nécessaires. L'utilisation de caméras avec des champs non recouvrants devient un choix pertinent. Les caméras dans ce cas n'observent pas la même scène et

aucune fusion d'information n'est effectuée. Dans ce type de systèmes, la problématique de suivi multi-objet consiste en la ré-identification d'objets, quand ils disparaissent d'une caméra et ils réapparaissent dans une autre caméra. Les challenges sont alors différents de ceux des caméras avec des champs recouvrants à cause de la discontinuité spatiale et temporelle dans l'observation des objets. Les informations fournies par le réseau de caméras peuvent être exploitées avec ou sans calibration du réseau [Vezzani et al., 2013] selon les méthodes et les caractéristiques utilisées. La difficulté est plus importante quand le réseau n'est pas calibré, car la description des objets doit être robuste aux différentes variations comme l'éclairage, le point de vue, la différence entre les capteurs, *etc.*

Le suivi multi-objet dans un réseau de caméras avec des champs non recouvrants se base sur l'utilisation des caractéristiques d'apparence et de l'information spatio-temporelle. L'apparence consiste à décrire l'objet avec des modèles robustes aux changements de point de vue et aux variations d'éclairage, causées par la différence des paramètres des caméras et leur positionnement dans la scène. L'information spatio-temporelle consiste à exploiter la topologie du réseau afin de déterminer le chemin le plus probable d'un objet quand il sort du champ de vision d'une caméra. Les deux approches sont souvent combinées afin que la ré-identification soit fiable. Les caractéristiques d'apparence et spatio-temporelle nécessitent un apprentissage du temps de transition d'objets, de la relation entre les paramètres des caméras (couleur, échelle, *etc*) et de la topologie du réseau.

Dans tous les travaux de suivi multi-objet dans un réseau de caméras avec des champs non recouvrants, le principe de base reste le même. Les régions d'entrées/sorties sont définies pour chaque caméra. Un suivi multi-objet est réalisé pour chaque caméra. Lorsqu'un objet quitte la caméra, les informations liées à la région de sortie et les caractéristiques de l'objet sont transmises aux caméras voisines. Afin de réduire la bande passante et de permettre un passage à l'échelle, chaque caméra ne communique qu'avec ces voisines proches. Lorsque la caméra, recevant le message, détecte un nouvel objet, la phase de ré-identification est exécutée. La ré-identification consiste à comparer les caractéristiques de l'objet détecté à celles que la caméra a reçues. L'identité du nouvel objet est définie par le résultat de la comparaison. La différence entre les systèmes existants réside dans le choix des caractéristiques décrivant les objets, la méthode de ré-identification, l'apprentissage des paramètres entre caméras et la formulation de problème. En effet, un grand nombre de travaux ont été réalisés dans [Huang and Russell, 1997, Kettnaker and Zabih, 1999, Javed et al., 2003, Dick and Brooks, 2005, Song and Roy-Chowdhury, 2008, Kuo et al., 2010a, Wang et al., 2011, Chen et al., 2013b, Chen et al., 2013a].

Dans [Javed et al., 2003] le problème de suivi inter-camera est formulé par la maximisation de la probabilité a posteriori (Maximum a Posteriori - MAP). La probabilité a posteriori dépend de la probabilité de transition entre caméras, la probabilité d'apparence et la probabilité spatio-temporelle. La probabilité spatio-temporelle est composée du temps de transition, les régions d'entrée-sortie et la vitesse d'un objet à sa sortie de la caméra. La fonction de changement de luminosité pour chaque paire de caméras est apprise et modélisée par une distribution Gaussienne dont la variable est la distance entre deux histogrammes. L'estimation par noyau de la densité de probabilité est utilisée pour déterminer la probabilité spatio-temporelle d'un objet entre les régions d'entrée-sortie. Dans [Dick and Brooks, 2005], un modèle de Markov ayant comme états les régions d'entrée-sortie des caméras est utilisé pour le suivi inter-caméra. Les transitions

entre états sont modélisées par une matrice de transition apprise à partir du mouvement d'une personne. Dans [Wang et al., 2011, Wang et al., 2014], le problème de suivi est formulé avec un réseau de Petri probabiliste. Chaque objet est décrit par un histogramme 3D de couleur, un HOG, le temps de transition entre caméras qui est modélisé avec un mélange de Gaussiennes, le ratio et la taille. La ré-identification s'effectue en sélectionnant l'objet dont le score de correspondance des caractéristiques est le plus élevé. Dans [Chen et al., 2013a], un modèle d'apparence de référence est construit pour chaque caméra. Chaque objet est décrit par ses caractéristiques biométriques : couleur des cheveux, couleur de la peau, la taille, le poids, couleur de corps, couleur de torse et couleur des jambes. Dans [Chen et al., 2013b], chaque objet est décrit par l'information couleur (histogramme de spectre de couleur dominant (*Majour color spectrum histogram*), histogramme couleur, couleurs locales dominantes, distribution spatiale des couleurs) et l'information de la texture (LBP). Afin de remédier aux problèmes de changement d'apparence d'un objet entre caméras, un classifieur est construit avec la méthode Adaboost. Le descripteur utilisé pour la construction du classifieur contient la caméra de sortie, la caméra d'entrée et la distance entre les caractéristiques d'un objet disparu de la caméra de sortie et apparaissant dans la caméra d'entrée. Le classifieur est utilisé pour la ré-identification d'objets. Dans [Alahi et al., 2010], une cascade de descripteurs a été développée pour assurer le suivi dans un réseau de caméras quelconque (aucune calibration n'est nécessaire et les caméras peuvent être mobiles). Dans [Cong et al., 2010], un descripteur basé sur la couleur et la position est utilisé pour la ré-identification de personnes dans un réseau de caméras avec des champs non recouvrants.

1.4 Architectures embarquées de vision

Les architectures embarquées de vision sont diverses et variées, selon leur puissance de calcul, la consommation énergétique, la taille, le coût et la complexité de leur programmation. La meilleure plate-forme est celle qui offre une grande puissance de calcul, consomme peu d'énergie, a une petite taille, a un faible coût et est facilement programmable. Afin d'atteindre ces performances, des architectures assurant un traitement très proche du capteur ont été développées. Ces plates-formes sont appelées caméras intelligentes (*Smart Cameras*).

Une caméra intelligente est composée d'un imageur relié à une unité de traitement et des mémoires. Dans la plupart des caméras intelligentes existantes, une unité supplémentaire assurant la communication est reliée à l'unité de traitement. L'imageur peut être un capteur CCD (*Charge Coupled Device*) [Fleck et al., 2006], un capteur CMOS (*Complementary Metal Oxide Semiconductor*) [Dias et al., 2007, Winkler and Rinner, 2010] ou un capteur événementiel [Litzenberger et al., 2007]. La performance de la caméra intelligente est liée à l'architecture de l'unité de traitement et à la bande passante entre l'imageur et l'unité de traitement. L'unité de traitement peut être composée d'un FPGA (*Field Programmable Gate Array*) [Mosqueron et al., 2007, Dias et al., 2007], un processeur SIMD (*Single Instruction Multiple Data*) [Kleihorst et al., 2006], un micro-contrôleur [Hengstler et al., 2007, Magno et al., 2008], un DSP (*Digital Signal Processor*) [Arth et al., 2007, Yan et al., 2013] ou une plate-forme multi-

coeur [Bramberger et al., 2006, Fleck et al., 2006]. La quantité de mémoire dans une caméra intelligente varie de quelques centaines de kilo octets (ko) [Rahimi et al., 2005] à quelques centaines de Mega octets (Mo) [Bramberger et al., 2006]. Le coût de la plateforme est directement liée à ses composantes. La puissance consommée par les caméras intelligentes varie de quelques milliwatt (mW) [Rahimi et al., 2005] à quelques watt (W) [Bramberger et al., 2006]. La programmation des caméras intelligentes peut se faire avec du VHDL ou du Verilog quand l'unité de traitement est un FPGA et en assembleur ou en langage haut niveau (C, C++, python) [Rowe et al., 2007, Yan et al., 2013] pour les processeurs programmables. De nombreuses caméras intelligentes ont été développées. Le tableau 1.1 présente quelques caméras ainsi que leurs performances. Des comparaisons des caméras existantes ont été effectuées dans [Rinner et al., 2008, Seema and Reisslein, 2011, Wu and Hong, 2013].

L'utilisation d'un FPGA comme unité de traitement, permet une bonne accélération des algorithmes, en exploitant la parallélisation et l'optimisation au niveau bit. Un FPGA est adéquat pour les traitements bas niveau réguliers qui opèrent au niveau pixel, comme la soustraction de fond, le filtrage morphologique ou la correspondance de blocs (*Template Matching*). Cependant, son utilisation n'est pas efficace pour les algorithmes séquentiels irréguliers, comme le suivi basé sur le filtre de Bayes. Le codage en virgule flottante et les opérations de haut niveau comme l'exponentielle, l'inversion matricielle ou le logarithme sont coûteuses sur un FPGA car ils nécessitent le développement d'unités spécialisées qui nécessitent beaucoup de ressources du FPGA. Le coût et la consommation d'un FPGA dépendent du modèle utilisé mais, dans la plupart des cas, ils sont plus élevés comparés à ceux d'un micro-contrôleur ou d'un DSP. Le développement sur un FPGA nécessite plus de temps à cause de la programmation au niveau bit, mais des langages de synthèse haut niveau (High-level synthesis−HLS) peuvent être utilisés pour réduire ce temps.

Les processeurs programmables de type DSP ou micro-controleur peuvent fournir une puissance de calcul suffisante toute en ayant une basse consommation et un faible coût. Les performances d'un processeur sont définies par la fréquence d'horloge, la mémoire interne et la présence d'extensions comme, un multiplieur matériel, une unité *Multiply Accumulate* (MAC), une unité de calcul flottant (*Floating Point Unit*−FPU) ou une unité de traitement vectoriel (extension SIMD). Pour obtenir un bon compromis entre la puissance de calcul et la consommation énergétique, l'utilisation de plates-formes hétérogènes est recommandée. En effet, dans [Kleihorst et al., 2006] un processeur SIMD est associé au traitement bas niveau, un DSP ou un processeur générique aux traitements moyen et haut niveaux.

Des caméras intelligentes commercialisables comme razerCam [2], Pixy [3], les caméras intelligentes de matrix-vision [4] ou les caméras intelligentes à base de Raspberry Pi [5] sont aujourd'hui accessibles à tout le monde. D'autres caméras industrielles sont répertoriées dans [SmartCamera, 2008]. Le tableau 1.2 présente quelques caméras intelligentes commercialisées.

Le choix d'une caméra intelligente dépend donc de l'application, des contraintes sur la consommation et sur la vitesse de traitement.

2. http ://www.evt-web.com/en/products/smart-cameras/razercam/
3. http ://cmucam.org/projects/cmucam5
4. http ://www.matrix-vision.com/
5. https ://www.raspberrypi.org/

TABLE 1.1 – Caméras intelligentes de recherche

Référence	Imageur	Unité de traitement	Mémoire	Puissance
Cyclops [Rahimi et al., 2005]	ADCM1700, CIF	CPLD + AMTEL ATmega128L @7.37 MHz	64 ko SRAM + 512 ko Flash	<200 mW
WiCa [Kleihorst et al., 2006]	2 capteurs CMOS, VGA	Xetal IC3D@84 MHz (50 GOPS) + Amtel 8051	128 ko + mémoire image (2 images 256×256)	-
TRIcam [Arth et al., 2006]	pas de capteur (vidéo en entrée)	DSP TMS320C6414 @600 MHz + FPGA	128 Mbit de SRAM + 4 Mbit Flash	-
[Bramberger et al., 2006]	CMOS $LM-9618$, VGA@30 fps	plusieurs DSP TMS320C6415 @600 MHz (9600 MIPS)	784 Mo	35 W
[Dias et al., 2007]	4 Mpixels LUPA-4000, VGA@200 fps	ALTERA Startix EP1S60F1020C7	10 Mo de SRAM + 64 Mo de SDRAM	-
CMUcam3 [Rowe et al., 2007]	OV6620 CIF@50	ARM7TDMI @60 MHz	64 ko de RAM + 128 ko de Flash	500 mW
[Litzenberger et al., 2007]	capteur d'évènements asynchrone 128×128	DPS Blackfin BF537@600MHz	128 ko de mémoire interne et 32 Mo de SDRAM externe	2.5 W
CITRIC [Chen et al., 2008]	OV9655, VGA@30 fps, SXGA@15 fps	Intel xscale PXA jusqu'@624 MHz	64 Mo de RAM + 16 MB de NOR-Flash	<1 W
DSPcam [Kandhalu et al., 2009]	OV9653, VGA@30 fps, SXGA@15 fps	DPS Blackfin@600 MHz	32 Mo de SDRAM + 4 Mo de Flash	-
TRUSTcam [Winkler and Rinner, 2010]	Quickcam Pro9000, SVGA	ARM cortex-A8 @480 MHz + DPS TMS320C64xx @360 MHz	256 Mo de RAM + 256 Mo de NAND-Flash	-
TrustEYE.M4 [Winkler et al., 2014]	OV5642, VGA	ARM cortex-M4 + Raspberry Pi	192 ko de SRAM + 1 Mo Flash + 4 Mo de mémoire externe + mémoire de Raspberry Pi	-

1.5 Conclusion

Un état de l'art de suivi multi-objet pour un réseau de caméras a été présenté. Nous avons structuré le chapitre en 3 trois grands axes : suivi mono-caméra, suivi multi-caméra et architectures embarquées de vision susceptibles d'être déployées dans un réseau de caméras. Pour le suivi mono-caméra, les étapes principales ont été présentées. Les méthodes utilisées par chaque étape ont été classifiées. De même, les méthodes de suivi multi-caméra

TABLE 1.2 – Caméras intelligentes commercialisées

Référence	Imageur	Unité de traitement	Mémoire	Puissance	coût
Pixy (CMUcam5)	OV9715, 1280 × 800	NXP LPC4330 (dual core ARM cortex (M4+M0)) @204 MHz	264 ko de RAM + 1 Mo Flash	∼700 mW	69 $
À base de Raspberry PI 1	raspiCam, jusqu'2592 × 1944 @15 fps	BCM2835 (ARM1176JZF-S @700 Mhz)	256-512 Mo RAM	1.2-3.5 W [6]	∼60-80 $
À base de Raspberry PI 2	raspiCam, jusqu'2592 × 1944 @15 fps	BCM2836 (quadcore ARM Cortex-A7 @900 Mhz)	1 Go RAM	3 W	90 $
RazerCam	752 × 480 @60 fps ou 2056 × 1560 @30 fps	XILINX ZYNQ (FPGA + dual core ARM Cortex-A9)	512 Mo ou 1 Go DDR3 +4-16 Go Flash	4 W	>1000 $ [7]
mvBlueGEMINI (matrix-vision)	1280 × 1024 @60 fps	FPGA + dual core ARM Cortex-A9 @800 MHz	1 Go DDR3 + 4 Go Flash	<5 W	-
mvBlueLYNX (matrix-vision)	CCD jusqu'à2448 × 2050 jusqu'à @104 fps ou CMOS jusqu'à 2592 × 1944 jusqu'@117 fps	DSP jusqu'@800MHz + OMAP (ARM Cortex-A8 @1 GHz)	512 Mo DDR3 + Flash (sur carte micro SD)	<5 W	-

ont été présentées et groupées suivant la topologie du réseau. Dans le dernier axe, un récapitulatif des caméras intelligentes est fourni.

L'état de l'art montre que le suivi-multi objet peut s'effectuer en utilisant différents algorithmes et différentes architectures de calcul. Le choix des méthodes et des architectures est liée au scénario applicatif et aux performances attendues. Dans le prochain chapitre, nous discutons l'état de l'art présenté en regards de la problématique de la thèse définie. A l'issue de la discussion, nous présentons une vue globale de notre système de suivi multi-objet élaboré pour un réseau de caméras avec des champs de vision non recouvrants.

Chapitre 2

Vue globale du système de suivi

Sommaire

L'objectif de ce chapitre est de donner une vue globale de notre système de suivi d'objets. Dans la section 2.1, nous définissons le scénario de notre étude et les critères de choix des méthodes algorithmiques et des architectures de calcul. Dans les section 2.2, une analyse et une discussion de l'état de l'art présenté dans le chapitre 1 sont effectuées. Dans la section 2.3, la vue globale du système est présentée.

2.1 Scénario d'étude

L'objectif de la thèse est le suivi multi-objet dans un réseau de caméras avec des champs de vision non recouvrants. Le scénario applicatif visé peut être la surveillance sur une large zone, le bâtiment intelligent, ou l'aide aux personnes. Dans ces scénarios, les caméras sont fixes, la topologie du réseau est connue et les objets à suivre sont des personnes. Tous les objets en mouvement sont considérés comme des objets d'intérêt. Nous ne nous intéressons pas, dans cette thèse, aux scènes de foule où le nombre d'objets en mouvement est élevé et où les occultations sont difficilement solvables avec une seule caméra. Nous supposons que le nombre d'objets vus par une caméra reste moyen (moins d'une dizaine d'objets). L'objectif du système est d'assurer une bonne performance de suivi tout en atteignant le temps réel (>10 fps) sur des architectures de calcul de faibles complexités. Les architectures de faible complexité sont un choix conditionné par les contraintes de coût et de consommation énergétique liées au déploiement d'un réseau de caméras. Les critères de choix, selon les différentes parties, sont les suivants :

— **Suivi mono-caméra :** le choix des méthodes se base sur la maximisation du compromis entre la performance de suivi et la complexité de calcul et de mémoire connaissant la plate-forme de vision embarquée utilisée.

— **Suivi multi-caméra :** comme nous savons que le réseau contient des caméras avec des champs non recouvrants, nous visons un système qui n'a pas besoin d'une calibration précise afin d'assurer un simple déploiement, un traitement distribué pour garantir une faible bande passante de communication inter-caméra et pour faciliter le passage à l'échelle, et un bon compromis entre la performance de ré-identification et la complexité de calcul des algorithmes.
— **Architecture de vision :** le choix est basé sur le coût de la plate-forme, sa consommation énergétique, la puissance de calcul offerte et sa facilité de programmation.

2.2 Discussion et analyse de l'état de l'art

2.2.1 Suivi mono-caméra

2.2.1.1 Détection

La détection peut s'effectuer avec la soustraction de fond, le flux optique ou par une méthode d'apprentissage, comme expliqué dans la section 1.2.1. Le choix d'une méthode dépend du scénario applicatif, des architectures de calcul utilisées et des performances de suivi attendues.

La soustraction de fond et le flux optique considèrent tous les objets en mouvement comme des objets d'intérêt. La détection par apprentissage peut définir les objets d'intérêt comme une classe ou plusieurs classes d'objets. La soustraction de fond est destinée aux scénarios utilisant les caméras fixes. Les méthodes de flux optique et de détection par apprentissage peuvent être utilisées quelle que soit la nature des caméras (fixe ou mobile). La complexité des méthodes est variable suivant les algorithmes utilisés.

La plupart des algorithmes de soustraction de fond s'exécutent en temps réel sur un PC [Barnich and Van Droogenbroeck, 2011]. Le flux optique en temps réel n'est pas atteignable sur les processeurs génériques type PC, et dans la plupart des cas, des implémentations sur des architectures type *Graphics Processing Unit* (GPU) [Chase et al., 2008] sont utilisées. Dans [Zhang et al., 2014], une étude de l'algorithme du flux optique Lucas-Kanade est effectuée sur un DSP C66x de Texas Instrument consommant 10 W et contenant 8 coeurs fonctionnant à 1.25 GHz chacun. Les résultats obtenus montrent que pour atteindre 30 fps sur des images de résolutions 640×480 le nombre de DSP C66x nécessaires varie entre 2 et 8 selon les paramètres de l'algorithme. La nécessité de cette puissance de calcul démontre la complexité du flux optique. Les méthodes de détection par apprentissage sont coûteuses en calcul et le temps réel est difficilement atteignable même sur des PCs performants [Dollár et al., 2010]. Un détecteur de personnes à base de HOG + SVM a été implémenté sur la caméra intelligente CITRIC (tableau 1.1), dans [He et al., 2011]. Le temps nécessaire pour effectuer la détection sur une image de résolution 320×240 est de 37 secondes, ce qui est très loin du temps réel.

Les performances de détection dépendent de plusieurs paramètres. Le flux optique et la soustraction de fond sont plus sensibles aux variations d'éclairage et aux occultations que les méthodes d'apprentissage. La qualité de détection des méthodes d'apprentissage est liée à la base de données utilisée pour l'apprentissage du classificateur.

La soustraction de fond s'avère la méthode la plus pertinente pour notre contexte. D'un

côté, la complexité du calcul de la soustraction de fond est la plus faible parmi toutes les méthodes de détection, ce qui est prometteur pour un fonctionnement en temps réel sur des architectures embarquées. D'un autre côté, elle est adaptée au scénario applicatif qui se restreint aux caméras fixes.

2.2.1.2 Descripteurs

La description d'un objet dans un système de suivi peut se faire avec des caractéristiques de mouvement ou d'apparence. La complexité liée à l'extraction des caractéristiques dépend de leur nature.

La caractéristique du mouvement nécessite moins d'opérations, car seules la position de l'objet et sa taille sont nécessaires. La vitesse et l'accélération sont souvent estimées à partir de la position en utilisant un modèle de mouvement, fourni par l'algorithme de suivi. Le descripteur formé à partir de la caractéristique de mouvement est un vecteur d'état qui contient tous les paramètres décrivant l'objet (position, vitesse, taille, *etc*). Étant donné que le nombre d'éléments constituant le vecteur d'état est faible (moins de dix dans la plupart des cas), la mémoire nécessaire pour son stockage est faible et son évolution dans l'algorithme de suivi ne nécessite pas beaucoup d'opérations.

L'extraction de l'apparence d'un objet, quant à elle, demande plus de calcul, parce qu'elle nécessite une étape supplémentaire. L'apparence est extraite, seulement, une fois la position de l'objet connue. Suivant le modèle d'apparence utilisé, la complexité de l'extraction est variable. Par exemple, le nombre d'opérations n'est pas le même pour le calcul d'un histogramme de couleur, d'un HOG ou la représentation de l'objet en *template* de couleur. De plus, le nombre de paramètres définissant l'apparence d'un objet est plus important que le nombre d'éléments composant le vecteur d'état de la caractéristique de mouvement. Par exemple, décrire un objet avec un histogramme de couleur nécessite un vecteur de 3×256 éléments, quand aucune quantification n'est utilisée. Plus la taille de descripteur est élevée, plus la mémoire de stockage et le nombre d'opérations nécessaires pour son évolution sont importants.

La description d'un objet est sensible à la phase de détection. En effet, si un objet est mal détecté, il sera mal décrit. La fiabilité de la caractéristique de mouvement dépend de la fiabilité du modèle de mouvement de l'algorithme de suivi. Suivant la nature des objets, une caractéristique peut être plus pertinente qu'une autre. La caractéristique de mouvement est adéquate quand les objets ont la même apparence, par exemple, un groupe de personnes habillées de la même façon. L'apparence est pertinente quand les objets se déplacent d'une manière strictement aléatoire pour qu'ils soient décrits avec un modèle du mouvement fidèle, comme le déplacement des insectes, par exemple.

Pour la conception d'un algorithme de faible complexité pour les systèmes embarqués, la caractéristique de mouvement est la plus adéquate. L'apparence peut s'avérer intéressante pour la discrimination des objets quand des occultations se produisent, en particulier quand les objets d'intérêt sont des personnes.

2.2.1.3 Suivi

Les algorithmes de suivi sont classifiés en trois catégories : suiveurs de points, suiveurs de formes paramétriques et suiveurs de formes non paramétriques, comme expliqué dans la

section 1.2.3. Les suiveurs de points utilisent un vecteur d'état comme entrée. Le vecteur d'état contient souvent des caractéristiques de mouvement pour les systèmes linéaires Gaussiens (les implémentations à base de filtre de Kalman) et mouvement et/ou apparence dans le cas général (les implémentations à base de filtre à particules). En effet, un modèle d'évolution du vecteur d'état est utilisé pour les suiveurs de points. La complexité du suivi est dépendante des caractéristiques utilisées pour la description de l'objet, du modèle d'évolution et du type d'implémentation. Pour les modèles d'évolution simples qui utilisent la caractéristique du mouvement, les suiveurs de points sont moins complexes que les suiveurs de formes. Les suiveurs de formes sont basés sur des caractéristiques d'apparence extraites au niveau pixel. Les vecteurs de caractéristiques d'apparence sont souvent de grandes tailles. L'utilisation des formes non paramétriques nécessite des caractéristiques extraites au niveau pixel type contour ou silhouette. Comme les caractéristiques sont non paramétriques, le nombre d'éléments les composant est élevé et dépend souvent de la taille des objets. En règle générale, plus les caractéristiques utilisées pour le suivi sont de petites tailles, plus le suivi est de faible complexité.

L'utilisation de la caractéristique de mouvement pour le suivi d'objets impose l'utilisation de suiveurs de points. La faible complexité de ce suivi est prometteuse pour un fonctionnement en temps réel sur des architectures embarquées.

2.2.2 Suivi multi-caméra

Le suivi d'objets dans un réseau de caméras dépend de la topologie du réseau : caméra avec des champs recouvrants ou caméras avec des champs non recouvrants. L'utilisation de caméras avec des champs recouvrants nécessite une calibration et une synchronisation précises des caméras. La fusion centralisée fournit de meilleures performances mais nécessite beaucoup de calcul et de bande passante de communication. La fusion distribuée assure la flexibilité du système et la facilité de passage à l'échelle.

Dans la thèse, nous nous intéressons aux réseaux de caméras avec des champs non recouvrants. Les réseaux calibrés nécessitent, selon les caractéristiques de ré-identification utilisées, un apprentissage de la topologie du réseau, des fonctions de transfert de couleur entre caméras et des connaissances a priori sur le mouvement des objets. Les réseaux non calibrés nécessitent l'utilisation de caractéristiques invariantes aux changements de points de vue et d'éclairage. La connaissance de la topologie permet de limiter le nombre de caméras où l'objet est recherché, et donc de réduire la complexité de calcul et la bande passante de communication inter-caméra. La complexité des réseaux de caméras non calibrées est plus importante car les caractéristiques utilisées pour la ré-identification sont plus complexes.

Afin d'assurer un fonctionnement sur des réseaux de caméras intelligentes, la bande passante de communication et la complexité de calcul sont des contraintes fortes, car elles agissent directement sur la consommation du système. L'utilisation d'un traitement distribué sur un réseau calibré est la solution la plus prometteuse pour notre contexte.

2.2.3 Architectures de vision

Les architectures embarquées de vision présentent différentes caractéristiques, comme expliqué dans la section 1.4. Les caméras intelligentes à base de FPGA sont efficaces pour les traitements bas niveau réguliers qui ne nécessitent pas d'opérations en virgule flottante. Cependant, elles sont difficiles à programmer. Les caméras intelligentes à base de processeurs programmables offrent une facilité de programmation et une large gamme de performance. Celles utilisant des DSP sont adéquates pour les traitements utilisant des opérations MAC qui sont souvent des traitements bas et moyen niveaux comme la convolution, la segmentation ou le calcul de distances. Les processeurs SIMD comme unité de traitement offrent un bon compromis entre la consommation et la puissance de calcul quand plusieurs données nécessitent le même traitement, ce qui est le cas des traitements bas niveau s'effectuant sur les pixels de l'image. Les processeurs génériques type micro-contrôleur sont utilisés pour différents types de traitements et sont plus adaptés pour les traitements irréguliers comme l'exécution d'un système d'exploitation où pour la prise de décision.

La puissance de calcul, la consommation énergétique et le coût des caméras intelligentes qui sont sur le marché (tableau 1.2) sont différents d'une caméra à une autre. En effet, certaines caméras intelligentes comme Pixy consomment quelques mW et coûtent quelques dizaines d'euros. Cependant, elles offrent une faible puissance de calcul (faible fréquence d'horloge, FPU peu performante, faible capacité mémoire, *etc*). Étant données ces performances, peu d'algorithmes de vision peuvent fonctionner sur ce type de plates-formes.

Les caméras intelligentes à base de Raspberry Pi peuvent être considérées comme de moyenne gamme. En effet, leur consommation varie entre 1 et 3,5 W, suivant la version du Raspberry Pi utilisée et le coût est de quelques dizaines d'euros. Le processeur présent dans la carte de traitement est suffisamment performant (700 MHz pour la version 1 et 1 GHz en quadcore pour la version 2, entre 256 et 512 Mo de RAM) pour faire fonctionner certains algorithmes de traitement d'image.

La dernière gamme des caméras intelligentes est la gamme haute performance comme RazerCam qui contient un FPGA, un processeur embarqué performant (Dual core ARM Cortex-A9 @ 1 GHz) et une grande capacité mémoire. Ces plates-formes offrent une grande puissance de calcul avec une consommation modérée (4 W pour la RazerCam). Le seul inconvénient est le coût de ces caméras intelligentes, qui est de l'ordre de centaines voire de milliers d'euros.

Les caractéristiques des caméras intelligentes de la gamme moyenne sont adaptées pour notre contexte. En effet, pour une consommation modérée et un faible coût, nous avons une certaine flexibilité sur le choix des algorithmes vue la puissance de calcul qu'elles permettent. Les caméras intelligentes permettant une basse consommation et un faible coût ont une faible puissance de calcul et une taille mémoire faible pour exécuter les algorithmes de vision. Les caméras intelligentes de haute performance sont coûteuses.

2.3 Vue globale du système

Comme résultat à la discussion effectuée dans la section 2.2, nous fournissons la vue globale de notre système de suivi multi-objet pour un réseau de caméras avec des champs de vision non recouvrants. Dans notre système, chaque caméra contient une chaîne de suivi multi-objet. Quand un objet sort du champ de vision d'une caméra, cette caméra communique les informations de l'objet aux caméras voisines où l'objet est susceptible d'apparaître. Les caméras voisines attendent donc l'apparition de l'objet aux endroits correspondants. Une fois qu'une caméra ré-identifie l'objet, elle transmet un message aux autres caméras pour qu'elles arrêtent d'attendre.

Suivi multi-objet mono-caméra

Nous utilisons la soustraction de fond pour la détection d'objets. Les objets sont décrits par des vecteurs contenant les caractéristiques de mouvement. Le suivi s'effectue avec un algorithme suiveur de points.

Réseau de caméras avec des champs non recouvrants

Nous utilisons un traitement distribué tenant compte de la connaissance a priori fournie par le réseau de caméras pour ré-identifier les objets transitant entre les caméras. La ré-identification s'effectue avec des caractéristiques simples afin d'assurer une faible bande passante.

Architectures embarquées de vision

Nous utilisons une caméra intelligente de gamme moyenne. Les caméras intelligentes décrites dans le tableau 1.1 sont difficiles à se procurer vu qu'elles ne sont pas commercialisées. Parmi les caméras intelligentes commercialisées, la caméra moyenne gamme à base de la carte Raspberry Pi s'avère la plus pertinente pour un déploiement en réseau vue son prix, sa consommation et la puissance de calcul qu'elle offre. Nous utilisons donc une caméra intelligente composée de la carte Raspberry Pi version 1 comme unité de traitement et du module RaspiCam comme imageur.

Chapitre 3

Détection d'objets

Sommaire

La détection d'objets en mouvement avec la soustraction de fond s'avère être la méthode la plus adéquate pour notre contexte où les caméras sont fixes. L'objectif est de développer une méthode de détection robuste à la présence d'ombre et à l'effet fantôme (*"ghost"*) sans utilisation d'un post-traitement, tout en ayant une faible complexité de calcul et de faibles besoins en mémoire. Un objet fantôme est une fausse détection causée par le mouvement d'un objet appartenant à l'arrière-plan. Dans ce chapitre, nous présentons une méthode de détection d'objets basée sur un algorithme de soustraction de fond utilisant l'information d'intensité et l'information du gradient. La méthode est appliquée pour la détection de personnes dans des environnements d'intérieur et d'extérieur.

La première section du chapitre analyse les algorithmes de soustraction de fond existants. Dans la seconde section, nous présentons notre méthode. La complexité de calcul, les besoins en mémoire et les résultats obtenus en appliquant la méthode développée sur deux séquences vidéos des bases de données PETS2009 [1] et CAVIAR [2] sont fournis dans la troisième section.

3.1 Analyse des algorithmes de soustraction de fond

La complexité de calcul, la capacité mémoire ainsi que la qualité de segmentation dépendent de l'algorithme de soustraction de fond utilisé. Plusieurs comparaisons des algorithmes d'extraction de fond ont été effectuées dans la littérature. Nous nous basons

1. http ://www.cvg.reading.ac.uk/PETS2009/a.html
2. http ://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

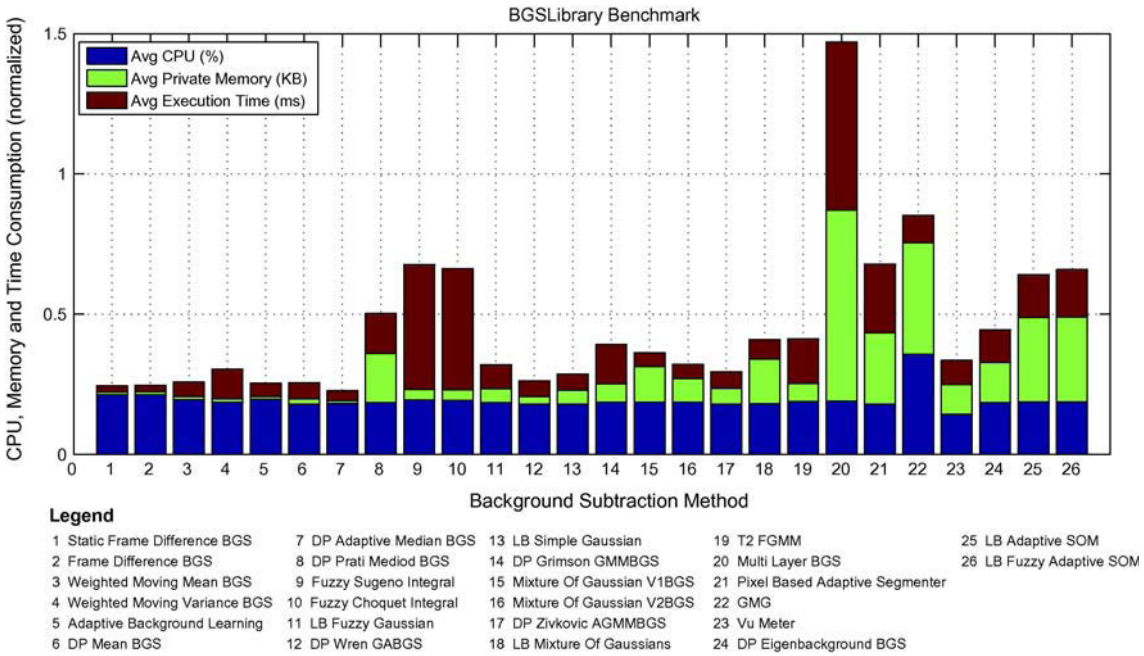

FIGURE 3.1 – Le temps d'exécution, l'occupation de processeur et la capacité mémoire normalisés des algorithmes de soustraction de fond [Sobral and Vacavant, 2014]. La plateforme de traitement est un PC avec un Core i5-2410M fonctionnant à 2.3 GHz et 4 Go de RAM.

sur les résultats de [Sobral and Vacavant, 2014] qui sont donnés par le tableau 3.1 pour la qualité de segmentation et par la figure 3.1 pour la complexité des algorithmes. La comparaison de la qualité de segmentation est basée sur la métrique FSD. FSD est calculée à partir de trois métriques comme indiqué dans l'équation (3.1) : Fmesure (équation (3.7)), Dscore et Structural SIMilarity (SSIM). Fmeasure est la moyenne harmonique de la précision *"Precision"* et du rappel *"Recall"* du résultat. Les métriques Précision et Rappel sont calculées à partir des pixels correctement et faussement classifiés comme appartenant à un objet (vrais positifs TP_{pix}, faux positifs FP_{pix} et faux négatifs FN_{pix}). La métrique SSIM évalue la qualité visuelle et perceptuelle du résultat de la soustraction de fond. La métrique Dscore considère la distance de l'erreur de segmentation par rapport à un objet réel. Les métriques sont détaillées dans [Sobral and Vacavant, 2014].

$$FSD = \frac{\overline{Fmesure} + \overline{SSIM} + (1 - \overline{Dscore})}{3} \tag{3.1}$$

Avec $\overline{x}$ la moyenne normalisée de la métrique x pour un algorithme de soustraction de fond donné.

La complexité d'un algorithme est traduite par le besoin en mémoire, le temps d'exécution et l'occupation de processeur. Afin de pouvoir comparer les algorithmes, les auteurs ont normalisé ces trois paramètres.

Les algorithmes présentant les meilleures performances de segmentation dans le tableau 3.1 (FSD en gras) ont une complexité de calcul et des besoins en mémoire

TABLE 3.1 – Comparaison de la qualité de segmentation des algorithmes de soustraction de fond [Sobral and Vacavant, 2014]. FSD est une mesure de la qualité de segmentation calculée à partir de trois métriques : Fmesure, Dscore et Structural SIMilarity (SSIM). Une grande FSD implique une meilleure qualité de segmentation. Les séquences vidéos utilisées sont celles de la base de données Background Models Challenge (BMC).

Référence	Algorithme	ID	FSD
-	Static Frame Difference	StaticFrameDifferenceBGS	0.119
-	Frame Difference	FrameDifferenceBGS	0.799
-	Weighted Moving Mean	WeightedMovingMeanBGS	0.818
-	Weighted Moving Variance	WeightedMovingVarianceBGS	0.814
-	Adaptive Background Learning	AdaptiveBackgroundLearning	0.896
-	Temporal Mean	DPMeanBGS	0.642
[McFarlane and Schofield, 1995]	Adaptive Median	DPAdaptiveMedianBGS	0.691
[Calderara et al., 2006]	Temporal Median	DPPratiMediodBGS	0.888
[Zhang and Xu, 2006]	Fuzzy Sugeno Integral	FuzzySugenoIntegral	0.874
[El Baf et al., 2008a]	Fuzzy Choquet Integral	FuzzyChoquetIntegral	0.884
[Sigari et al., 2008]	Fuzzy Gaussian	LBFuzzyGaussian	0.738
[Wren et al., 1997]	Gaussian Average	DPWrenGABGS	**0.922**
[Benezeth et al., 2008]	Simple Gaussian	LBSimpleGaussian	0.767
[Stauffer and Grimson, 1999]	Gaussian Mixture Model	DPGrimsonGMMBGS	0.808
[KaewTraKulPong and Bowden, 2002]	Gaussian Mixture Model	MixtureOfGaussianV1BGS	0.910
[Zivkovic, 2004, Zivkovic and van der Heijden, 2006]	Gaussian Mixture Model	MixtureOfGaussianV2BGS	0.899
[Zivkovic, 2004, Zivkovic and van der Heijden, 2006]	Gaussian Mixture Model	DPZivkovicAGMMBGS	0.746
[Bouwmans et al., 2008]	Gaussian Mixture Model	LBMixtureOfGaussians	0.907
[El Baf et al., 2008b, Bouwmans and El Baf, 2009, El Baf et al., 2009]	Type-2 Fuzzy GMM-UM	T2FGMM_UM	0.745
[El Baf et al., 2008b, Bouwmans and El Baf, 2009, El Baf et al., 2009]	Type-2 Fuzzy GMM-UV	T2FGMM_UV	0.628
[Zhao et al., 2012]	Type-2 Fuzzy GMM-UM with MRF	T2FMRF_UM	0.030
[Zhao et al., 2012]	Type-2 Fuzzy GMM-UV with MRF	T2FMRF_UV	0.716
[Yao and Odobez, 2007]	Multi-Layer BGS	MultiLayerBGS	**0.974**
[Hofmann et al., 2012]	Pixel-Based Adaptive Segmenter	PixelBasedAdaptiveSegmenter	**0.985**
[Godbehere et al., 2012]	GMG	GMG	0.730
[Goyat et al., 2006]	VuMeter	VuMeter	0.735
[Oliver et al., 2000]	Eigenbackground/SL-PCA	DPEigenbackgroundBGS	0.114
[Maddalena and Petrosino, 2008]	Adaptive SOM	LBAdaptiveSOM	**0.952**
[Maddalena and Petrosino, 2010]	Fuzzy Adaptive SOM	LBFuzzyAdaptiveSOM	0.848

élevés, selon la figure 3.1. L'algorithme assurant le bon compromis entre la complexité et la qualité de segmentation est DPWrenGABGS qui utilise une distribution Gaussienne par pixel. La normalisation des paramètres de complexité ne nous permet pas de connaître la vitesse d'exécution d'un algorithme et ses besoins en mémoire. Afin d'avoir une information sur ces paramètres, nous exploitons les résultats de [Barnich and Van Droogenbroeck, 2011] présentés dans les figures 3.2 et 3.3. Les algorithmes considérés sont ViBe [Barnich and Van Droogenbroeck, 2011], First-Order low pass filter qui correspond à l'algorithme AdaptativeBackgroundLearning de tableau 3.1 et dont la mise à jour de l'arrière plan s'effectue avec l'équation (1.1), Gaussian Model qui utilise une mise à jour du modèle de la variance et qui est proche des algorithmes DPWrenGABGS et LBSimpleGaussian de tableau 3.1, GMM correspondant à l'algorithme MixtureOfGaussianV1BGS de tableau 3.1, EGMM qui correspond à l'algorithme DPZivkovicAGMMBGS de tableau 3.1, Codebook [Kim et al., 2005], Bayesian Histogram [Li et al., 2003] et Zipfian Sigma-Delta [Manzanera, 2007]. La métrique utilisée pour quantifier la qualité de segmentation est le pourcentage de pixels correctement classifiés (PCC) calculée avec l'équation (3.2). La complexité de l'algorithme est traduite par le nombre d'images traitées par seconde.

$$PCC = \frac{TP_{pix} + TN_{pix}}{TP_{pix} + TN_{pix} + FP_{pix} + FN_{pix}} \tag{3.2}$$

Avec TP_{pix} le nombre de pixels correctement classifiés comme appartenant à un objet, TN_{pix} le nombre de pixels correctement classifiés comme appartenant à l'arrière plan, FP_{pix} le nombre de pixels faussement classifiés comme appartenant à un objet et FN_{pix} le nombre de pixels faussement classifiés comme appartenant à l'arrière plan.

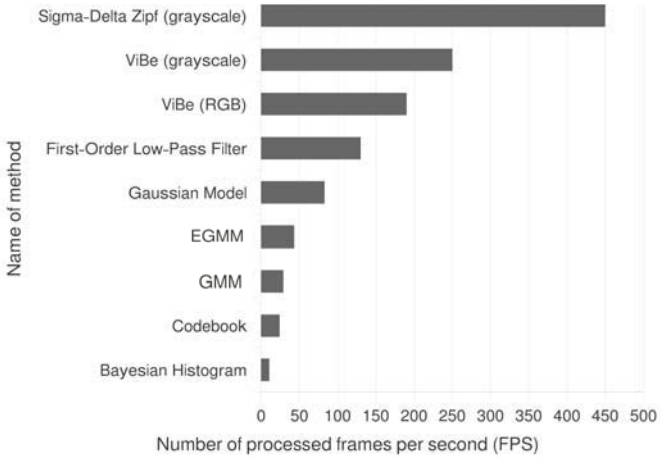

FIGURE 3.2 – Nombre d'images traitées par seconde pour les algorithmes de soustraction de fond [Barnich and Van Droogenbroeck, 2011]. La plate-forme de traitement : Core i7 @2.67 GHz, 6 Go de RAM. Implémentation en langage C. Résolution des images : 640×480.

Les figures 3.2 et 3.3 montrent que Zipfian Sigma-Delta est l'algorithme ayant un faible temps d'exécution, ce qui lui permet de traiter 450 images par seconde pour des images de résolution 640×480. Avec cette vitesse d'exécution, Zipfian Sigma-Delta surpasse tous

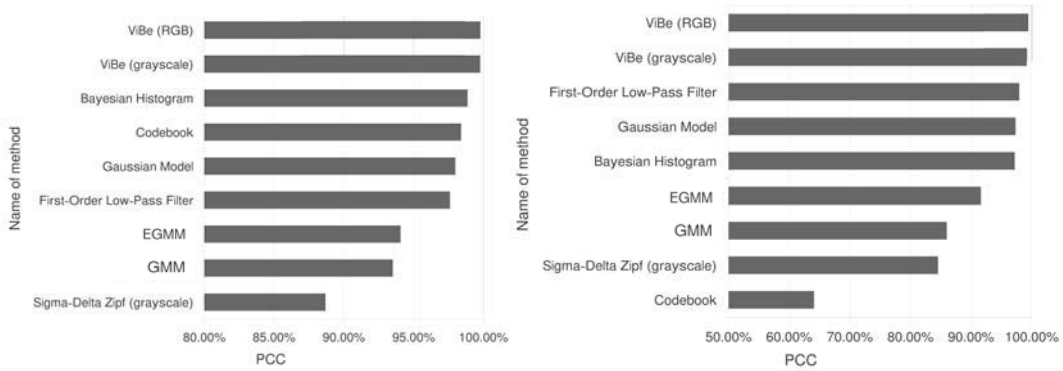

FIGURE 3.3 – Comparaison de la qualité de segmentation de quelques algorithmes de soustraction de fond [Barnich and Van Droogenbroeck, 2011]. Les séquences vidéos utilisées : a) : House, b) : PETS2001.

les autres algorithmes, même le First-Order low pass filter qui est considéré comme un algorithme basique. Les algorithmes EGMM, GMM, Codebook et Bayesian Histogram sont lourds en calcul. La vitesse de traitement est inférieure à 50 fps sur la machine Core i7 2.67 GHz. Par ailleurs, un traitement en temps réel est difficilement atteignable sur un processeur embarqué avec ces algorithmes. L'algorithme Gaussian Model s'exécute à une vitesse d'environ 80 fps. ViBe offre la meilleure qualité de segmentation pour les deux séquences vidéos, contrairement au Zipfian Sigma-Delta. La PCC du Gaussian model est plus élevée que celle de Zipfian Sigma-Delta. Les résultats de segmentation de Zipfian Sigma-Delta, Gaussian Model et ViBe sont présentés dans la figure 3.4. Ces résultats montrent que la segmentation avec Zipfian Sigma-Delta présente beaucoup de bruit (pixels blancs non connectés), ce qui réduit la valeur de PCC. Comme nous nous intéressons à la détection d'objets, ce type de bruit peut être filtré avec un filtre morphologique ou par l'analyse des composantes connectées en considérant seulement les régions connectées ayant une surface supérieure à un seuil. La qualité visuelle de la sortie obtenue par Gaussian Model est moins bonne que celle de Zipfian Sigma-Delta pour l'isolation de l'objet d'intérêt avec l'analyse des composantes connectées. Ainsi, le meilleur compromis atteint par l'algorithme DP-WrenGABGS (à base de Gaussian Model) dans l'étude de [Sobral and Vacavant, 2014] n'est pas forcement pertinent pour la détection d'objets sur ces séquences vidéos. Autrement dit, la qualité de segmentation au niveau pixel peut ne pas avoir une signification au niveau région.

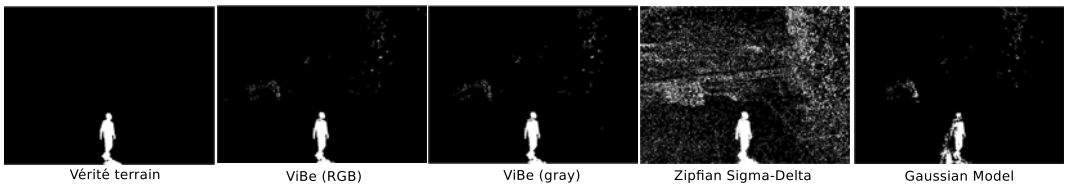

FIGURE 3.4 – Résultat de segmentation avec ViBe, Zipfian Sigma-Delta et Gaussian Model [Barnich and Van Droogenbroeck, 2011].

En se basant sur les figures 3.2 et 3.3, ViBe est l'algorithme présentant le meilleur compromis entre le temps d'exécution et la qualité de segmentation. Cependant, les auteurs n'ont pas étudié les besoins en mémoire des algorithmes. L'algorithme ViBe a besoin de stocker plusieurs échantillons par pixel : 20 échantillons par pixel dans l'implémentation [Barnich and Van Droogenbroeck, 2011]. Chaque échantillon est une valeur d'intensité (8 bits) ou de couleur (24 bits). Ainsi, le modèle d'arrière plan est représenté par plusieurs images en mémoire (20 images dans ce cas). Zipfian Sigma-Delta, quant à lui, a besoin de stocker seulement une image d'arrière plan (8 bits par pixel) et une image de variance dont le nombre de bits par pixel est choisi par l'utilisateur.

La méthode First-Order low pass filter nécessite 4 octets par pixel pour la modélisation de l'arrière plan. Ces besoins en mémoire sont dûs au codage en flottant causé par la manière dont la mise à jour de l'arrière plan s'effectue. En effet, un codage en entier peut être considéré. Dans ce cas, 8 bits sont nécessaires pour la partie entière et N bits pour la partie fractionnaire. La partie fractionnaire est introduite par l'utilisation du paramètre α qui contrôle la vitesse de mise à jour de l'arrière plan, comme indiqué par l'équation (1.1). Une petite valeur de α implique une grande valeur de N, pour ne pas perdre en précision. Nous retrouvons ce désavantage dans les algorithmes utilisant les modèles Gaussiens, car la mise à jour du modèle est basée sur la même équation (1.1).

La liste des algorithmes présentés dans le tableau 3.1 et la figure 3.2 n'est pas exhaustive. En effet, dans [Casares et al., 2010, Casares, 2014], un autre algorithme de soustraction du fond appelé *Adaptive Light-Weight* (ALW) est présenté. Les auteurs ont comparé les besoins en mémoire de leur algorithme ALW avec les algorithmes de soustraction du fond GMM [Stauffer and Grimson, 1999], DPEigenbackgroundBGS [Oliver et al., 2000] et Codebook [Kim et al., 2005]. Le résultat de cette comparaison est fourni par le tableau 3.2.

TABLE 3.2 – Besoins en mémoire des méthodes de soustraction de fond quand des images en gris sont utilisées.

Méthode	GMM	DPEigenback-groundBGS	Codebook	ALW	First-Order low pass filter (équation 1.1)	ViBe	Zipfian Sigma-Delta
Mémoire par pixel (octet)	32	28	91	6.25	4	20	2

D'après le tableau 3.2, Zipfian Sigma-Delta est l'algorithme nécessitant le peu de mémoire, ce qui le rend un bon candidat pour notre contexte. Une autre méthode, qui n'est pas considérée est celle développée dans [Javed et al., 2002] qui fusionne l'information de la couleur avec l'information du gradient. En effet, deux soustractions de fond indépendantes sont effectuées en utilisant des versions modifiées de l'algorithme DPGrimsonGMMBGS. La première soustraction de fond s'effectue sur l'information couleur de l'image. Dans la deuxième soustraction de fond, le gradient (amplitude et orientation) est utilisé. Les deux résultats obtenus sont alors fusionnés au niveau région. Si l'inégalité de l'équation (3.3) est vérifiée, la région segmentée avec l'information couleur est considérée comme objet. La

complexité liée à cette méthode est supérieure à deux fois la complexité de l'algorithme DPGrimsonGMMBGS qui est proche de la complexité de GMM.

$$\frac{\sum_{(i,j)\in\partial R_a}(\nabla I(i,j)G(i,j))}{|\partial R_a|} \geqslant P_b \tag{3.3}$$

∂R_a : est le contour de la région R_a déterminée par la soustraction de fond utilisant l'information de la couleur, $\nabla I(i,j)$: est le contour de l'image d'entrée I et $G(i,j)$: le résultat de la soustraction de fond à base de l'information du gradient.

Après cette analyse, Zipfian Sigma-Delta est donc l'algorithme présentant le moins de complexité en calcul et en mémoire. En prenant en considération le fait que la qualité de segmentation d'objets avec Zipfian Sigma-Delta peut être améliorée par le reste de la chaîne de détection (filtrage, analyse des composantes connectées), Zipfian Sigma-Delta est un bon choix pour un système embarqué de détection d'objets.

Dans la prochaine section, nous présentons notre méthode de détection d'objets basée sur le l'algorithme Zipfian Sigma-Delta amélioré par l'information du gradient de l'image d'entrée.

3.2 Méthode proposée

La méthode de détection d'objets proposée se compose de deux modules. Le premier est basé sur la soustraction de fond. Le deuxième effectue l'analyse des composantes connectées de l'image binaire résultante. Aucun filtrage morphologique n'est utilisé à cause de la complexité de calcul qu'il engendre sur un processeur embarqué.

3.2.1 Module de soustraction de fond

La soustraction de fond s'effectue en fusionnant hiérarchiquement la sortie de l'algorithme Zipfian Sigma-Delta [Manzanera, 2007] appliqué sur l'intensité de l'image avec l'amplitude du gradient de l'image d'entrée. Cette méthode est inspirée de la méthode de [Javed et al., 2002] où l'information couleur et l'information du gradient sont utilisées pour la soustraction de fond. Notre méthode est différente de la méthode de [Javed et al., 2002] par le fait qu'aucun modèle d'arrière plan à base d'un mélange de Gaussiennes n'est utilisé, aucune soustraction de fond à base de gradient n'est effectuée, seule l'amplitude du gradient de l'image d'entrée est utilisée. L'autre différence est dans la manière dont l'information est fusionnée pour la prise de décision. Dans la méthode proposée, l'amplitude du gradient d'un pixel est calculée uniquement quand le pixel est classifié comme appartenant à un objet avec l'algorithme Zipfian Sigma-Delta, ce qui assure une faible complexité de l'algorithme. La sortie de notre méthode est la boîte englobant le contour d'un objet présent dans la scène. Les caractéristiques de l'objet sont extraites à partir des pixels classifiés par Zipfian Sigma-Delta comme appartenant à l'objet et qui sont à l'intérieur de la boîte résultante. Pour cette raison, l'influence du bruit de segmentation engendré par Zipfian Sigma-Delta est réduite. Grâce à l'amplitude du

gradient, la détection devient robuste à la présence d'ombre et à la présence de *"ghost"* causée par l'initialisation du modèle d'arrière plan avec une image contenant des objets.

Zipfian Sigma-Delta est choisi pour sa faible complexité calculatoire et ses faibles besoins en mémoire. Dans l'algorithme Zipfian Sigma-Delta, l'arrière plan est modélisé par une image en niveau de gris M et une image de variance V. Les deux images sont mises à jour avec la modulation $\Sigma - \Delta$ utilisée dans la conversion analogique-numérique [Manzanera, 2007]. La mise à jour du pixel $M(x, y)$ de l'arrière plan est conditionnée par la valeur de la variance $V(x, y)$. La mise à jour de la variance s'effectue périodiquement après chaque Tv images. Les différentes étapes de Zipfian Sigma-Delta sont décrites dans l'algorithme 1.

Pour chaque instant t, la première étape consiste à mettre à jour M si la variance est supérieure à 2^{m-p}, avec 2^p est la plus grande puissance de 2 divisant t, et m est le nombre de bits pour coder la variance (la dynamique). En effet, la mise à jour de M s'effectue chaque image si $V > 2^{m-1}$, chaque deux images si $2^{m-2} \leqslant V < 2^{m-1}$, *etc.* La deuxième étape consiste à calculer la différence absolue D entre l'image courante I et l'arrière plan M. La troisième étape a pour but de mettre à jour la variance V si l'instant t est un multiple de Tv, avec Tv est un paramètre fixe dans l'algorithme. Afin que les valeurs de la variance ne dépassent pas la dynamique allouée, les bornes V_{min} et V_{max} sont utilisées. La quatrième étape consiste à classifier le pixel x comme appartenant à un objet si sa différence absolue $D(x)$ est supérieure à sa variance $V(x)$. Dans le cas contraire, le pixel x est considéré comme appartenant à l'arrière plan.

Une fois un pixel x de l'image est classifié comme appartenant à un objet avec l'algorithme Zipfian Sigma-Delta, l'amplitude du gradient $GMag_t(x)$ du pixel est calculée avec l'équation (3.4). Si l'amplitude du gradient est supérieure à un seuil prédéfini $th1$, elle est multipliée par la valeur de la différence absolue $D_t(x)$ calculée dans Zipfian Sigma-Delta, comme indiqué dans le diagramme de la figure 3.5. La multiplication a pour rôle d'amplifier la différence entre les pixels appartenant aux contours des objets avec le reste des pixels. Finalement, le résultat de multiplication est comparé à un seuil $th2$ afin de décider si le pixel appartient au contour d'un objet ou à l'arrière plan. La sortie de l'algorithme est une image binaire FGC_t contenant les contours des objets.

$$GMag_t(x) = |I_t(i+1, j) - I_t(i-1, j)| + |I_t(i, j+1) - I_t(i, j-1)| \tag{3.4}$$

i et j sont les coordonnées du pixel x.

3.2.2 Module d'analyse des composantes connectées

Le deuxième module effectue une analyse des composantes connectées sur l'image FGC_t, afin d'extraire les boîtes englobant les contours des objets. Dans notre implémentation, nous utilisons l'algorithme *Optimised Single Pass* [Ma et al., 2008] qui permet de labelliser les composantes connectées à la volée. Ainsi, l'image d'entrée n'a pas besoin d'être stockée, ce qui engendre une faible empreinte mémoire. Les auteurs montrent que l'algorithme fonctionne à plus de 100 fps pour une implémentation sur un FPGA Virtex-II XC2V6000 avec des images de résolution 640×480. Les ressources utilisées par l'algorithme

entrée: I_t : image à l'instant t, M_{t-1} : image de l'arrière plan à l'instant $t-1$, V_{t-1} : image de la variance à l'instant $t-1$
sortie : fg_t : image binaire résultante de la soustraction de fond à l'instant t, M_t : image de l'arrière plan à l'instant t, V_t : image de la variance à l'instant t

si *initialisation* $(t == 0)$ **alors**
  pour chaque *pixel* x **faire**
    $M_0(x) = I_0(x)$;
    $V_0(x) = V_{min}$;
  fin
sinon
  $rank = t\%2^m$; $pow2 = 1$;
  $pow2 = 2 \times pow2$;
  répéter
    $pow2 = 2 \times pow2$
  jusqu'à $(rank\%pow2 == 0)$ **et** $(pow2 < 2^m)$;
  pour chaque *pixel* x **faire**
    # étape 1 : mise à jour de l'arrière plan
    si $V_{t-1}(x) > \frac{2^m}{pow2}$ **alors**
      si $M_{t-1}(x) < I_t(x)$ **alors** $M_t(x) = M_{t-1}(x) + 1$;
      si $M_{t-1}(x) > I_t(x)$ **alors** $M_t(x) = M_{t-1}(x) - 1$;
      autres cas $M_t(x) = M_{t-1}(x)$;
    sinon
      $M_t(x) = M_{t-1}(x)$
    fin
    # étape 2 : calcul de la différence absolue
    $D_t(x) = |I_t(x) - M_t(x)|$;
    # étape 3 : mise à jour de la variance
    si $t\%Tv == 0$ **alors**
      si $V_{t-1}(x) < N \times D_t(x)$ **alors** $V_t(x) = V_{t-1}(x) + 1$;
      si $V_{t-1}(x) > N \times D_t(x)$ **alors** $V_t(x) = V_{t-1}(x) - 1$;
      autres cas $V_t(x) = V_{t-1}(x)$;
    fin
    $V_t(x) = max(min(V_t(x), V_{max}), V_{min})$;
    # étape 4 : classification du pixel
    si $D_t(x) < V_t(x)$ **alors** $fg_t(x) = 0$ **sinon** $fg_t(x) = 1$;
  fin
fin

Algorithm 1: Zipfian Sigma-Delta

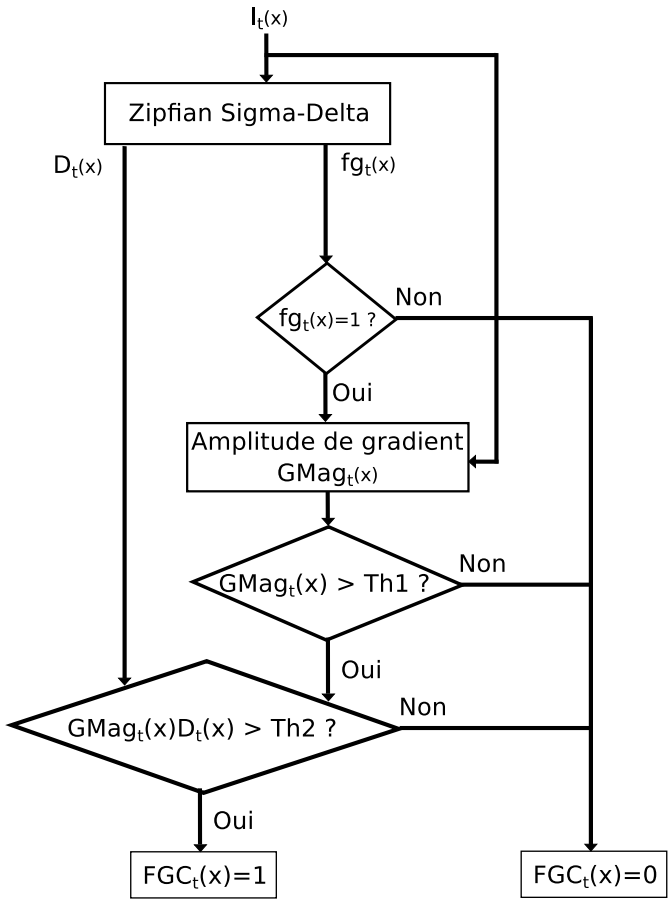

FIGURE 3.5 – Organigramme de l'algorithme de soustraction de fond proposé.

se résument à 4 *blocks RAM* (2.8%), 600 *slice flip flops* (0.9%) et 1757 *LUTs* (2.6%) pour une fréquence d'horloge maximale de 40,63 MHz. Le faible pourcentage d'éléments logiques utilisés sur le FPGA démontre la faible complexité de l'algorithme.

Afin de classifier une composante connectée comme objet d'intérêt, les caractéristiques ratio, nombre de pixels de contour et la taille de la boîte englobant ce contour sont utilisées. En effet, seuls les objets dont les boîtes ayant un ratio, une taille et un nombre de pixels supérieurs à des seuils prédéfinis sont considérés comme objets d'intérêt. De plus, les boîtes proches les unes des autres sont fusionnées pour former une seule boîte (des seuils de fusion sont définis dans l'algorithme). La distance utilisée pour détecter si deux boîtes sont proches est la distance euclidienne entre les deux points les plus proches des boîtes. Si deux boîtes s'intersectent, la distance entre elles est la distance minimale qui est égale à 0. Cette distance est détaillée dans le chapitre 5. Cette fusion permet de rectifier les erreurs de segmentation engendrant une discontinuité dans la région de l'objet.

Pour extraire des caractéristiques décrivant l'objet (comme les histogrammes), le résultat de segmentation obtenu par Zipfian Sigma-Delta à l'intérieur de la boîte englobant l'objet est utilisé. Ainsi, les pixels de l'arrière plan et les pixels de l'objet à l'extérieur de la boîte englobante ne sont pas considérés. L'évaluation de la méthode proposée ainsi que des exemples d'images sont présentés dans la prochaine section.

3.3 Résultats et discussion

La méthode de détection proposée est évaluée selon la qualité de détection, la complexité de calcul et les besoins en mémoire.

3.3.1 La qualité de détection

La qualité de la méthode proposée est évaluée sur les deux bases de données PETS2009 et CAVIAR qui sont prises avec des capteurs de différente qualité et dans des environnements différents. Les métriques utilisées pour l'évaluation sont : Précision (équation (3.5)), Rappel (équation (3.6)), FMesure (équation (3.7)), NMODA (Normalized Multiple Object Detection Accuracy) (équation (3.9)) et NMODP (Normalized Multiple Object Detection Precision) (équation (3.11)).

$$Precision = \frac{TP_{obj}}{TP_{obj} + FP_{obj}} \tag{3.5}$$

$$Rappel = \frac{TP_{obj}}{TP_{obj} + FN_{obj}} \tag{3.6}$$

$$Fmesure = \frac{2\ .\ Precision\ .\ Rappel}{Precision + Rappel} \tag{3.7}$$

$$MODA(t) = 1 - \frac{FP_{obj}(t) + FN_{obj}(t)}{N_g(t)} \tag{3.8}$$

$$NMODA = 1 - \frac{\sum_{t=1}^{N_{frame}} FP_{obj}(t) + FN_{obj}(t)}{\sum_{t=1}^{N_{frame}} N_g(t)} \tag{3.9}$$

$$MODP(t) = \frac{\sum_{i=1}^{N_{mapped}(t)} \frac{|G_i \bigcap D_i|}{|G_i \bigcup D_i|}}{N_{mapped}(t)} \tag{3.10}$$

$$NMODP = \frac{\sum_{t=1}^{N_{frame}} MODP(t)}{N_{frame}} \tag{3.11}$$

Pour chaque image à l'instant t, $TP_{obj}(t)$ correspond au nombre d'objets correctement détectés, $FP_{obj}(t)$ au nombre de fausses détections (fausses alarmes), et $FN_{obj}(t)$ au nombre d'objets non détectés (miss-détections). ($N_g(t) = TP_{obj}(t) + FN_{obj}(t)$) est le nombre réel d'objets dans l'image. $N_{mapped}(t)$ est le nombre d'association entre les objets estimés et les objets de vérité terrain dans l'image t. G_i est la boîte englobant l'objet i dans la vérité terrain. D_i est la boîte fournie par le détecteur qui englobe l'objet i.

Afin de calculer les métriques, une association entre la vérité terrain et le résultat de détection doit être effectuée. Le critère d'association est l'intersection entre les boîtes fournies par le détecteur et celles de la vérité terrain. Une matrice de coût est construite à partir de pourcentage d'intersection des boîtes. Cette matrice constitue l'entrée de l'algorithme *Munkres*[3] qui est aussi appelé *algorithme Hongrois*. *Munkres* a pour but d'assurer une seule association au maximum par boîte détectée et par boîte de vérité terrain. Ainsi, les détections associées à une vérité terrain sont considérées comme des vrais positifs, les détections qui ne sont pas associées à une vérité terrain sont des faux positifs et les boîtes de la vérité terrain qui ne sont pas associées à des détections constituent les faux négatifs.

Les métriques Précision, Rappel et Fmesure, ne sont pas calculées au niveau pixel, contrairement à ce qui est fait dans [Sobral and Vacavant, 2014], car nous nous intéressons à la qualité de détection et non pas à la qualité de segmentation. Les métriques NMODA et NMODP traduisent respectivement la fidélité et la précision du résultat de détection [Kasturi et al., 2009]. MODA (équation (3.8)) représente le nombre de faux positifs et de faux négatifs par rapport au nombre réel d'objets dans une image. MODP (équation (3.10)) est le rapport entre le nombre de pixels appartenant à l'intersection de boîtes associées et le nombre de pixels appartenant à l'union de ces boîtes. NMODA et NMODP sont les normalisations de MODA et MODP. Les métriques sont calculées avec le code de [Poiesi and Cavallaro, 2015] [4].

La méthode proposée est comparée à la méthode Zipfian Sigma-Delta originale en utilisant les séquences vidéos PETS2009-S2L1 et CAVIAR-OneStopNoEnter2front. La vérité terrain de la séquence PETS2009 est celle fournie par [Poiesi and Cavallaro, 2015][4]. La vérité terrain de la séquence CAVIAR-OneStopNoEnter2front est récupérée du site la base de données CAVIAR. Les valeurs des paramètres du Zipfian Sigma-Delta sont fixés comme dans [Manzanera, 2007] ($Vmin = 2$, $Vmax = 254$, $Tv = 1$, $N = 2$). Les seuils du gradient

3. http ://csclab.murraystate.edu/bob.pilgrim/445/munkres.html
4. http ://www.eecs.qmul.ac.uk/ andrea/thdt.html

TABLE 3.3 – Comparaison de la qualité de détection de la méthode proposée avec l'algorithme Zipfian Sigma-Delta. Les paramètres sont fixée comme suit : $Vmin = 2$, $Vmax = 254$, $Tv = 1$, $N = 2$, th1=10, th2=200.

Base de données	Algorithme	Précision	Rappel	Fmesure	NMODA	NMODP
PETS2009 S2L1-view1	méthode proposée	0.93	0.85	0.89	0.79	0.66
	Zipfian Sigma-Delta	0.92	0.83	0.87	0.76	0.63
CAVIAR OneStop-NoEnter2front	méthode proposée	0.97	0.83	0.89	0.80	0.67
	Zipfian Sigma-Delta	0.90	0.83	0.87	0.74	0.69

sont fixés à $th1 = 10$ et $th2 = 200$. Le seuil intervenant dans la fusion des boîtes englobantes qui sont proches est fixé empiriquement à 0 pour la base de données PETS2009 (pas de fusion) et à 2 pour la base de données CAVIAR. Les résultats de détection obtenus sont présentés dans le tableau 3.3. .

Nous constatons que la méthode proposée fournit des meilleurs résultats selon les métriques Fmesure et NMODA, pour les deux bases de données, en comparaison avec l'algorithme Zipfian Sigma-Delta original. L'utilisation du gradient permet de diminuer le bruit généré à la sortie de Zipfian Sigma-Delta. La diminution du bruit engendre une séparation des objets proches et une réduction des pixels actifs connectés. L'algorithme d'analyse des composantes connectées peut alors filtrer ces pixels (ligne 4 de la figure 3.6).

La précision de détection (NMODP) obtenue avec notre méthode est meilleure pour la séquence PETS2009 et moins bonne pour la séquence CAVIAR. En effet, dans la séquence PETS2009, la sensibilité de Zipfian Sigma-Delta à la présence d'ombre engendre des boîtes englobantes larges par rapport à la vérité terrain (lignes 1 et 2 de la figure 3.6). Cependant, l'utilisation du gradient permet de diminuer l'effet de l'ombre sur le résultat de détection (ligne 2 de la figure 3.6). Les boîtes englobantes détectées avec notre méthode sont, alors, plus fidèles à la vérité terrain, ce qui permet d'augmenter la valeur de NMODP.

Pour la séquence CAVIAR, il y a beaucoup de bruit à la sortie de l'algorithme Zipfian Sigma Delta, à cause de la mauvaise qualité du capteur (ligne 4 de la figure 3.6). L'utilisation du gradient pour la diminution du bruit, engendre dans certains cas une discontinuité du contour de l'objet, ce qui implique une boîte englobante de taille réduite par rapport à la vérité terrain, ce qui fait diminuer la valeur de NMODP. Ce problème peut être résolu en augmentant le seuil de fusion des boîtes englobantes détectées. Par exemple en fixant le seuil de fusion à 10 pixels au lieu de 2, nous obtenons : Précision=0.98, Rappel=0.84, Fmesure=0.91, NMODA=0.83 et NMODP=0.69. L'utilisation d'un seuil élevé engendre cependant une mauvaise séparation d'objets proches (ligne 6 de la figure 3.6). La valeur du seuil de fusion doit assurer un bon compromis entre la précision de la détection et la séparation d'objets proches. Les sorties de détection ainsi que les images de segmentation sont présentées dans la figure 3.6 pour les bases de données CAVIAR, PETS2009 et une expérimentation effectuée au laboratoire.

FIGURE 3.6 – Exemples de résultats de détection. La première colonne est le résultat de segmentation obtenu avec la méthode Zipfian Sigma-Delta. La deuxième colonne est le résultat de segmentation obtenu avec notre méthode. La troisième colonne représente le résultat de détection : Orange : Zipfian Sigma-Delta et Vert : Notre méthode.

Afin de démontrer la robustesse de la méthode proposée à l'effet de *ghost*, nous avons initialisé l'arrière plan avec une image contenant des objets. Les résultats de l'expérience sont présentés dans la figure 3.7. Ces résultats peuvent être améliorés en adaptant les seuils $th1$ et $th2$.

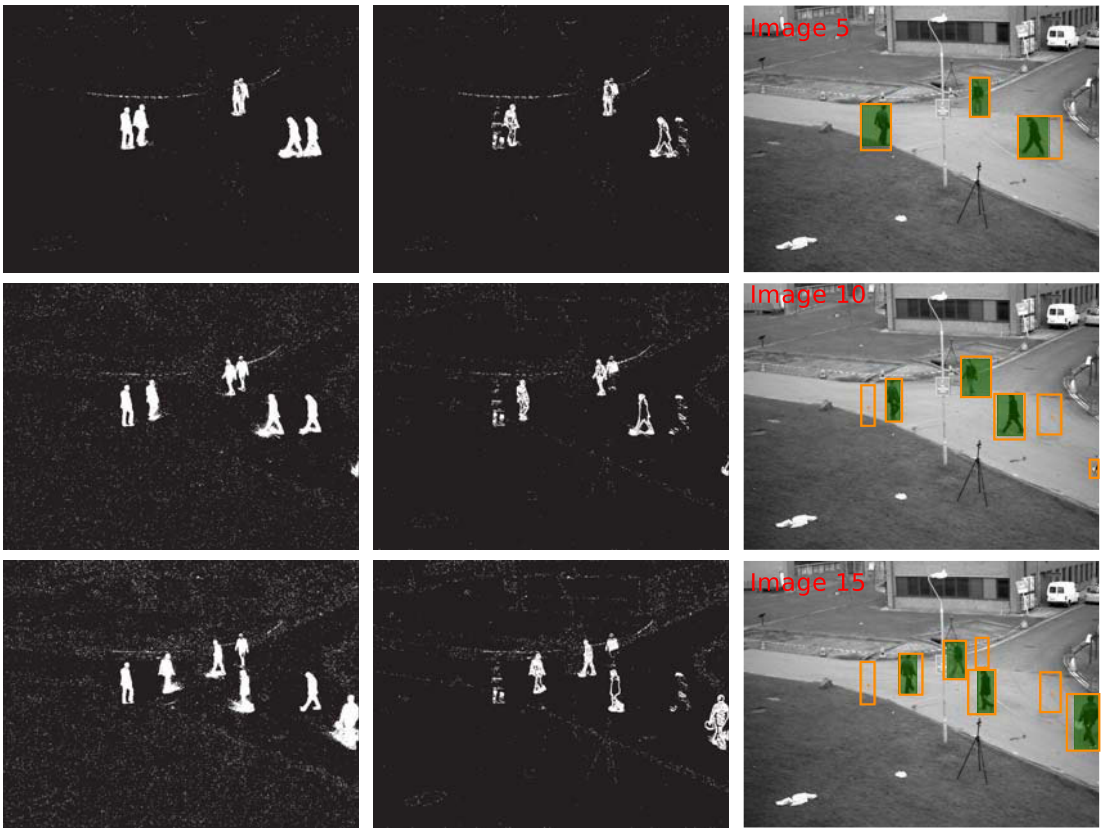

FIGURE 3.7 – Résultats de détection obtenu lorsque l'arrière plan est initialisé avec une image contenant des objets. La première colonne est le résultat de segmentation obtenu avec la méthode Zipfian Sigma-Delta. La deuxième colonne est le résultat de segmentation obtenu avec notre méthode. La troisième colonne représente le résultat de détection : Orange : Zipfian Sigma-Delta et Vert : Notre méthode.

3.3.2 Besoins en mémoire

Les besoins en mémoire de la détection dépendent de la mémoire requise par l'algorithme de soustraction de fond et par l'algorithme de l'analyse des composantes connectées.

L'algorithme Zipfian Sigma-Delta nécessite 16 bits par pixel. En effet, 8 bits sont utilisés pour chaque pixel de l'image de l'arrière plan M et 8 bits pour chaque pixel de l'image de la variance V. Le calcul de l'amplitude du gradient s'effectue à la volée. Ainsi, seulement 3 lignes de l'image sont nécessaires à stocker pour l'implémentation de l'équation (3.1). Au final, le besoin mémoire par pixel reste inférieur à 3 octets, ce qui est inférieur aux besoins en mémoire indiqués dans le tableau 3.2, des autres méthodes.

La mémoire requise par l'algorithme d'analyse des composantes connectées dépend

de la largeur de l'image, du nombre de labels et du type de caractéristiques extraites (coordonnées des boîtes, centre de gravité, *etc.*). La mémoire de l'algorithme est faiblement influençable par la résolution de l'image car la labellisation s'effectue à la volée et seule la taille des caractéristiques est dépendante de la résolution. L'empreinte mémoire de l'analyse des composantes connectées a été mesurée dans [Ma et al., 2008] pour des images de résolution 640 × 480. Pour cette résolution d'image, le pire cas implique 76800 labels avec 17 bits pour chaque label, quand la caractéristique de surface (nombre de pixels composant un objet) est extraite. Ainsi, une mémoire totale de 3,3 ko est nécessaire pour cette méthode, contrairement à la méthode classique parcourant deux fois l'image qui nécessite 796,875 ko. Dans notre implémentation en C, en comptant le nombre de bits utilisés par nos structures de données, nous obtenons une empreinte mémoire de 208×"la largeur de l'image" donc 17,5 ko pour la résolution 640 × 480 et 8,5 ko pour la résolution 320 × 240. La fusion des boîtes proches engendre une empreinte mémoire qui dépend du nombre maximum d'objets détectés. 168 bits par objet sont nécessaires.

La mémoire utilisée par notre implémentation est supérieure à la mémoire du pire cas de [Ma et al., 2008] à cause de la différence des caractéristiques extraites. Dans [Ma et al., 2008] seule la surface des objets est extraite. Dans notre cas, en plus de la surface de l'objet, nous extrayons les coordonnées de la boîte englobant l'objet, et les coordonnées de son centre de gravité. De plus, contrairement à une implémentation sur un FPGA où la dynamique d'une variable peut être contrôlée à un bit près, dans l'implémentation en C, nous disposons seulement des dynamiques 8, 16, 32 et 64 bits. Par exemple, pour représenter une valeur 256, nous sommes obligés d'utiliser une variable sur 16 bits ce qui diminue l'efficacité mémoire.

3.3.3 Complexité de calcul

3.3.3.1 Complexité théorique

La complexité de calcul de détection est liée à la soustraction de fond et à l'analyse des composantes connectées. L'algorithme Zipian Sigma-Delta nécessite 10 comparaisons sur 8 bits, 5 additions sur 8 bits et l décalages, avec $N = 2^l$. L'utilisation de l'amplitude du gradient par pixel rajoute, dans le pire cas, une complexité de 4 comparaisons sur 8 bits, 3 additions sur 8 bits et 1 multiplication sur 8 bits dont le résultat est sur 16 bits. L'amplitude du gradient n'est calculée que pour les pixels qui n'appartiennent pas à l'arrière plan, ce qui représente une très faible portion dans l'image (figure 3.6). Toutes les opérations sont effectuées sur des entiers. Si nous considérons que les opérations effectuées ont le même coût, ce qui dans la réalité dépend de l'architecture du processeur, pour l'opération de multiplication, l'utilisation de l'information du gradient ajoute une complexité qui est égale à 50% dans le pire des cas. Cependant, ce pire cas est seulement vérifié pour les pixels actifs du résultat final.

La segmentation en utilisant l'algorithme First-Order low pass filter nécessite 3 additions, 2 comparaisons et 1 multiplication. Cependant, toutes ces opérations sont sur des nombres réels de type float ou double à cause de la précision nécessaire pour la mise à jour de l'arrière plan qui est introduite par le paramètre α compris entre 0 et 1 (équation (1.1)).

Les opérations nécessitent donc une FPU [5] pour une exécution rapide. La rapidité d'exécution dépend de l'architecture de la FPU. Dans le cas d'absence de FPU, les opérations sur les réels sont émulées avec des opérations entières, ce qui engendre un nombre d'opérations supérieur à celui de la méthode proposée.

Dans notre implémentation de l'algorithme d'analyse des composantes connectées, le nombre d'opérations arithmétiques et logiques nécessaires pour le traitement d'un pixel, dans le pire cas, se résume à 11 comparaisons sur 8-bits, 3 comparaisons sur 16 bits, et 6 additions sur 32 bits. Cependant, plusieurs accès mémoires pour la lecture-écriture sont nécessaires. Toutes les opérations effectuées pour la soustraction de fond et pour l'analyse des composantes connectées sont des opérations entières.

Dans l'algorithme d'analyse des composantes connectées, seuls les pixels actifs (classifiés comme pixels appartenant à un objet par la segmentation de fond) sont traités. L'ajout de gradient comme caractéristique réduit le nombre de pixels actifs ce qui implique une réduction du temps d'analyse de l'image entière.

3.3.3.2 Complexité mesurée sur un processeur

Le nombre de cycles d'horloge nécessaires pour l'exécution des deux méthodes de détections, qui incluent la soustraction du fond et l'analyse des composantes connectées, est mesuré sur un processeur x86 core 2 duo fonctionnant à 3 GHz en utilisant l'outil *callgrind* [Nethercote et al., 2006]. Pour la séquence PETS2009, la détection basée sur la méthode de soustraction de fond proposée nécessite en moyenne 5.15% de plus de cycles d'horloge que la détection utilisant la méthode Zipfian Sigma-Delta originale. Pour la séquence CAVIAR, notre méthode de détection nécessite en moyenne 11% de plus de cycles d'horloge que la détection utilisant la méthode Zipfian Sigma-Delta originale. Le pourcentage est élevé pour la séquence CAVIAR, car la sortie de Zipfian Sigma-Delta est bruitée ce qui nécessite un calcul du gradient pour plus de pixels. Ces résultats obtenus confirment la faible complexité introduite par l'ajout de l'information du gradient.

3.4 Conclusion

Dans ce chapitre, nous avons présenté une méthode de détection d'objets à base de soustraction de fond. Après l'analyse des méthodes existantes, nous avons sélectionné l'algorithme Zipfian Sigma-Delta. Les limites de cet algorithme sont le bruit de segmentation, la sensibilité à la présence d'ombre et la mauvaise gestion de l'effet *"ghost"*. Afin de réduire le bruit, nous avons ajouté l'information du gradient calculée à partir de l'image d'entrée. La méthode présentée est comparée à l'algorithme Zipfian Sigma-Delta en utilisant les séquences vidéos des bases de données PETS2009 et CAVIAR. Les résultats obtenus montrent une amélioration de la qualité de détection grâce à l'ajout de l'information du gradient de l'image d'entrée tout en gardant une faible complexité.

Dans le prochain chapitre, nous nous intéressons au deuxième module composant la chaîne qui est l'algorithme de suivi multi-objet.

5. FPU : Floating Point Unit est un accélérateur matériel permettant de réaliser des opérations sur des nombres réels avec un codage en virgule flottante

Chapitre 4

Suivi multi-cible

Sommaire

Dans la thèse nous nous intéressons au suivi par détection. L'algorithme de suivi a pour but de relier les résultats de détection issues des images successives, afin de déterminer la trajectoire des objets et d'assurer une cohérence des identités. Suite à la discussion de la section 2.2.1.3, les algorithmes de suivi exploitant la caractéristique de mouvement sont choisis afin d'assurer une faible complexité. Chaque objet est représenté par un vecteur contenant des informations sur son état qui peuvent être les coordonnées de sa position, les coordonnées de sa vitesse, sa taille et son ratio de taille, *etc.* L'objectif de l'algorithme de suivi est d'estimer conjointement le nombre et les états des objets qui varient au cours de temps. L'estimation conjointe est basée sur l'ensemble des observations fournies par un détecteur imparfait présentant des non-détections et des fausses alarmes.

Dans ce chapitre, nous analysons les algorithmes de suivi adéquats pour notre contexte afin de sélectionner l'algorithme assurant un bon compromis entre la complexité de calcul, la quantité de mémoire et la qualité de suivi. Une description et une analyse de la complexité de l'algorithme sélectionné sont effectuées. Par la suite, les effets du portage de l'algorithme approximé sont étudiés pour une architecture embarquée de type ARM, en utilisant des données synthétiques. Afin de comprendre les limites de l'algorithme sélectionné dans un scénario de suivi visuel d'objets nous avons utilisé les observations fournies par un détecteur basé sur la soustraction de fond.

4.1 Description des méthodes de suivi multi-cible

4.1.1 Introduction

Le choix d'un algorithme de suivi est contraint par sa complexité de calcul et ses besoins en mémoire, quand des architectures embarquées sont utilisées. Nous avons présenté, dans la section 1.2.3, les différentes méthodes utilisées pour la résolution du problème de suivi d'objets quand la caractéristique de mouvement est utilisée. Du point de vue de la complexité, les méthodes peuvent être classifiées en deux catégories : les méthodes se basant sur un historique d'observations et celles dont l'historique se résume au dernier état de l'objet.

L'utilisation d'un historique peut s'avérer utile pour la gestion des initialisations et des terminaisons de pistes ainsi que pour la rectification des erreurs d'association. Cependant, son inconvénient principal est la grande quantité de mémoire requise ainsi que la complexité de calcul qui est dans la plupart des cas exponentielle. Un autre problème est que la décision optimale s'effectue en différé ce qui crée une latence de traitement.

Les méthodes n'utilisant pas d'historique considèrent que tout le passé de l'objet est conservé dans son état actuel. Le problème dans ces méthodes est l'impossibilité de rectifier une erreur qui s'est produite dans le passé. De plus, l'initialisation et la terminaison de pistes sont gérées en se basant sur des a priori ou/et une logique externe. Cependant, ces méthodes présentent un grand avantage pour les architectures embarquées vu que le nombre de paramètres à traiter et à mémoriser est faible.

La complexité liée à la combinatoire de la théorie des graphes pour une solution optimale est illustrée dans la figure 4.1, où chaque arête présente une possibilité d'association. Dans les méthodes à base de la théorie des graphes, seuls les graphes bipartis sont considérés car ils sont les plus prometteurs pour un fonctionnement sur une architecture embarquée. Plus précisément nous nous intéressons à la méthode GNN où un filtre de Kalman est utilisé pour le filtrage de la trajectoire.

Dans ce qui suit, nous faisons un rappel sur le filtre du Kalman et nous décrivons les méthodes GNN, JPDA, MHT et le filtre de Bayes multi-cible.

4.1.2 Filtre de Kalman

Le filtre de Kalman est la solution optimale lorsque le filtre de Bayes classique, pour le suivi d'un seul objet, est appliqué a des systèmes linéaires affectés d'un bruit Gaussien. Dans ce cas, l'état d'un objet est décrit par une Gaussienne dont la moyenne et la matrice

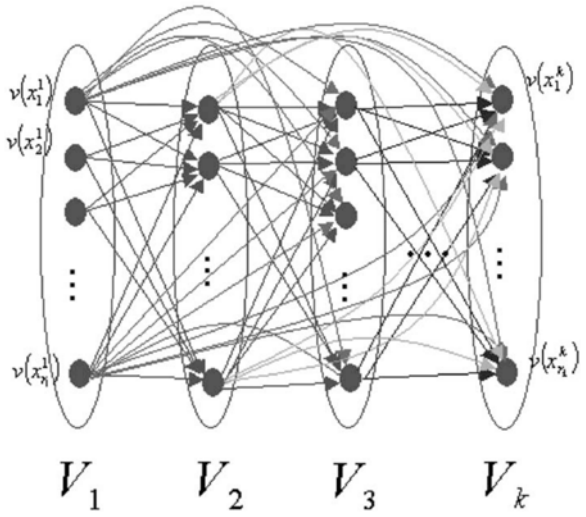

FIGURE 4.1 – Association pistes-observations avec la théorie des graphes [Shafique and Shah, 2005]. V_k est l'ensemble d'observation à l'instant k. Les liens entre les observations sont les différentes possibilités d'association. Le nombre de liens montre la complexité de la théorie des graphes pour une solution optimale.

de covariance représentent, respectivement, le vecteur d'état et la matrice de covariance de l'objet. Le vecteur d'état contient les différents paramètres qui décrivent l'état de l'objet : position, vitesse, couleur, ... Le filtre de Kalman permet de faire évoluer la Gaussienne représentant l'objet au cours de temps. L'évolution s'effectue en deux étapes : prédiction et mise à jour.

L'étape de prédiction se base sur un modèle d'évolution linéaire et Gaussien, donné par l'équation (4.1), qui décrit la dynamique de l'objet.

$$\mathrm{x}_{k|k-1} = F_k x_{k-1|k-1} + \mathrm{w}_{k-1} \tag{4.1}$$

— $\mathrm{x}_{k-1|k-1}$ et $\mathrm{x}_{k|k-1}$ sont les vecteurs d'état avant et après la prédiction. Ils sont de taille $n \times 1$.
— F_k est la matrice de transition qui est de taille $n \times n$.
— w_k est le vecteur du bruit d'évolution qui est un bruit Gaussien de moyenne nulle et de covariance Q_k (matrice $n \times n$). La taille du vecteur est $n \times 1$.

La prédiction a pour objectif de prédire l'état $\mathrm{x}_{k|k-1}$ et la covariance $\mathrm{P}_{k|k-1}$ de l'objet à partir de son état $\mathrm{x}_{k-1|k-1}$ et de sa covariance $\mathrm{P}_{k-1|k-1}$ précédents, en utilisant les équations (4.2) et (4.2).

$$\mathrm{x}_{k|k-1} = F_k \, \mathrm{x}_{k-1|k-1} \tag{4.2}$$

$$\mathrm{P}_{k|k-1} = F_k \, \mathrm{P}_{k-1|k-1} \, {F_k}^T + Q_k \tag{4.3}$$

L'étape de mise à jour consiste à corriger l'état $\mathrm{x}_{k|k-1}$ et la covariance $\mathrm{P}_{k|k-1}$ prédits de l'objet en utilisant les observations fournies par le détecteur et le modèle de mesure linéaire et Gaussien donné par l'équation (4.4).

$$\mathrm{z}_k = H_k\,\mathrm{x}_k + \mathrm{v}_k \tag{4.4}$$

— H_k est la matrice d'observation de taille $m \times n$.
— z_k est le vecteur d'observation qui est de taille $m \times 1$.
— v_k est le vecteur du bruit de mesure qui est un bruit Gaussien de moyenne nulle et de covariance R_k (matrice $m \times m$). La taille du vecteur est $m \times 1$.

Le résultat de cette étape est un état corrigé $\mathrm{x}_{k|k}$ et une covariance corrigée $\mathrm{P}_{k|k}$ fournis respectivement par les équations (4.5) et (4.6).

$$\mathrm{x}_{k|k} = \mathrm{x}_{k|k-1} + \mathrm{K}_k(z_k - \hat{z}_k) \tag{4.5}$$

Où $\hat{z}_k = H_k\,\mathrm{x}_{k|k-1}$ est l'observation prédite par le modèle d'état.

$$\mathrm{P}_{k|k} = \mathrm{P}_{k|k-1} - K_k H_k\,\mathrm{P}_{k|k-1} \tag{4.6}$$

Où K_k représente le gain de Kalman qui est donné par l'équation (4.7).

$$\mathrm{K}_k = \mathrm{P}_{k|k-1}\,{H_k}^T {S_k}^{-1} \tag{4.7}$$

S_k est la covariance de l'innovation $z_k - \hat{z}_k$ donnée par l'équation (4.8).

$$S_k = H_k\,\mathrm{P}_{k|k-1}\,{H_k}^T + R_k \tag{4.8}$$

4.1.3 Global Nearest Neighbor

Le principe de GNN [Blackman and Popoli, 1999] est de déterminer l'association minimisant la somme des distances entre les pistes et les observations tout en respectant les conditions suivantes :

— Une observation peut être associée à une piste au maximum
— Une piste peut être associée à une observation au maximum

Dans le GNN, un filtre de Kalman est utilisé pour la prédiction et la mise à jour des pistes. Dans ce cas, une observation et une matrice d'innovation sont prédites pour chaque piste. La prédiction des observations et des covariances d'innovation sont déterminées en projetant les états et les covariances des pistes sur l'espace d'observation en utilisant le modèle d'évolution et le modèle de mesure. La distance entre les observations prédites et les observations fournies par le détecteur est calculée avec la distance de Mahalanobis décrite par l'équation (4.9). Chaque distance définit un coût d'association. En appliquant des contraintes sur le mouvement des objets, seules les associations correspondant à des distances inférieures à un seuil sont considérées. Les distances des autres associations sont remplacées par la valeur du seuil. Cette étape est appelée fenêtrage statistique (*"gating"*) et permet de déterminer les observations susceptibles d'être associées à une piste donnée. La figure 4.2 décrit le processus du fenêtrage statistique.

$$d_{Mahalanobis} = (z(i) \quad \hat{z}(c))^T S(c)^{\ 1}(z(i) \quad \hat{z}(c)) \tag{4.9}$$

$z(i)$ est la $i^{\text{éme}}$ observation fournie par le détecteur. $\hat{z}(c)$ et $S(c)$ sont respectivement l'observation prédite et la covariance d'innovation de la piste de la cible c.

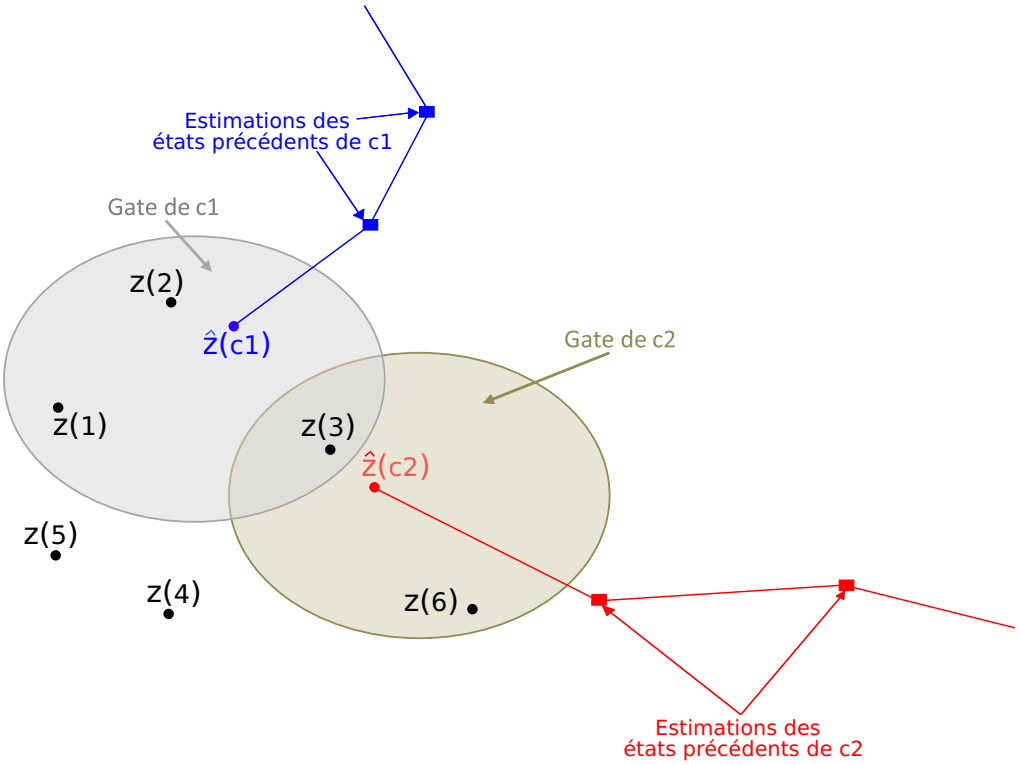

FIGURE 4.2 – Fenêtrage statistique (*Gating*). $z(i)$ sont les observations fournies par le détecteur. $\hat{z}(cj)$ est l'observation prédite avec le modèle d'état pour la piste cj. $c1$ et $c2$ sont des pistes de cibles. La fenêtre statistique (*Gate*) d'une cible est l'espace de mesure où l'observation de la cible est susceptible d'apparaître avec une probabilité prédéfinie. Toutes les observations à l'extérieur de la fenêtre statistique ne peuvent pas être associées à la cible.

Tout le processus du fenêtrage statistique se résume dans une matrice de coût dont le nombre de lignes et le nombre de colonnes correspondent respectivement au nombre de pistes et au nombre d'observations. Chaque élément c, i de la matrice correspond à la distance de l'association de la piste c à l'observation i. L'association pistes-observations optimale est déterminée à partir de la matrice de coût en utilisant la méthode exhaustive ou des algorithmes d'optimisation comme l'algorithme Hongrois appelé aussi l'algorithme de Munkres [Bourgeois and Lassalle, 1971] [1] ou l'algorithme Auction [Blackman and Popoli, 1999]. La méthode exhaustive consiste à énumérer toutes les associations possibles. L'association optimale est celle dont la somme des distances correspondant aux couples piste-observation est minimale. La méthode exhaustive est coûteuse

1. http ://csclab.murraystate.edu/bob.pilgrim/445/munkres_old.html

en calcul. L'algorithme Hongrois est d'une complexité $O(N^2M)$, M et N sont respectivement le minimum et le maximum entre le nombre de pistes et le nombre d'observations. L'état de chaque piste est mis à jour en utilisant l'observation qui lui est associée.

À l'issue du GNN, certaines pistes et certaines observations ne figurent pas dans l'association optimale. En effet, ce cas se produit quand le nombre d'objets est variable au cours de temps, en présence de fausses alarmes ou en présence de non-détections. Ainsi, il faut des heuristiques supplémentaires pour résoudre ce problème. Les heuristiques fournissent la logique permettant d'initialiser des nouvelles pistes et de terminer les pistes, respectivement, à partir des observations et des pistes ne figurant pas dans l'association optimale. L'initialisation et la terminaison de pistes permettent d'estimer à chaque instant le nombre d'objets présents dans la scène, ce que le GNN ne fait pas intrinsèquement, d'où sa première limitation. La deuxième limitation est sa faible robustesse aux faux positifs et aux faux négatifs de détecteur et sa faible capacité à assurer une association correcte quand les cibles sont proches.

4.1.4 Joint Probabilistic Data Association

Contrairement à la méthode GNN qui utilise une seule observation pour la mise à jour de la piste, le filtre JPDA [Bar-Shalom and Li, 1995, Bar-Shalom et al., 2009] considère toutes les observations de la cible présentes dans sa fenêtre statistique. JPDA tient compte des non-détections et des fausses alarmes de détecteur. L'utilisation d'une mise à jour en considérant toutes les observations à l'intérieur de la fenêtre statistique assure une robustesse aux fausses alarmes. Cependant comme le GNN, l'initialisation et la terminaison des pistes ne sont pas gérées intrinsèquement, ce qui nécessite une logique supplémentaire pour l'estimation du nombre de cibles.

La première étape de JPDA est le fenêtrage statistique qui a pour but de déterminer quelles observations sont susceptibles d'être associées à une piste. La prédiction est équivalente à celle du GNN qui est une prédiction à base de Kalman. En se basant sur le résultat du fenêtrage statistique, une matrice d'hypothèses est générée en formulant les hypothèses d'association pistes-observations possibles qui respectent les deux conditions :

- une observation a une source unique
- une cible génère au maximum une observation tout en prenant en considération les non-détections

La génération d'hypothèses est un processus combinatoire. Le nombre d'hypothèses augmente avec le nombre de cibles ayant des observations en commun et avec le nombre d'observations partagées entre cibles. La probabilité de chaque hypothèse est calculée avec l'équation (4.10), pour le cas où le nombre de fausses alarmes est modélisé avec une loi de Poisson.

$$p_{hypo}(h) = p(h|Z^k) = \frac{1}{cst} \prod_i \{\lambda^{-1} f_c[z_k(i)]\}^{\tau_i} \prod_c P_D^{\delta_c} (1 - P_D)^{(1-\delta_c)} \tag{4.10}$$

Avec :

- h est une hypothèse d'association pistes-observations
- cst est une constante de normalisation

— $p_{hypo}(h)$ est la probabilité de l'hypothèse h
— Z^k est l'ensemble des observations de l'instant 0 à l'instant k
— $f_c[z_k(i)] = \mathcal{N}(\ z_k(i); \hat{z}_k(c), S_k(c)\ )$ est la valeur de la vraisemblance quand la piste c est associée à l'observation i
— $z_k(i)$ est l'observation i à l'instant k
— $\hat{z}_k(c)$ est l'observation prédite de la piste c et $S_k(c)$ est sa covariance d'innovation
— τ_i est une variable binaire égale à 1 si l'observation i est associée à la piste c dans l'hypothèse h et 0 dans le cas contraire
— De même δ_c est une variable binaire égale à 1 si la cible correspondant à la piste c est détectée dans l'hypothèse h et 0 dans le cas contraire.
— λ est la densité spatiale des fausses alarmes.

En supposant que les états des cibles, conditionnés par le passé, sont mutuellement indépendants, la probabilité d'association $\beta_k(c,i)$ de la piste c à l'observation i à l'instant k est calculée en sommant les probabilités des hypothèses contenant cette association, comme indiqué dans l'équation (4.11).

$$\beta_k(c,i) = \sum_{h \in HYP_{ci}} p_{hypo}(h) \tag{4.11}$$

Avec HYP_{ci} l'ensemble des hypothèses où l'association de la piste c avec l'observation i est considérée.

La dernière étape du filtre JPDA est la mise à jour des états (équation (4.12)) et des covariances (équation (4.15)) des cibles.

$$\mathrm{x}_{k|k}(c) = \hat{\mathrm{x}}_{k|k-1}(c) + W_k(c) v_k(c) \tag{4.12}$$

Avec

$$v_k(c) = \sum_{i=1}^{M_k} \beta_k(c,i)[z_k(i) - \hat{z}_k(c)] \tag{4.13}$$

$$W_k(c) = \mathrm{P}_{k|k-1}(c) {H_k}^T S_k(c)^{-1} \tag{4.14}$$

— $\mathrm{x}_{k|k}(c)$ est l'état de la piste c après mise à jour
— $\hat{\mathrm{x}}_{k|k-1}(c)$ est l'état de la piste c après prédiction
— $W_k(c)$ est le gain de Kalman de la piste c
— $v_k(c)$ est l'innovation pondérée par la probabilité d'association
— M_k est le nombre d'observations fournies par le détecteur à l'instant k
— $\mathrm{P}_{k|k-1}(c)$ est la matrice de covariance de la piste c après prédiction
— H_k est la matrice d'observation
— A^T est la transposée de la matrice A

$$\mathrm{P}_{k|k}(c) = \beta_k(c,0)\mathrm{P}_{k|k-1}(c) + [1 - \beta_k(c,0)]\mathrm{P}^j_{k|k}(c) + \tilde{\mathrm{P}}_k(c) \tag{4.15}$$

Avec

$$\mathrm{P}^{j}_{k|k}(c) = \mathrm{P}_{k|k-1}(c) - W_k(c)S_k(c)W_k(c)^T \tag{4.16}$$

$$\tilde{\mathrm{P}}_k(c) \triangleq W_k(c)\left[\sum_{i=1}^{M_k}\beta_k(c,i)[z_k(i)-\hat{z}_k(c)][z_k(i)-\hat{z}_k(c)]^T - v_k(c)v_k(c)^T\right]W_k(c)^T \tag{4.17}$$

$\beta_k(c,0)$ est la probabilité que la piste c n'est associée à aucune observation.

L'avantage du JPDA est sa robustesse aux erreurs de détection. Cette robustesse est assurée grâce à l'association de plusieurs observations à une même piste. Les ressources nécessaires pour faire fonctionner JPDA sont non déterministes car le nombre d'hypothèses est dépendant du scénario de suivi. Ainsi, le pire cas doit être considéré pour pouvoir allouer les ressource nécessaires sur le calculateur embarqué. Cependant, le nombre d'hypothèses dans le pire cas est élevé à cause de la combinatoire d'association. Étant donné ce grand nombre d'hypothèses, la complexité de calcul et les besoins en mémoire sont élevés. Comme le GNN, le filtre JPDA ne gère pas l'initialisation et la terminaison des pistes ce qui nécessite une logique supplémentaire.

4.1.5 Multiple Hypothesis Tracking

Le principe de la méthode MHT [Cox and Hingorani, 1996, Blackman, 2004] est de formuler à partir des observations reçues et des pistes existantes, toutes les hypothèses valables possibles en respectant deux règles :

- il ne faut pas associer une même observation à plusieurs cibles dans la même hypothèse
- il ne faut pas que deux observations soient associées à la même cible dans la même hypothèse

Les hypothèses formulées à l'instant actuel sont considérées comme des pistes existantes à l'instant suivant auxquelles des nouvelles observations seront associées, d'où une complexité en exponentielle. La figure 4.3 montre comment une hypothèse affirmant l'existence de deux pistes à l'instant $k-1$, se propage à l'instant k quand les observations fournies par le détecteur suivent le scénario de la figure 4.2.

En se basant sur les associations pistes-observations, le score de chaque piste est calculé en utilisant le ratio de vraisemblance (*Log-Likelihood Ratio*). Le score d'une piste permet de confirmer ou de supprimer la piste selon sa valeur. Une fois que les scores des pistes sont déterminés, le score d'une hypothèse est évalué en sommant les scores des pistes contenues dans l'hypothèse. La probabilité d'une hypothèse est calculée à partir de son score. Les hypothèses de faibles probabilités sont supprimées afin de réduire la combinatoire. Les probabilités des hypothèses restantes permettent de calculer les probabilités des pistes. La probabilité d'une piste est la somme des probabilités des hypothèses contenant la piste. Les pistes de faibles probabilités sont supprimées. Les probabilités des pistes permettent de définir l'origine de chaque observation. Dans le but de limiter le nombre d'hypothèses, seule une fenêtre temporelle de largeur T est considérée pour l'historique. Cette fenêtre

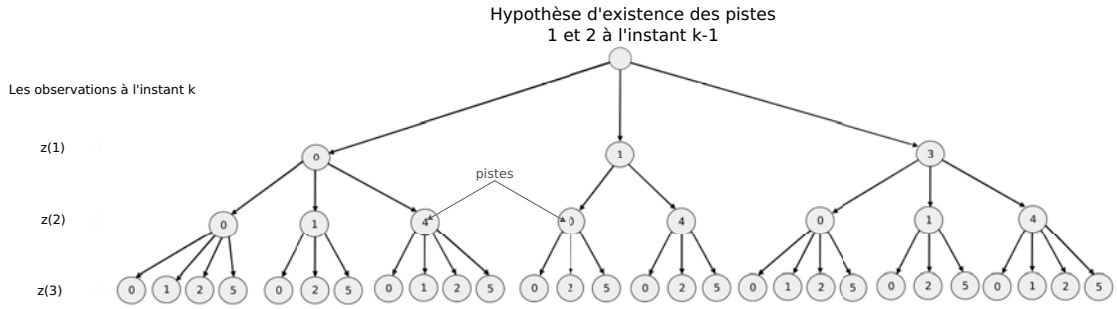

FIGURE 4.3 – Propagation d'hypothèses dans un MHT orienté hypothèse. L'intérieur des cercles représente la piste de la cible engendrant l'observation : 0 signifie que l'observation est une fausse alarme, 1 signifie que l'observation est générée par la cible dont la piste est 1 et 2 signifie que l'observation est générée par la cible dont la piste est 2. 3, 4 et 5 signifient que l'observation est générée par une nouvelle cible. Chaque étage de l'arbre représente une observation.

permet de sélectionner à un instant k l'hypothèse la plus probable à l'instant k T et d'éliminer les autres hypothèses.

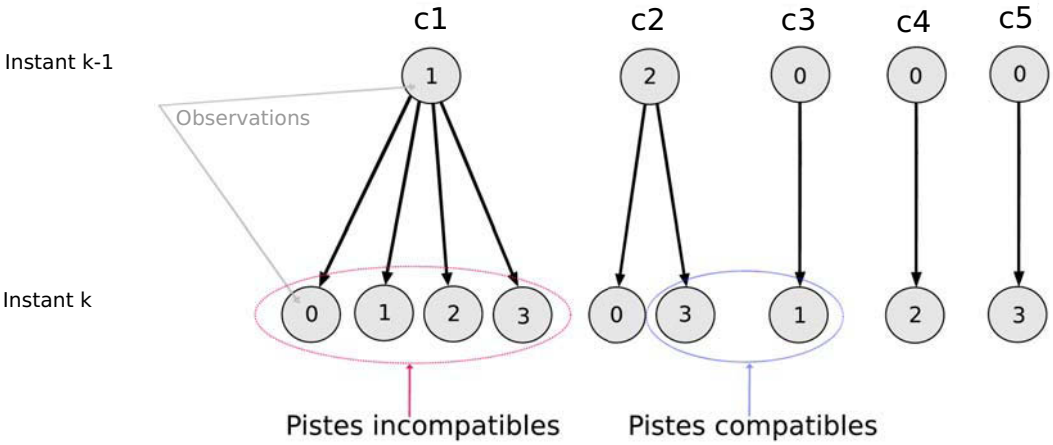

FIGURE 4.4 – Propagation de pistes dans un MHT orienté piste. *ci* est la piste de la cible i. L'intérieur des cercles représente l'observation engendrée par la cible de la piste : 0 signifie qu'aucune observation n'est générée par la cible, $i = 1, 2, 3$ signifie que l'observation i est générée par la cible. Les liens entre les observations à l'instant k 1 et les observations à l'instant k d'une cible représente les différentes possibilités d'évolution de la piste de la cible.

Comme le nombre de pistes est inférieur au nombre d'hypothèses, une variante de MHT propage les pistes d'un instant k 1 à un instant k, en considérant toutes les observations (figure 4.4). L'hypothèse d'association est formulée après la propagation des pistes. Les pistes apparatenant à la même branche sont des pistes compatibles et ne peuvent pas figurée dans la même hypothèse d'association. Pour plus de détail sur les implémentations des MHT orientées hypothèses et orientées pistes, le lecteur est invité à

consulter [Amditis et al., 2012].

Même avec tous ces efforts pour réduire la combinatoire, le nombre d'hypothèses explose de manière exponentielle ce qui nécessite un espace de stockage et une puissance de calcul considérables. À cause de la combinatoire non déterministe, les ressources nécessaires pour faire fonctionner un MHT dépendent du scénario de suivi, ce qui n'est pas adéquat pour des architectures embarquées de calcul.

4.1.6 Le filtre de Bayes multi-cible

4.1.6.1 Principe

Le filtre de Bayes multi-cible est le résultat d'extension du filtre de Bayes classique au cas de plusieurs cibles. Contrairement aux méthodes présentées jusqu'à maintenant, ce filtre n'est pas une méthode d'association et ne traite pas chaque cible indépendamment des autres, ce qui lui permet d'éviter le problème combinatoire des méthodes d'association. Les états des cibles sont représentés avec un état global multimodal et l'ensemble d'observations avec une observation globale. L'état global et l'observation globale sont modélisés avec un ensemble aléatoire fini ou *Random Finite Set* (RFS).

Un RFS est défini par une distribution de probabilité discrète qui caractérise le nombre d'éléments (cardinalité) de l'ensemble et une famille de distributions conjointes qui caractérise la distribution des éléments de l'ensemble sachant la cardinalité [Vo, 2008]. L'état global à l'instant k est :

$$\mathrm{X_k} = \{x_{1,k}, x_{2,k}, x_{3,k}, \ldots, x_{n(k),k}\} \tag{4.18}$$

Avec $n(k)$ le nombre de cibles à l'instant k et $x_{i,k}$ le vecteur d'état d'une cible.

L'observation à l'instant k est :

$$Z_k = \{z_{1,k}, z_{2,k}, z_{3,k}, \ldots, z_{m(k),k}\} \tag{4.19}$$

Avec $m(k)$ le nombre d'observations à l'instant k et $z_{j,k}$ un vecteur d'observation.

Les étapes du filtre sont la prédiction (équation (4.20)) et la correction (équation (4.21)).

Prédiction

$$\pi_{k|k-1}(\mathrm{X_k}|Z^{(k-1)}) = \int f_{k|k-1}(\mathrm{X}_k|\mathrm{X}_{k-1})\pi_{k-1|k-1}(\mathrm{X}_{k-1}|Z^{(k-1)})\mathrm{dX}_{k-1} \tag{4.20}$$

$Z^{(k-1)} = \bigcup_{i=0}^{k-1} Z_i$: toutes les observations de l'instant 0 à l'instant $k-1$.
$\pi_{k-1|k-1}(\mathrm{X}_{k-1}|Z^{(k-1)})$: est la densité a posteriori multi-cible à l'instant $k-1$
$f_{k|k-1}(\mathrm{X}_k|\mathrm{X}_{k-1})$: est la densité de transition multi-cible définie à partir d'un modèle d'évolution des cibles.

Correction

$$\pi_{k|k}(\mathrm{X_k}|Z_k) = \frac{g_k(Z_k|\mathrm{X}_k)\pi_{k|k-1}(\mathrm{X}_k|\mathrm{Z^{(k-1)}})}{\int g_k(Z_k|\mathrm{X}_k)\pi_{k|k-1}(\mathrm{X}_k|\mathrm{Z^{(k-1)}})\mathrm{dX}_k} \tag{4.21}$$

$g_k(Z_k|\mathrm{X}_k)$ est la vraisemblance multi-cible construite à partir du modèle d'observation.

Comme le filtre de Bayes classique, le filtre multi-cible n'est pas implémentable à partir des équations de prédiction et de correction. Ainsi, des approximations ont été faites et les filtres Probability Hypothesis Density (PHD) et cardinalized Probability Hypothesis Density (CPHD) ont été construits en faisant des hypothèses sur les RFS utilisés.

Pour le filtre PHD le RFS est supposé être un RFS de Poisson ce qui signifie que la densité discrète caractérisant la cardinalité du RFS suit une loi de Poisson. La prédiction, dans ce cas, consiste à propager le moment d'ordre 1 de la densité multi-cible. Le moment d'ordre 1 est appelé PHD ou fonction d'intensité du RFS modélisant l'état global.

La naissance de nouvelles pistes est modélisée par deux RFS dans le filtre PHD : naissance spontanée et naissance à partir de division de cibles existantes. Le filtre PHD intègre des paramètres qui sont : la probabilité de détection de cibles, la probabilité de survie de cibles, la densité de fausses alarmes et l'information a priori sur la naissance de cibles (par division et spontanée).

Dans le filtre CPHD, le RFS modélisant les états est un RFS Independent Identically Distributed Cluster (IID cluster). Dans un RFS IID cluster, la cardinalité est modélisée par une densité de probabilité discrète $p(.)$ qui ne suit pas forcement une loi de Poisson et qui satisfait $N = \sum_{n=0}^{\infty} np(n) = \int \mathcal{V}(x)\mathrm{d}x$. De plus pour une cardinalité donnée, les éléments du RFS sont indépendants et identiquement distribués avec une densité de probabilité $\mathcal{V}(.)/N$. Où $\mathcal{V}(.)$ est la fonction d'intensité du RFS et N est le nombre d'éléments.

Contrairement au filtre PHD qui ne propage que la fonction d'intensité, le filtre CPHD propage la fonction d'intensité et la densité de probabilité de la cardinalité du RFS (le nombre de cibles). L'ajout de cette information supplémentaire permet d'assurer une meilleure stabilité dans l'estimation du nombre de cibles par rapport au filtre PHD [Vo et al., 2007]. Cependant, une complexité de calcul supplémentaire est ajoutée. Le CPHD ne modélise pas la naissance de cibles à partir de la division de cibles existantes. Cependant, comme le filtre PHD il intègre tous les autres paramètres.

Le filtre PHD peut être implémenté de deux manières différentes selon comment la densité a posteriori multi-cible est représentée : en utilisant des particules où en utilisant un mélange de Gaussiennes. Dans la littérature, le filtre CPHD ne possède qu'une implémentation à base d'un mélange de Gaussiennes.

4.1.6.2 Approximation de la densité multi-cible avec des particules

Cette implémentation consiste en l'approximation de la densité multi-cible a posteriori avec un ensemble de particules ce qui a donné naissance au Sequential Monte Carlo PHD ou Particle PHD [Vo et al., 2003, Sidenbladh, 2003, Vo et al., 2005]. Les implémentations à base du filtre à particules permettent de traiter des problèmes non-linéaires et non-Gaussiens. Cependant, les mêmes inconvénients des filtres à particules affectent ces implémentations. En effet, la complexité augmente avec le nombre de particules et la dimension du vecteur d'état. Une contrainte supplémentaire s'ajoute quand les états des cibles sont estimés à partir des particules représentant la densité a posteriori, car des méthodes de segmentation comme les algorithmes Expectation Maximation (complexité

$O(\tau T^2 \mathcal{M})$) ou K-means (complexité $O(\tau T \mathcal{M})$) sont utilisées [Clark and Bell, 2007]. τ est le nombre d'itérations de l'algorithme de segmentation, T le nombre de cibles et $\mathcal{M}$ le nombre de particules.

4.1.6.3 Approximation de la densité multi-cible avec un mélange de Gaussiennes

L'approximation de la densité multi-cible avec un mélange de Gaussiennes (équation (4.22)) s'applique pour les deux filtres PHD et CPHD. Les implémentations à base de mélanges de Gaussiennes concernent les systèmes linaires affectés par des bruits additifs et Gaussiens. Ces implémentations donnent naissance aux filtres Gaussian Mixture PHD (GMPHD) [Vo and Ma, 2006] et Gaussian Mixture CPHD (GMCPHD) [Vo et al., 2007]. Dans les filtres GMPHD et GMCPHD, chaque cible est représentée par une composante Gaussienne du mélange. La moyenne de la Gaussienne représente le vecteur d'état de la cible, la covariance représente la matrice de covariance de la cible et le poids permet de confirmer ou de terminer la piste de la cible. L'extraction des états des cibles à partir du mélange de Gaussiennes s'effectue par un seuillage sur le poids des Gaussiennes. Ce processus d'extraction d'états est plus simple comparé à celui utilisé dans les implémentations à base de particules.

$$GM = \sum_{i=1}^{J} \omega^i \mathcal{N}(\mathrm{x}; m^i, \mathrm{P}^i), \tag{4.22}$$

J représente le nombre de composantes Gaussiennes dans le mélange. ω^i,m^i et P^i représentent respectivement, le poids, la moyenne et la covariance de la composante Gaussienne i.

La dérivation des implémentations GMPHD et GMCPHD à partir des filtres PHD et CPHD est réalisée en faisant les hypothèses suivantes :

1. Le modèle d'état est linéaire et Gaussien
   — la densité de transition $f_{k|k-1}$ d'une cible de l'état x_{k-1} à l'instant $k-1$ vers l'état x_k à l'instant k est :

   $$f_{k|k-1}(\mathrm{x}_k \,|\, \mathrm{x}_{k-1}) = \mathcal{N}(\mathrm{x}_k; F_k\, \mathrm{x}_{k-1}, Q_k) \tag{4.23}$$

   Avec F_k et Q_k sont, respectivement, la matrice de transition et la matrice de covariance de bruit du modèle d'évolution.
   — la vraisemblance que l'observation z est générée par la cible ayant l'état x_k à l'instant k est :

   $$g_k(\mathrm{z} \,|\, \mathrm{x}_k) = \mathcal{N}(\mathrm{z}; H_k\, \mathrm{x}_k, R_k) \tag{4.24}$$

   Avec H_k et R_k sont, respectivement, la matrice d'observation et la matrice de covariance de bruit du modèle d'observation.
2. Les probabilités de survie et de détection de cibles sont indépendantes de leurs états, i.e. pour une cible ayant l'état x à l'instant k

— la probabilité de survie de la cible est :

$$p_{s,k}(x) = p_{s,k} \tag{4.25}$$

— la probabilité que le détecteur détecte la cible est :

$$p_{D,k}(x) = p_{D,k} \tag{4.26}$$

3. La naissance de nouvelles cibles est modélisée avec le mélange de Gaussiennes suivant :

$$\gamma_k(\mathrm{x}) = \sum_{i=1}^{J_{\gamma,k}} \omega_{\gamma,k}^{(i)} \mathcal{N}(\mathrm{x}; m_{\gamma,k}^{(i)}, \mathrm{P}_{\gamma,k}^{(i)}) \tag{4.27}$$

Même si l'utilisation de GMPHD et de GMCPHD est limitée aux systèmes linéaires et Gaussiens, leur complexité dépend du nombre Gaussiennes composant le mélange et du nombre d'observations fournies par le détecteur. Ainsi, les ressources nécessaires pour faire fonctionner ces filtres sont déterministes et ne dépendent pas du scénario de suivi. Cette complexité de calcul les rend attractives quand des architectures à base de processeur embarqué sont visées.

4.2 Bilan des méthodes de suivi multi-cible

Dans cette section, les méthodes décrites dans la section 4.1.1 sont analysées en se basant sur les résultats de la littérature du point de vue de la complexité de calcul, des besoins en mémoire et de la qualité de suivi.

Dans [Blackman and Popoli, 1999], les performances de GNN, PDA qui est une version de JPDA pour une seule cible et MHT sont comparées avec trois scénarios. Les auteurs considérent une seule cible manœuvrante dans les scénarios 1 et 2, et une seule cible non manœuvrante dans le scénario 3. La probabilité de détection est fixée à 1 dans le premier scénario et à 0.75 dans les deux autres. En faisant varier la densité des fausses alarmes, le pourcentage du maintien de la piste de la cible est mesuré. Les résultats obtenus montrent que la méthode MHT surpasse GNN et PDA. En effet, le MHT permet d'obtenir les mêmes résultats que GNN quand la densité de fausses alarmes est 10 à 100 fois plus élevée. PDA fournit les mêmes résultats que GNN pour 3 fois plus de fausses alarmes. Les auteurs ont commenté que la méthode MHT nécessite plus de temps de développement comparée aux méthodes GNN et PDA. De plus, sa complexité dépasse largement celui des deux méthodes.

Dans [Vo and Ma, 2006], JPDA et GMPHD ont été comparés pour un nombre fixe de cibles égal à 2. Le nombre de cibles connu donne un avantage au JPDA, car il l'intègre directement dans ses formules. Même avec cet avantage, la différence de performance sur la maintien des pistes est petite (inférieure à 0.1 au sens de la métrique Circular Position Error Probability (CPEP) [Vo and Ma, 2006]) pour la plus grande densité de fausses alarme utilisée ($10^{-4}\mathrm{m}^{-2}$). Comme l'écart de performance est proportionnel à la densité de fausses alarmes, la diminution de la densité réduit l'écart.

Dans [Panta et al., 2004], une comparaison entre un MHT orienté piste et l'implémentation SMC-PHD est effectuée. Les résultats obtenus sur des simulations montrent que le filtre SMC-PHD fourni un meilleur suivi que MHT. De même, dans [Lamard et al., 2012a] une comparaison entre un MHT orienté piste et le filtre GMCPHD a été réalisée pour un contexte d'aide à la conduite en utilisant des données réelles. Le nombre de cibles atteint un maximum de 4 cibles. Les résultats obtenus montrent que l'estimation de la position est similaire pour les deux méthodes, l'estimation du nombre de cibles est meilleur avec GMCPHD et que l'exécution de GMCPHD est 9 fois plus lente que celle du MHT utilisé, avec 0.0046 s pour une itération MHT et 0.042 s pour une itération de GMCPHD. Les spécifications de la machine utilisée pour l'exécution des algorithmes ne sont pas fournie par les auteurs.

Les résultats obtenus dans [Vo et al., 2007] montrent que la variance sur le nombre estimé de cibles est 12.5 fois plus petite dans GMCPHD comparé au GMPHD. Cependant, GMPHD est plus réactif au changement du nombre de cibles, ce qui lui permet de confirmer ou de supprimer une cible après seulement quelques itérations. Cet effet est avantageux pour l'initialisation de nouvelles cibles, mais, son inconvénient est l'instabilité du nombre de cibles quand les cibles ne sont pas détectées, d'où la différence flagrante dans la variance sur le nombre de cibles estimé entre GMPHD et GMCPHD.

Les auteurs ont aussi comparé les performances de GMCPHD par rapport au JPDA pour un nombre fixe de cibles égal à 4. Les résultats obtenus montrent que, pour le scénario étudié, le GMCPHD surpasse le filtre JPDA quand la densité de fausses alarmes est élevée ($2 \times 10^{-5} m^{-2}$) et quand la probabilité de détection est faible (0.7). Pour un fenêtrage statistique sur une région dont la probabilité que l'observation soit incluse dans cette région est de 0.99, le temps d'exécution du JPDA est de même ordre de grandeur que celui du GMCPHD.

Le temps d'exécution pour une itération du filtre Particle PHD [Sidenbladh, 2003] est 1.58 secondes, composé de 0.38 secondes pour la prédiction et la correction et 1.2 secondes pour l'extraction des états des cibles à partir des particules modélisant la densité de multi-cible a posteriori. Ce résultat est obtenu avec une implémentation en Matlab sur un PC de bureau dont les spécifications ne sont pas fournies. Le test est réalisé par simulation sur un scénario de suivi de 3 voitures : le nombre maximum d'objets est fixé à 5, chaque objet est représenté avec 1000 particules, les vecteurs d'état et d'observation contiennent 4 éléments qui sont les coordonnées de position, l'amplitude de la vitesse et la direction de mouvement. L'estimation des états des cibles est déterminée en trouvant le mélange de Gaussiennes le plus proche de la densité multi-cible a posteriori. Ce temps de calcul obtenu pour un scénario simple de suivi montre bien la complexité calculatoire de la méthode à base de particules.

Dans [Vo et al., 2007], la vitesse d'exécution de GMPHD est 2.8 à 4 fois plus rapide que celle de GMCPHD avec 0.027 s à 0.046 s par itération pour GMPHD, pour des implémentations Matlab sur un notebook standard dont les spécifications ne sont pas fournies. Dans les scénarios de simulation, le nombre de cibles varie de 1 à 10. Le nombre de composantes de Gaussiennes dans le mélange est fixé à 100, ce qui permet de suivre jusqu'à 100 cibles au maximum.

4.2.1 Choix de l'algorithme de suivi

En se basant sur ces comparaisons, nous éliminons la méthode GNN car elle présente une mauvaise robustesse aux fausses alarmes et aux non-détections comparée aux autres méthodes, ce qui se produit souvent dans le suivi visuel des objets. De plus, une logique supplémentaire est nécessaire pour la gestion de l'initialisation et de la terminaisons des pistes.

La performance du JPDA surpasse mais reste proche de celle de GMPHD pour un scénario de suivi dont le nombre de cibles est connu a priori. Dans un scénario quelconque, où le nombre de cibles varie en fonction de temps, cette vérité peut changer selon la logique utilisée pour la gestion des initialisations et des terminaisons de pistes. Le nombre d'hypothèses générées dans JPDA dépend du scénario de suivi et plus précisément du nombre de pistes se partageant des observations et du nombre d'observations partagées entre pistes. Ainsi, le pire cas doit être considéré afin d'éviter des problèmes d'implémentations, notamment les fuites de mémoire. Comme la génération d'hypothèses dans JPDA est un processus combinatoire, le nombre d'hypothèses pour le pire cas est élevé. Ce nombre d'hypothèses est donné par l'équation (4.28) [Konstantinova and Alexiev, 2001].

$$\mathrm{Nb}_{\mathrm{hypothese}} = \sum_{i=0}^{\min(N,M)} \frac{M!}{i!(M-i)!} \frac{N!}{(N-i)!} \tag{4.28}$$

Où M est le nombre d'observations partagées entre N pistes. Par exemple, pour $M = 10$ et $N = 5$, le nombre d'hypothèses est de 63591, ce qui nécessite une grande quantité de mémoire et de calcul. JPDA est éliminé à cause du critère combinatoire.

L'implémentation du filtre PHD en utilisant des particules présente des résultats de suivi surpassant ceux de MHT mais le temps d'exécution n'est pas prometteur pour un fonctionnement en temps réel sur un processeur embarqué.

GMPHD, GMCPHD et MHT traitent tous les problèmes de suivi multi-cible en intégrant l'initialisation, la terminaison et le maintien des pistes, et assurent une bonne qualité de suivi. Les résultats rapportés par la littérature montrent que l'exécution de GMCPHD est plus lente que MHT et GMPHD. Même si le temps d'exécution du MHT orienté piste rapporté dans [Lamard et al., 2012a] est petit, le temps d'exécution et la mémoire nécessaires pour le fonctionnement du MHT dépendent du scénario de suivi et des paramètres de l'implémentation. Ce temps d'exécution n'est pas garanti pour d'autres scénarios.

Par élimination, le filtre GMPHD s'avère celui permettant le bon compromis entre la complexité de calcul et la qualité de suivi. En effet, il assure des performances proches de celle de GMCPHD et possède un temps d'exécution intéressent sur un notebook standard. De plus, grâce à sa complexité qui ne dépend que du nombre de Gaussiennes composant le mélange et du nombre d'observations, le filtre GMPHD est un bon candidat pour un fonctionnement en temps réel sur un processeur embarqué. Dans la prochaine section, l'algorithme GMPHD est détaillé.

4.3 Gaussian Mixture Probability Hypothesis Density

Le filtre GMPHD considère que la naissance d'une nouvelle cible peut être causée par la division d'une cible existante ou par une cible qui s'introduit pour la première fois dans la région de surveillance (naissance spontanée). Pour les deux naissances, des informations a priori sont requises. La naissance à partir de la division de cibles nécessite la définition d'un mélange de Gaussiennes modélisant la probabilité de division d'une cible ainsi que l'évolution de son état après division. Pour la naissance spontanée, un mélange de Gaussiennes modélisant la probabilité de naissance de cibles sur des régions de l'espace d'état est utilisé. Nous ne nous intéressons qu'aux naissances spontanées, car la division de cibles ne concerne que certains types d'objets, par exemple, les missiles dans le domaine militaire. En effet, ce n'est pas notre cas, vu que les objets considérés sont des personnes.

Les étapes de prédiction et de correction du filtre GMPHD sont comme suit :

Prédiction : supposons que la densité multi-cible $\mathcal{V}_{k-1|k-1}(.)$ à l'instant $k-1$ est le mélange de $J_{k-1|k-1}$ Gaussiennes défini par l'équation (4.29).

$$\mathcal{V}_{k-1|k-1}(\mathrm{x}_{k-1}) = \sum_{i=1}^{J_{k-1|k-1}} \omega_{k-1|k-1}^{(i)} \mathcal{N}(\mathrm{x}_{k-1}; m_{k-1|k-1}^{(i)}, \mathrm{P}_{k-1|k-1}^{(i)}) \tag{4.29}$$

La densité multi-cible $\mathcal{V}_{k|k-1}(.)$ qui résulte de la prédiction est le mélange de $J_{k-1|k-1}+J_{\gamma,k}$ Gaussiennes défini par l'équation (4.30). $J_{\gamma,k}$ est le nombre de composantes Gaussiennes formant le mélange qui modélise la naissance de nouvelles cibles.

$$\mathcal{V}_{k|k-1}(\mathrm{x}_{k|k-1}) = \sum_{i=1}^{J_{\gamma,k}} \omega_{\gamma,k}^{(i)} \mathcal{N}(\mathrm{x}_{k|k-1}; m_{\gamma,k}^{(i)}, \mathrm{P}_{\gamma,k}^{(i)}) + \sum_{i=1}^{J_{k-1|k-1}} \omega_{p}^{(i)} \mathcal{N}(\mathrm{x}_{k|k-1}; m_{p}^{(i)}, \mathrm{P}_{p}^{(i)}) \tag{4.30}$$

Où $\sum_{i=1}^{J_{\gamma,k}} \omega_{\gamma,k}^{(i)} \mathcal{N}(\mathrm{x}_{k|k-1}; m_{\gamma,k}^{(i)}, \mathrm{P}_{\gamma,k}^{(i)})$ représente le mélange de Gaussiennes qui prédit la naissance de nouvelles cibles. $\sum_{i=1}^{J_{k-1|k-1}} \omega_{p}^{(i)} \mathcal{N}(\mathrm{x}_{k|k-1}; m_{p}^{(i)}, \mathrm{P}_{p}^{(i)})$ est le mélange de Gaussiennes résultant de l'évolution du mélange de Gaussiennes $\mathcal{V}_{k-1|k-1}(.)$ existant à l'instant $k-1$. Les termes composant ce dernier sont donnés par les équations suivantes :

$$\omega_{p}^{(i)} = p_{s,k} \omega_{k-1|k-1}^{(i)} \tag{4.31}$$

$$m_{p}^{(i)} = F_k m_{k-1|k-1}^{(i)} \tag{4.32}$$

$$\mathrm{P}_{p}^{(i)} = Q_k + F_k \, \mathrm{P}_{k-1|k-1}^{(i)} \, F_k^T \tag{4.33}$$

Correction : supposons que la densité multi-cible après prédiction $\mathcal{V}_{k|k-1}(.)$ est le mélange de $J_{k|k-1} = J_{k-1|k-1} + J_{\gamma,k}$ Gaussiennes défini par l'équation (4.34).

$$\mathcal{V}_{k|k-1}(\mathrm{x}_{k|k-1}) = \sum_{i=1}^{J_{k|k-1}} \omega_{k|k-1}^{(i)} \mathcal{N}(\mathrm{x}_{k|k-1}; m_{k|k-1}^{(i)}, \mathrm{P}_{k|k-1}^{(i)}) \tag{4.34}$$

La densité multi-cible a posteriori $\mathcal{V}_{k|k}(.)$ qui résulte de la correction est le mélange de $J_{k|k-1}(M_k + 1)$ Gaussiennes défini par l'équation (4.35). M_k est le nombre d'observations fournies par le détecteur à l'instant k.

$$\mathcal{V}_{k|k}(\mathrm{x}_k) = \sum_{i=1}^{J_{k|k-1}} \omega_{cnd}^{(i)} \mathcal{N}(\mathrm{x}_k; m_{k|k-1}^{(i)}, \mathrm{P}_{k|k-1}^{(i)}) + \sum_{j=1}^{M_k} \sum_{i=1}^{J_{k|k-1}} \omega_{cd}(z_j)^{(i)} \mathcal{N}(\mathrm{x}_k; m_{cd}(z_j)^{(i)}, \mathrm{P}_{cd}^{(i)}) \tag{4.35}$$

Où $\sum_{i=1}^{J_{k|k-1}} \omega_{cnd}^{(i)} \mathcal{N}(\mathrm{x}_k; m_{k|k-1}^{(i)}, \mathrm{P}_{k|k-1}^{(i)})$ représente le mélange de Gaussiennes résultant de la mise à jour de $\mathcal{V}_{k|k-1}(.)$ en considérant les non détections. $\sum_{j=1}^{M_k} \sum_{i=1}^{J_{k|k-1}} \omega_{cd}(z_j)^{(i)} \mathcal{N}(\mathrm{x}_k; m_{cd}(z_j)^{(i)}, \mathrm{P}_{cd}^{(i)})$ est le mélange de $J_{k|k-1} M_k$ Gaussiennes résultant de la mise à jours de $\mathcal{V}_{k|k-1}(.)$ en considérant les observations fournies par le détecteur. Les différents termes intervenant dans les deux mélanges de Gaussiennes sont donnés par les équations suivantes :

$$\omega_{cnd}^{(i)} = (1 - p_{D,k}) \omega_{k|k-1}^{(i)} \tag{4.36}$$

$$\omega_{cd}(z)^{(i)} = \frac{1}{\sqrt{(2\pi)^n |S^{(i)}|}} \times \frac{p_{D,k} \omega_{k|k-1}^{(i)} e^{-0.5(z-m^{(i)})^T {S^{(i)}}^{-1} (z-m^{(i)})}}{\kappa_k(z) + L(z)} \tag{4.37}$$

$$m^{(i)} = H_k m_{k|k-1}^{(i)} \tag{4.38}$$

$$S^{(i)} = R_k + H_k \, \mathrm{P}_{k|k-1}^{(i)} \, H_k^T \tag{4.39}$$

$$L(z) = \sum_{j=1}^{J_{k|k-1}} \frac{p_{D,k} \omega_{k|k-1}^{(j)}}{\sqrt{(2\pi)^n |S^{(j)}|}} e^{-0.5(z-m^{(j)})^T {S^{(j)}}^{-1} (z-m^{(j)})} \tag{4.40}$$

$$m_{cd}(z)^{(i)} = m_{k|k-1}^{(i)} + K_k^{(i)} (z - m^{(i)}) \tag{4.41}$$

$$\mathrm{P}_{cd}^{(i)} = [I - K_k^{(i)} H_k] \, \mathrm{P}_{k|k-1}^{(i)} \tag{4.42}$$

$$K_k^{(i)} = \mathrm{P}_{k|k-1}^{(i)} \, H_k^T {S^{(i)}}^{-1} \tag{4.43}$$

$\kappa_k(.)$ est l'intensité du RFS de fausses alarmes à l'instant k. n dans ce cas est la dimension du vecteur d'observation.

Afin d'extraire les pistes des cibles, les composantes Gaussiennes du mélange ayant un poids supérieur à un seuil prédéfini (0.5) sont sélectionnées. Les moyennes des Gaussiennes représentent les vecteurs d'état des cibles. Les cibles correspondant à ces composantes sont des cibles confirmées.

Dans le filtre GMPHD de base, les auteurs n'ont pas introduit l'affectation des identités aux objets suivis. L'intégration des identités a été effectuée pour le GMPHD dans [Clark et al., 2006a]. L'idée est d'associer à chaque composante du mélange de Gaussiennes une identité et de propager cette identité à travers la phase de prédiction et de correction du filtre. À chaque instant, chaque composante du mélange de Gaussiennes modélisant la naissance spontanée prend une nouvelle identité. Durant la prédiction chaque composante garde sa propre identité. Durant la correction, toutes les composantes générées en associant des observations à une composante résultante de la prédiction, garde la même identité de cette composante.

Les équations de prédiction et de corrections du filtre GMPHD montrent que le nombre de composantes dans le mélange de Gaussiennes augmente à chaque itération de manière que $J_{k|k} = (|Z_k| + 1)(J_{k-1|k-1} + J_{\gamma,k})$. La limitation du nombre de composantes est donc nécessaire pour éviter l'explosion de la complexité. [Vo and Ma, 2006] ont développé la méthode décrite dans l'algorithme 2, permettant de fusionner les composantes qui sont proches et d'éliminer les composantes avec des faibles poids.

entrée: $\{\omega_{k|k}^{(i)}, m_{k|k}^{(i)}, \mathrm{P}_{k|k}^{(i)}\}_{i=1}^{J_{k|k}}$: mélange de Gaussiennes, TH : seuil de suppression, U : seuil de fusion
sortie : $\{\tilde{\omega}_{k|k}^{(i)}, \tilde{m}_{k|k}^{(i)}, \tilde{\mathrm{P}}_{k|k}^{(i)}\}_{i=1}^{l}$: mélange de Gaussiennes
initialisation :
  $l = 0$
  $I = \{i = 1, \ldots, J_{k|k} \mid \omega_{k|k}^{(i)} > \mathrm{TH}\}$
répéter
  l=l+1
  $j = \underset{i \in I}{\operatorname{argmax}}\ \omega_{k|k}^{(i)}$
  $L = \{i \in I \mid (m_{k|k}^{(i)} - m_{k|k}^{(j)})^T (\mathrm{P}_{k|k}^{(i)})^{-1} (m_{k|k}^{(i)} - m_{k|k}^{(j)}) < \mathrm{U}\}$
  $\tilde{\omega}_{k|k}^{(l)} = \sum_{i \in L} \omega_{k|k}^{(i)}$
  $\tilde{m}_{k|k}^{(l)} = \frac{1}{\tilde{\omega}_{k|k}^{(l)}} \sum_{i \in L} \omega_{k|k}^{(i)} m_{k|k}^{(i)}$
  $\tilde{P}_{k|k}^{(l)} = \frac{1}{\tilde{\omega}_{k|k}^{(l)}} \sum_{i \in L} \omega_{k|k}^{(i)} (\mathrm{P}_{k|k}^{(i)} + (\tilde{m}_{k|k}^{(l)} - m_{k|k}^{(i)})(\tilde{m}_{k|k}^{(l)} - m_{k|k}^{(i)})^T)$
  $L = L \backslash i$
jusqu'à $I = \emptyset$;
si $l > J_{max}$ **alors**
  Remplace $\{\tilde{\omega}_{k|k}^{(i)}, \tilde{m}_{k|k}^{(i)}, \tilde{\mathrm{P}}_{k|k}^{(i)}\}_{i=1}^{l}$ par les J_{max} composantes avec les plus forts poids
fin

Algorithm 2: Fusion et suppression des composantes Gaussiennes [Vo and Ma, 2006]

La méthode de fusion et de suppression des composantes du mélange ne gère pas automatiquement les identités associées aux Gaussiennes. Dans [Clark et al., 2006a], lorsque plusieurs composantes sont fusionnées, l'identité assignée à la composante résultante est l'identité de la composante avec le plus fort poids. Afin d'assurer la cohérence des identités, deux composantes du mélange ne peuvent pas avoir une même identité. Ainsi, pour chaque identité, la composante avec le plus fort poids garde sa propre identité et toutes les autres composantes reçoivent des nouvelles identités.

Deux questions concernant le filtre GMPHD se posent :

1. Quelle est la complexité théorique (nombre d'opérations et besoins en mémoire) et la complexité réelle (temps d'exécution et mémoire requise pour une exécution sur un processeur embarqué) du filtre GMPHD ?
2. Comment se comporte le filtre GMPHD dans un système de suivi visuel d'objets ?

L'analyse théorique de la complexité de GMPHD est effectuée dans la section 4.4. La mesure de la complexité est effectuée pour le filtre GMPHD original ainsi que pour deux versions dégradées du filtre dans la section 5.4. L'analyse du comportement du filtre dans un système de vision est présentée dans la section 4.6.

4.4 Analyse détaillée de la Complexité du filtre GMPHD

En se basant sur les formules du filtre GMPHD, nous analysons le type et le nombre d'opérations utilisés ainsi que la quantité mémoire requise. L'étape de prédiction donnée par l'équation (4.30) nécessite les équations (4.31) à (4.33). La correction (équation (4.35)) est basée sur les équations (4.36), (4.37), (4.41) et (4.42). Chaque terme utilisé dans les formules est calculé une seule fois, par exemple, dans l'équation (4.40), seulement la somme est comptabilisée car les termes $\frac{p_{D,k}\omega_{k|k-1}^{(j)}}{\sqrt{(2\pi)^n|S^{(j)}|}}e^{-0.5(z-m^{(j)})^T S^{(j)^{-1}}(z-m^{(j)})}$ pour $i = 1,\ldots,J_{K|K-1}$ sont déjà calculés dans l'équation (4.37). Le tableau 4.1 indique le type et le nombre d'opérations pour les équations intervenant dans la prédiction et la correction, pour une seule itération.

Le tableau 4.1 n'inclut pas les opérations de la méthode de suppression et de fusion des composantes du mélange de Gaussiennes permettant la régularisation du nombre de composantes. Supprimer les composantes ayant un faible poids revient à parcourir le vecteur de poids et de comparer chaque élément au seuil de suppression, ce qui implique $J_{k|k} = (J_{max} + J_\gamma)(M+1)$comparaisons.

Pour la fusion, la première étape est de déterminer la composante correspondant au poids le plus élevé. En considérant que le nombre d'opérations nécessaires pour la localisation de l'élément max sur un vecteur de taille n est $N_{op}(n)^{max}$, alors le nombre d'opérations de cette étape, dans le pire cas, est égal à $[J_{k|k}(\sum_{i=0}^{i=J_{k|k}-1} N_{op}(J_{k|k}-i)]$, avec $J_{k|k} = (J_{max}+J_\gamma)(M+1)$.

La deuxième étape consiste à calculer pour la composante sélectionnée sa distance de Mahalanobis par rapport aux autres composantes et de comparer cette distance au

TABLE 4.1 – Nombre et type des opérations utilisées dans le filtre GMPHD.

	Éq (4.31) / Éq (4.32)	Éq (4.33) / Éq (4.36)	Éq (4.37) / Éq (4.41)	Éq (4.42)
ADD	0	$2n^2 J_{k-1}$ $(n-1)$	$J_{k\|k-1}(Mm^2+mn^2+nm^2-m)$	$J_{k\|k-1}n^2$ $(n+m-2)$
	$J_{k-1}n$ $(n-1)$	0	$J_{k\|k-1}(mn^2+nm^2-2nm+2Mnm)$	
SUB	0	0	$MJ_{k\|k-1}m$	$J_{k\|k-1}n^2$
	0	1	$J_{k\|k-1}Mm$	
MUL	J_{k-1}	$2J_{k-1}n^3$	$J_{k\|k-1}(n+2+mn+mn^2+nm^2)$ $+J_{k\|k-1}M(m^2+m+2)$	$J_{k\|k-1}n^2$ $(n+m)$
	$J_{k-1}n^2$	J_{k-1}	$J_{k\|k-1}(mn^2+nm^2+Mnm)$	
DIV	0	0	$J_{k\|k-1}+M$	0
	0	0	0	
EXP	0	0	$MJ_{k\|k-1}$	0
	0	0	0	
SQRT	0	0	$J_{k\|k-1}$	0
	0	0	0	
MINV	0	0	$J_{k\|k-1}$	0
	0	0	0	
DET	0	0	$J_{k\|k-1}$	0
	0	0	0	

ADD=addition, SUB=soustraction, MUL=multiplication, DIV=division, EXP=exponentielle, SQRT=racine-carrée, MINV= inversion de matrice de taille $m \times m$, DET= le déterminant d'une matrice de taille $m \times m$, M : le nombre d'observations à l'instant k, n : la dimension du vecteur d'état et m : la dimension du vecteur d'observation.

seuil de fusion. Dans le pire cas, quand aucune fusion n'est effectuée, le nombre d'opérations nécessaires est $J^2_{k|k}(N_{op}^{Mahalanobis} + 1\text{comparaison})$, où $J_{k|k} = (J_{max} + J_\gamma)(M+1)$ et $N_{op}^{Mahalanobis}$ est le nombre d'opérations pour le calcul de la distance de Mahalanobis qui est égal à (n^3+n^2)multiplications+(n^2-1)additions+une inversion de matrice de taille $n \times n$. n étant la taille du vecteur d'état. L'étape dont le nombre d'opérations n'est pas déterministe est l'étape de fusion qui dépend du nombre de composantes fusionnées. Le nombre d'opérations pour la fusion de l composantes est : $l-1$additions pour la fusion des poids, $n$divisions+lnmultiplications+$n(l-1)$additions pour la fusion des vecteurs d'état et n^2divisions+$2n^2l$multiplications+$(2l-1)n^2l$additions+lnsoustractions.

La dernière étape de GMPHD est l'estimation des états des cibles à partir du mélange de Gaussiennes. Seules les composantes ayant un seuil supérieur au seuil de confirmation

de cibles sont sélectionnées. La réalisation de cette étape nécessite J_{max}comparaisons. .

Concernant la mémoire, le GMPHD nécessite la mémorisation du mélange de Gaussiennes, des matrices du modèle d'état, des observations et quelques variables intermédiaires. Pour chaque composante du mélange de Gaussiennes, il faut sauvegarder deux scalaires (un pour le poids et un pour l'identité), un vecteur d'état de taille n et une matrice de covariance de taille $n \times n$. Le nombre de composantes du mélange de Gaussiennes est $(J_{max} + J_\gamma)(M_{max} + 1)$ avec M_{max} est le nombre maximum d'observations que le détecteur est susceptible de fournir. Le modèle d'état nécessite à mémoriser la matrice de transition de taille $n \times n$, la matrice d'observation de taille $m \times m$ ainsi que les deux matrices de covariance des bruits de transition (taille $n \times n$) et d'observation (taille $m \times m$). La taille requise par les observations fournies par le détecteur est $M \times m$. Des variables intermédiaires sont nécessaires pour la manipulation des résultats intermédiaires : les poids des composantes du mélange après prédiction nécessitent un vecteur de taille $J_{max} + J_\gamma$. Dans l'étape de correction, les variables intermédiaires incluent les matrices intermédiaires (matrice d'innovation, gain de Kalman, *etc.*), ce qui correspond au stockage de $n^2 + 2m^2 + nm + 2m$ éléments.

Pour une configuration du filtre GMPHD où $n = 4$, $m = 2$, $M = 100$, $J_{max} = 10$ et $J_\gamma = 4$, le nombre d'opérations et la capacité mémoire requise pour les phases de prédiction et de correction sont indiqués dans le tableau 4.2. Avec ces paramètres, GMPHD peut suivre un nombre de cibles inférieur à 10 tout en initialisant au maximum 4 nouvelles pistes par itération.

TABLE 4.2 – Nombre d'opérations total résultant du scénario de suivi pour les paramètres de GMPHD suivant : $n = 4$, $m = 2$, $M = 100$, $J_{max} = 10$, $J_{\gamma,max} = 4$.

ADD	SUB	MUL	DIV	EXP	SQRT	MINV	DET	Nombre d'éléments en mémoire
31068	5825	26744	114	1400	14	14	14	29988

À titre d'exemple, supposons que les opérations du tableau 4.2 sont représentables en fonction d'une opération de base du processeur OP_{base} comme suit : ADD=OP_{base}, SUB=OP_{base}, MUL=OP_{base}, DIV=20OP_{base}, EXP=50OP_{base}, SQRT=30OP_{base}. La valeur de l'équivalence pour EXP, SQRT et DIV sont extraites de [Ueberhuber, 1997]. Pour MINV et DET, le nombre d'opérations de base équivalentes dépend de la méthode utilisée. Pour une matrice de 2×2, MINV et DET peuvent être déterminées avec un calcul analytique. Dans ce cas, l'équivalent en opérations de base est MINV=85OP_{base} et DET=3OP_{base}. Cependant, dés que la taille de la matrice est supérieure à 2×2, des méthodes numériques sont nécessaires pour le calcul du déterminant et de l'inverse de la matrice. l'équivalent en opérations de base dans ce cas peut exploser rapidement. En

appliquant cette équivalence, le nombre total d'opérations est égal à 137569. Pour un processeur embarqué type ARM Cortex A-9 fonctionnant à 1 GHz la puissance de traitement est 1 Gega opérations flottantes par seconde (GFLOPS) par coeur sans le coprocesseur NEON [2]. Les étapes de prédiction et de correction de GMPHD pour cette configuration du filtre s'exécutent avec une vitesse de 7000 itérations par seconde sur ce type de processeur. Le processeur de la Raspberry Pi version 1 (processeur graphique (GPU) non inclue) possède une puissance de traitement de 0.04 GFLOPS [3], ce qui signifie qu'une vitesse de 290 itérations par seconde peut être atteinte. Cette estimation concerne que les opérations arithmétiques. Cependant, la vitesse d'exécution dépend aussi des accès mémoire pour la lecture et l'écriture des variables. L'estimation du coût des accès mémoire est difficilement réalisable, car il dépend de l'implémentation du filtre et de l'architecture mémoire de la carte de traitement comme la taille de la mémoire cache, le nombre de caches, *etc.*

En supposant que chaque élément est un flottant codé sur 32 bits, la quantité de mémoire requise par les étapes de prédiction et de correction est inférieure à 120 ko pour ce scénario.

Les estimations de la complexité de calcul et des besoins en mémoire des étapes de prédiction et de correction de GMPHD démontrent la faisabilité de suivi sur un processeur embarqué. Cependant, comment réagira le filtre GMPHD si l'architecture est encore plus contrainte : absence de FPU ou une FPU peu optimisée qui nécessite plusieurs cycles d'horloge pour réaliser une opération, très peu de mémoire (<64 ko), faible fréquence d'horloge (<200 MHz), *etc* ? Les processeurs présentant ces caractéristiques permettent une très faible consommation (<500 mW) et une taille compacte. Un exemple de ces processeurs est celui la caméra intelligente CMUcam3 présentée dans le tableau 1.1. Le processeur de CMUcam3 fonctionne à 60 MHz et la mémoire de données est de 64 ko. Dans la prochaine section, nous effectuerons certains approximations et nous analysons les effets de ces approximations sur la qualité de suivi et les gains en vitesse d'exécution et en mémoire.

4.5 Simplification du filtre GMPHD pour le portage sur un calculateur embarqué

4.5.1 Simplifications

La simplification du filtre consiste à approximer et/ou à éviter les opérations nécessitant plusieurs opérations de base : EXP, SQRT, MINV et DET. MINV et DET sont considérés car dans le cas général, la taille des matrices peut dépasser 2×2 ce qui engendre un nombre d'opérations de base élevé à cause de l'utilisation des méthodes numériques.

La simplification peut s'effectuer en faisant des approximations sur des étapes de l'algorithme afin de diminuer le nombre de ces opérations ou en utilisant des approximations

2. http ://www.anandtech.com/show/6971/exploring-the-floating-point-performance-of-modern-arm-processors. Vérifié le 07/2015
3. http ://elinux.org/RPi_Performance. Vérifié le 07/2015

mathématiques pour les implémentations des opérations. Les approximations sont effectuées en supposant que le vecteur d'état est composé de quatre éléments : coordonnées de position (x, y) et coordonnées de vitesse (v_s, v_y) et un vecteur d'observation à deux éléments : coordonnées de position (x, y). De plus, nous supposons que la coordonnée x subit les mêmes effets que la coordonnée y dans le modèle d'état, i.e. les covariances initiales, les bruits du modèle de transition ainsi que les bruits du modèle d'observation sont identiques pour x et y. Nous nous basons sur ce scénario de suivi, car c'est le plus courant dans la littérature [Vo and Ma, 2006, Clark et al., 2006a, Clark et al., 2006b, Wang et al., 2007].

La première approximation consiste à représenter les matrices de covariances des composantes du mélange de Gaussiennes ainsi que les matrices de covariance d'innovation S utilisées dans l'étape de correction, par des matrices diagonales. Pour un vecteur d'état de taille n et pour un vecteur d'observation de taille m, cette approximation permet de convertir l'inversion d'une matrice $m \times m$ en m divisions (dont le coût en opérations de base est plus petit), le calcul de déterminant en m multiplications et de réduire le nombre d'éléments à mémoriser de n^2 à n pour chaque matrice de covariance. Cependant, les matrices de covariance résultant de l'étape de prédiction ne sont pas approximées par des matrices diagonales, dans le but de bénéficier de l'information de corrélation entre la position et la vitesse des cibles lorsque la matrice diagonale de covariance d'innovation S est calculée. L'utilisation de l'hypothèse que x et y subissent les mêmes effets à travers le modèle d'état permet d'éviter le calcul de la racine carrée de l'équation (4.37). Comme S est diagonale et possède $m = 2$ éléments qui sont identiques à cause de l'hypothèse, le déterminant $|S|$ de la matrice diagonale $S = [S_1, S_2]$ est $|S| = S_1 S_2 = S_1^2$. Ainsi, le carré s'élimine avec la racine carrée, ce qui permet de remplacer $\sqrt{|S|}$ par S_1. Cette approximation est possible que pour ce scénario où $m = 2$.

L'équivalence en opérations de base de l'exponentielle est assez élevé car les méthodes de calcul mathématiques utilisées pour le calcul de l'exponentielle, comme le Cordic [Walther, 1971] ou en approximation de Taylor, sont des méthodes itératives qui nécessitent plusieurs opérations pour atteindre la convergence. Dans notre cas, e^{-z} est approximée avec plusieurs segments de droites, en prenant en compte que z, qui représente la distance de Mahalanobis, est toujours positif. L'équation (4.44) présente un exemple d'approximation de e^{-z} avec deux segments de droite. Avec cette approximation, le calcul de l'exponentielle, dans ce cas, se traduit par deux comparaisons, une multiplication, une addition et quelques variables prédéfinies en mémoire.

$$e^{-z} = \begin{cases} a_1 z + b_1 & \text{if } 0 \leq z \leq v_1 \\ a_2 z + b_2 & \text{if } v_1 < z \leq v_2 \\ 0 & \text{if } z > v_2 \end{cases} \tag{4.44}$$

La méthode permettant la fusion et la suppression des composantes du mélange de Gaussiennes, est une méthode dont la complexité n'est pas négligeable. En effet, elle nécessite $J_{k|k}^2$ calcul de distances de Mahalanobis, plusieurs divisions, multiplications et additions, pour effectuer la fusion des composantes. Afin de limiter le nombre de composantes sans

avoir à effectuer tous ces calculs, aucune fusion de composantes n'est réalisée. Après la phase de correction, seules les J_{max} composantes ayant les poids les plus élevés sont gardées. Toutes les autres composantes sont supprimées.

La dernière approximation est l'implémentation du filtre en utilisant des variables entières dont la dynamique est inférieure ou égale à 32 bits (implémentation en virgule fixe au lieu de virgule flottante). Cette étape est dépendante de la dynamique des données d'entrée : observations, taille de la région de surveillance, bruit de transition, bruit d'état, probabilité de détection et probabilité de survie. Ainsi, l'implémentation réalisée n'est vérifiée que pour le scénario de suivi étudié. Le changement des données d'entrée ou les paramètres du filtre peut nécessiter de re-calculer les dynamiques des variables. Cependant une manière plus facile de faire consiste à changer la taille de la nouvelle région de surveillance pour qu'elle concorde ou soit incluse dans la région de surveillance du scénario étudié ici. Cela peut s'effectuer en changeant les unités, par exemple passer du centimètre au mètre. Concernant les paramètres du filtre, la dynamique des variables reste stable si les paramètres restent dans des intervalles prédéfinis.

4.5.2 Résultats

L'objectif est d'évaluer le filtre GMPHD sur un processeur embarqué de type ARM et d'estimer le gain en temps d'exécution et en mémoire ainsi que la dégradation de la qualité de suivi causé par les approximations effectuées. Nous utilisons trois versions : GMPHD pour le filtre original, FL_GMPHD qui représente la version approximée codée en virgule flottante (simple précision) et FX_GMPHD qui représente la version approximée codée en virgule fixe (toutes les variables sont entières).

4.5.2.1 Scénario de suivi

Nous considérons un suivi purement simulé dans un plan $x-y$ où le nombre de cibles est inconnu et varie au fil de temps. La région de surveillance est telle que $x \in [-500, 1000]$ et $y \in [-1000, 500]$. 6 cibles sont simulées dont les trajectoires sont fournies par la figure 4.5. Le nombre de cibles peut atteindre 5 cibles au maximum par itération. Le vecteur d'état à l'instant k est $\mathrm{x_k} = [x, y, v_x, v_y]^T$ où x et y sont les coordonnées de position et v_x et v_y les coordonnées de vitesse. Le modèle d'évolution des cibles est le modèle linaire et Gaussien fourni par l'équation (4.45). La probabilité qu'une cible de l'instant $k-1$ survive à l'instant k est $p_s = 0.97$.

$$\mathrm{x}_{k|k-1} = F_k\,\mathrm{x}_k + w_k \tag{4.45}$$

Avec :

$$F_k = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.46}$$

w_k est un bruit blanc de moyenne nulle et de covariance Q_k.

$$Q_k = \sigma_v^2 \begin{pmatrix} \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix} \tag{4.47}$$

Où $\sigma_v = 5m$ représente l'écart type de l'erreur sur la vitesse et $T = 1s$ la période d'échantillonnage. Chaque cible est détectée avec une probabilité de détection $p_D = 0.98$ et génère une observation suivant le modèle d'observation linaire et Gaussien fourni par l'équation (4.48).

$$\hat{z}_k = H_k \, \mathrm{x}_{k|k-1} + v_k \tag{4.48}$$

Avec

$$H_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{4.49}$$

v_k est un bruit blanc de moyenne nulle et de covariance R_k

$$R_k = \sigma_o^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4.50}$$

$\sigma_o = 10m$ l'écart type de bruit d'observation. Les observations simulées sont affectées avec des fausses alarmes. Les fausses alarmes sont modélisées avec un RFS de Poisson dont l'intensité est $\kappa_k(z) = \lambda_c V u(z)$ où $u(.)$ représente une loi uniforme sur la région de surveillance, $V = 2.25 \times 10^6 m^2$ est le volume de la région de surveillance et $\lambda_c = 10^{-5} m^{-2}$ est le nombre moyen de fausses alarmes par une unité de volume. Nous considérons que le nombre maximum d'observations est $M = 100$.

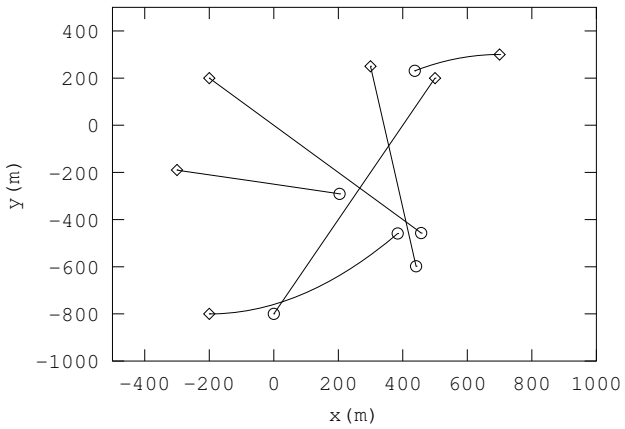

FIGURE 4.5 – Trajectoires des cibles simulées : ◇ et ○ sont, respectivement, le début et la fin d'une trajectoire.

Le mélange de Gaussiennes modélisant la naissance spontanée de cibles est $\gamma_k(\mathrm{x}) = \sum_{i=1}^{4} 0.1\mathcal{N}(\mathrm{x}; m_\gamma^{(i)}, P_\gamma^{(i)})$ où $P_\gamma^{(i)} = diag([50, 50, 50, 50])$ pour $i = 1 \dots 4$, $m_\gamma^{(1)} = [700, 300, 0, 0]^T$, $m_\gamma^{(2)} = [300, 250, 0, 0]^T$, $m_\gamma^{(3)} = [-200, 150, 0, 0]^T$ et $m_\gamma^{(4)} = [-200, 200, 0, 0]^T$. Ce modèle est choisi à partir de la connaissance a priori sur les endroits de naissance de cible. Cette connaissance est extraite des données de simulation.

Pour le GMPHD, les seuils de fusion et de suppression des composantes du mélange de Gaussiennes sont respectivement $U = th_{merging} = 4$ et $TH = th_{pruning} = 10^{-5}$. Le nombre de composantes dans le mélange de Gaussiennes est limité à $J_{max} = 10$. Dans FL_GMPHD et FX_GMPHD la fusion et la suppression ne sont pas effectuées et les J_{max} composantes avec les plus forts poids sont gardées à chaque itération.

Dans les versions approximées de GMPHD, la fonction e^{-z} est représentée avec les 5 segments de droite qui sont décrits dans l'équation (4.51). Les allures de la fonction e^{-z} et de son approximation sont fournies dans la figure (4.6).

$$e^{-z} = \begin{cases} -0.630z + 1 & \text{if } 0 \leq z \leq 1 \\ -0.280z + 0.648 & \text{if } 1 < z \leq 1.6 \\ -0.117z + 0.387 & \text{if } 1.6 < z \leq 2.8 \\ -0.035z + 0.158 & \text{if } 2.8 < z \leq 4 \\ -0.006z + 0.042 & \text{if } 4 < z \leq 7 \\ 0 & \text{if } z > 7 \end{cases} \tag{4.51}$$

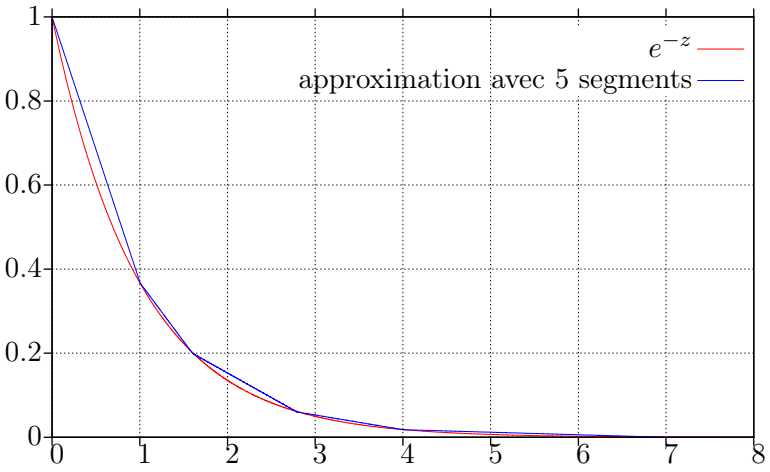

FIGURE 4.6 – Allure de e^{-z} comparée avec l'allure de son approximation avec 5 segments de droite. Au delà de $z = 7$, la valeur de e^{-z} est quasi nulle.

L'expérimentation consiste en 500 simulations indépendantes de longueur 110 itérations chacune. Toutes les simulations sont basées sur les mêmes trajectoires (figure 4.5). Une simulation consiste en la génération d'observations (fausses alarmes, non-détections et bruit d'observation inclus) à partir des trajectoires des cibles, en utilisant les valeurs des paramètres présentés : p_D, $\kappa_k(z)$ et σ_o.

4.5.2.2 Qualité de suivi

La qualité de suivi est mesurée avec la métrique *Optimal Sub-Pattern Assignment* (OSPA-T) [Ristic et al., 2011] définie par l'équation (4.52). OSPA-T prend en considération les erreurs liées aux positions des cibles, au nombre estimé de cibles et aux identités incorrectes quand les cibles sont représentées avec des points.

$$D_{OSPA-T}(\mathfrak{X}_k, \mathfrak{D}_k) = \left[\frac{1}{n}\left(\min_{\pi\in\prod_n}\sum_{i=1}^{m}\left(\min(c, d\left(\tilde{x}_{k,i}, \tilde{y}_{k,\pi(i)}\right)\right)\right)^p + (n-m)c^p\right)\right]^{\frac{1}{p}} \quad (4.52)$$

— $\mathfrak{X}_k = \{(l_1, x_{k,1}), \ldots, (l_m, x_{k,m})\}$ et $\mathfrak{D}_k = \{(s_1, y_{k,1}), \ldots, (s_n, y_{k,n})\}$ sont, respectivement, l'ensemble des pistes de la vérité terrain et l'ensemble des pistes estimées par l'algorithme de suivi, à l'instant k.
— m et n sont, respectivement, le nombre de pistes de la vérité terrain et le nombre de pistes estimées à l'instant k.
— $x_{k,i}$ et $x_{k,j}$ sont respectivement les positions de la $i^{\text{ème}}$ piste de la vérité terrain et de la $j^{\text{ème}}$ piste estimée par l'algorithme de suivi.
— l_i et s_j sont respectivement les identités de la $i^{\text{ème}}$ piste de la vérité terrain et de la $j^{\text{ème}}$ piste estimée par l'algorithme de suivi.
— $\tilde{x}_{k,i} = (l_i, x_{k,i})$ et $\tilde{y}_{k,j} = (s_j, y_{k,j})$.
— $\prod_n$ est l'ensemble des permutations de longueur m dont les éléments sont pris de $\{1, \ldots, n\}$.
— c est un paramètre permettant de modéliser le coût de l'erreur d'estimation de nombre de cibles et de borner l'erreur entre les pistes de la vérité terrain et les pistes estimées.
— p est un paramètre modélisant la sensibilité de la métrique aux données aberrantes (les pistes estimées qui ne sont proches d'aucune piste de la vérité terrain).

$$d(\tilde{x}, \tilde{y}) = \left(d(x, y)^{p'} + d(l, s)^{p'}\right)^{\frac{1}{p'}} \quad (4.53)$$

— $d(x, y)$ la distance entre la position x et la position y qui peut être la distance euclidienne.
— $d(l, s)$ modélise le coût des erreurs sur les identités des cibles. $d(l, s) = 0$ si $l = s$ et égal à α dans le cas contraire, avec $\alpha \in [0, c]$.

L'équivalence entre les identités de la vérité terrain et les identités estimées est déterminée avec l'association minimisant la distance globale entre les pistes estimées et les pistes de la vérité terrain. Les algorithmes d'association comme l'algorithme Munkres ou Auction sont utilisés.

La figure 4.7 représente la distance OSPA-T moyenne mesurée pour les trois différentes versions de GMPHD sur toutes les simulations. La qualité de suivi est inversement proportionnelle à la valeur de OSPA-T. Comme attendu, GMPHD fournit des meilleurs résultats comparé aux versions approximées. La distance OSPA-T des versions approximées augmente aux moments des naissances spontanées de cibles, i.e. aux itérations 20, 25 et 65. Cette augmentation est causée par le fenêtrage statistique résultant des approximations.

En effet, le fait que la valeur de $e^{-z} = 0$ si $z > 7$ force l'algorithme à ne pas tenir compte des observations qui sont loin des moyennes des Gaussiennes. L'effet du fenêtrage est plus important dans FX_GMPHD, car la dynamique des variables est plus réduite par rapport au codage en virgule flottante.

Par exemple, supposons que la composante t du mélange de Gaussiennes possède un faible poids à l'instant k, ce qui est le cas des composantes modélisant la naissance de cibles. Si la vraisemblance de l'observation qui lui est la plus proche (en sens de vraisemblance) $L_t = \max_{z \in Z_k} \frac{e^{-0.5(z-m^t)^T {S^t}^{-1}(z-m^t)}}{\sqrt{(2\pi)^n |S^t|}}$ est aussi faible (m^t et S^t sont respectivement la moyenne et la covariance d'innovation de la composante t), le résultat de la multiplication $\omega_{k-1} L_t$ sera nul dans FX_GMPHD à cause de la dynamique limitée. Ainsi, la cible associée à la composante t sera perdue. Dans la version FL_GMPHD, même si la valeur de $\omega_{k-1} L_t$ est très faible, la normalisation effectuée dans l'équation (4.37) engendre un poids non nul et la composante t ne sera pas perdue.

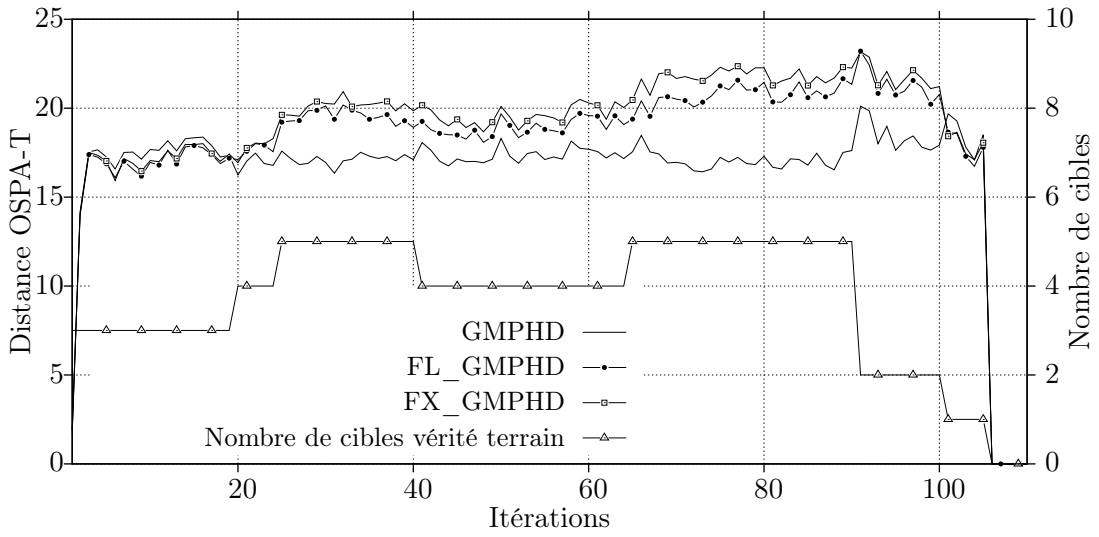

FIGURE 4.7 – Distance OSPA-T moyenne ($p' = p = 2$ et $\alpha = c = 50$) résultante des simulations : GMPHD est le filtre original, FL_GMPHD est la version approximée codée en virgule flottante et FX_GMPHD est la version approximée codée en virgule fixe.

Le fenêtrage statistique peut améliorer les performances de suivi en rendant le GMPHD moins sensible aux fausses alarmes. En effet, les versions approximées engendrent une faible distance OSPA-T comparées au GMPHD pour les itérations 1 à 20 et 100 à 105 dans la figure 4.7.

La quantification de la dégradation de la qualité du suivi est effectuée en calculant l'erreur relative moyenne (ERM) donnée par l'équation (4.54).

$$ERM = \frac{100}{K} \sum_{k=1}^{K} \frac{OSPA_{org}(k) - OSPA_{app}(k)}{OSPA_{org}(k)} \tag{4.54}$$

— K est la longueur de la séquence (110).

— $OSPA_{org}(k)$ est la valeur de la distance OSPA-T du filtre original (GMPHD) à l'instant k
— $OSPA_{app}(k)$ est la valeur de la distance OSPA-T du filtre approximé (FL_GMPHD ou FX_GMPHD) à l'instant k.

L'erreur relative moyenne est de 11.46% pour FL_GMPHD et 14.62% pour FX_GMPHD.

4.5.2.3 Vitesse d'éxecution

Les vitesses d'exécution des trois versions sont mesurées, à partir des implémentations en C, en utilisant des outils permettant de déterminer le nombre de cycles d'horloge nécessaires à partir des exécutions. Deux processeurs sont considérés pour l'expérimentation : ARM Cortex-A9 et x86.

Pour le Cortex-A9, la carte de développement Versatile Express [4] est utilisée. Le processeur présent dans la carte est composé de 4 coeurs Cortex-A9 ayant chacun une fréquence d'horloge de 400 MHz. Le co-processeur SIMD NEON est désactivé et seul un coeur est utilisé dans nos expérimentations. La nombre de cycles dans la Versatile Express est déterminé grâce au compteur matériel de performance (*hardware performance counter*), présent sur la carte. Deux exécutions sont effectuée sur le Cortex-A9. La première avec FPU activée (arm-FPU) et la deuxième avec FPU désactivée (arm-non-FPU).

Le processeur x86 est un core 2 Duo fonctionnant à une fréquence d'horloge de 3 GHz. Le nombre de cycles est mesuré avec l'outil *callgrind* de Valgrind [Nethercote et al., 2006]. L'exécution sur le x86 est effectuée avec FPU activée (x86-FPU).

Le nombre de cycles d'horloge est fourni dans la figure 4.8, pour les exécutions des trois versions du GMPHD sur les deux processurs.

Sur le x86, le FL_GMPHD est 2.4 plus rapide que le GMPHD. La conversion flottant-fixe permet un gain d'un facteur 1.21. Ce gain est faible car le x86 possède une FPU optimisée.

L'exécution des différentes versions sur le Cortex-A9, quand la FPU est activée, montre que le FL_GMPHD est 2.48 fois plus rapide que le GMPHD. Le codage en virgule fixe améliore la vitesse d'exécution d'un 1.44. Ce gain est plus important que celui perçu dans x86 car la FPU du Cortex-A9 est moins efficace.

La désactivation de la FPU du Cortex-A9 permet de simuler un processeur embarqué sans FPU. Dans ce cas, le FL_GMPHD est 3.68 fois plus rapide que le GMPHD et la conversion en virgule fixe améliore la vitesse d'exécution d'un facteur 8.88. Ainsi, la version FX_GMPHD est 32 fois plus rapide que le GMPHD original.

Les temps d'exécution peuvent être déduits de la figure 4.8 en divisant le nombre de cycles par la fréquence d'horloge. Même si la différence du nombre de cycle entre x86 et Cortex-A9 n'est pas importante, le temps d'exécution n'est pas le même à cause de la différence des fréquences d'horloge. En effet, pour le GMPHD, le temps d'exécution moyen

4. http ://infocenter.arm.com/help/index.jsp ?topic=/com.arm.doc.subset.boards.express/index.html

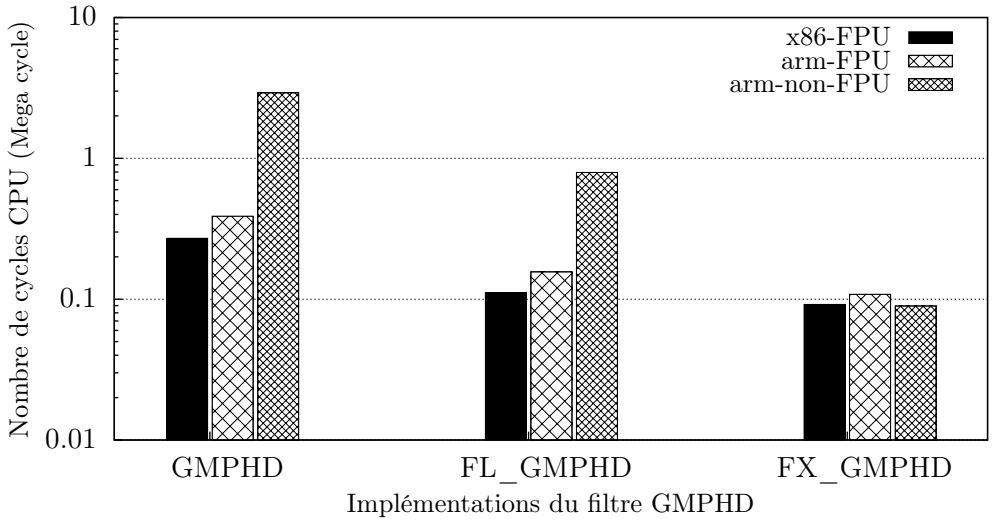

FIGURE 4.8 – Nombre de cycles CPU moyen par itération des différentes versions de GMPHD sur les deux processeurs x86 et ARM Cortex-A9.

est de 0.09 *ms* sur x86 et $0,97$ *ms* sur Cortex-A9 quand la FPU est activée. Ce résultat démontre la difficulté d'un traitement temps réel avec un processeur embarqué.

Si le processeur Cortex-A9 à 1 GHz possède une puissance de calcul de 1 GFLOPS, par interpolation, le processeur Cortex-A9 à 400 MHz possède une puissance de 400 Mega FLOPS. En se basant sur la complexité théorique du GMPHD calculée dans la section 4.4, ce processeur devrait effectuer 2900 itérations pas seconde. Cependant, le vitesse d'exécution mesurée est 1030 itérations par secondes. Cette différence est due au fait que l'estimation théorique n'inclut pas le coût de la phase de fusion et de suppression ainsi que les coûts des accès mémoire. En effet, en considérant la version FL_GMPHD qui n'utilise pas d'étape de fusion et de suppression, le nombre d'itérations par seconde est 2558 ce qui se rapproche de la valeur théorique estimée.

Les résultats obtenus dans cette section montrent la faisabilité du filtre GMPHD, même dans sa version originale, sur une architecture à base d'un processeur embarqué. Même si le scénario utilisé est simple (5 cibles au maximum), le fait que le processeur peut effectuer 1000 itérations par seconde, rend un scénario plus complexe faisable en temps réel.

4.5.2.4 Besoins en mémoire

Les besoins en mémoire des trois versions sont mesurés avec deux méthodes différentes. La première méthode est la méthode manuelle qui consiste à compter le nombre de bits des structures de données à partir des codes C des implémentations. La deuxième méthode consiste à utiliser l'outil *massif* de Valgrind [Nethercote et al., 2006], qui permet d'estimer la taille de la pile du programme exécuté. Les résultats obtenus avec les deux méthodes sont présentés dans le tableau 4.3.

TABLE 4.3 – Besoins en mémoire des différentes versions.

Implémentation	GMPHD	FL_GMPHD	FX_GMPHD
Estimation manuelle (ko)	128	59	37
Estimation avec *massif* (ko)	65	31	20

Les résultats du tableau 4.3 montrent que la mémoire estimée par l'outil *massif* est inférieure à celle estimée manuellement. Cette différence peut s'expliquer par le fait que l'estimation manuelle représente le pire cas et que dans la réalité le nombre d'observations n'atteint jamais le nombre maximum alloué. En effet, comme l'allocation de la mémoire pour le mélange de Gaussiennes s'effectue en statique en fonction du nombre de composantes précédentes dans le mélange, du nombre de Gaussiennes composant le modèle de naissance de cible est le nombre d'observations fournies par le détecteur, plus le nombre d'observations est petit, moins le programme utilise de mémoire, comparé au pire cas qui correspond à l'estimation manuelle. Les résultats montrent aussi que le besoin en mémoire est divisé par deux pour FL_GMPHD et par trois pour FX_GMPHD.

La mémoire des exécutables générés pour les différentes versions de GMPHD est fournie dans le tableau 4.4. Le tableau 4.4 montre que la taille de l'exécutable est plus importante quand la FPU est désactivée, ce qui peut s'expliquer par le fait que le compilateur inclut par défaut la librairie permettant l'émulation des opérations flottantes même si le code ne contient aucune opération flottante. Nous constatons aussi que les tailles des exécutables des approximations sont plus faible, car FL_GMPHD et FX_GMPHD nécessitent moins d'instructions que GMPHD. Les besoins en mémoire sont de l'ordre de 700 ko pour la

TABLE 4.4 – Taille mémoire des fichiers exécutables générés pour le processeur ARM Cortex-A9 pour les différentes version de GMPHD.

Implémentation	GMPHD	FL_GMPHD	FX_GMPHD
FPU activée (ko)	520	480	480
FPU désactivée (ko)	672	616	608

version originale de GMPHD et de l'ordre de 600 ko pour les versions approximées, ce qui peut être satisfaisant pour un grand nombre des architectures embarquées présentées dans le tableau 1.1.

Après cette analyse des besoins en mémoire et de la complexité de calcul du filtre GMPHD et après avoir démontré la faisabilité du GMPHD sur une architecture embarquée, nous nous intéressons dans, la prochaine section, aux limites du filtre original quand il est utilisé dans un système de vision.

4.6 GMPHD dans un système de vision

La première utilisation du filtre PHD (implémentation à base du filtre à particules) dans un système de vision a été réalisée dans [Wang et al., 2006] pour le suivi d'un nombre variable de groupes de personnes dans une vidéo. Une fois que l'implémentation du filtre à base d'un mélange de Gaussiennes a été élaborée, les mêmes auteurs ont refait le travail avec le filtre GMPHD [Wang et al., 2007]. Dans les deux travaux, les auteurs ont utilisé une soustraction de fond pour la détection des groupes de personnes. Par la suite, le filtre GMPHD a été utilisé dans [Hoseinezhad et al., 2009] afin de filtrer et d'améliorer la qualité d'un détecteur basé sur la méthode de soustraction de fond à base d'un modèle Gaussien. Les auteurs ont montré que leur détecteur assure avec moins de temps d'exécution des performances proches d'autres détecteur basés sur des méthodes de soustraction de fond plus complexes comme KDE et GMM. D'autres auteurs [Vijverberg et al., 2009, Eiselein et al., 2012, Lamard et al., 2012b, Yazdian-Dehkordi et al., 2012, Adeli et al., 2012] se sont intéressés aux occultations qui se produisent quand le filtre GMPHD est utilisé dans un système de vision. Afin de comprendre comment se produisent les occultations, nous avons utilisé le filtre GMPHD original avec un détecteur à base de la méthode de soustraction de fond First-Order low pass filter (chapitre 3) appliqué sur des images en niveau de gris de la base de données PETS2009-S2L1-view1. À chaque itération, les moyennes des Gaussiennes modélisant la naissance des cibles sont définies par les observations non associées aux cibles confirmées.

FIGURE 4.9 – Exemples d'images montrant le problème d'occultation du filtre GMPHD. La première ligne représente les résultats de détection et la deuxième ligne les résultats de suivi.

La figure 4.9 montre le problème des occultations dans le GMPHD original. Quand un objet est occulté par un autre objet ou par l'environnement pour quelques itérations, le filtre GMPHD perd les identités des objets. En effet, la perte des identités est causée par la suppression de la composante Gaussienne correspondant à la cible. Quand le détecteur fournit une observation pour un ensemble des objets en occultation, cette dernière contribue d'une manière importante dans le poids de la composante la plus proche au sens de

la vraisemblance. Par contre, sa contribution est faible dans les autres composantes. La faible contribution est due à la vraisemblance basée sur l'inverse de l'exponentielle. Ainsi, pour une petite différence dans la distance de Mahalanobis qui est le paramètre d'entrée de l'inverse de l'exponentielle, l'utilisation de vraisemblance amplifie cette différence. Lorsque les poids des composantes sont normalisés pour l'observation, avec l'équation (4.37), les valeurs des poids des composantes dont l'observation a faiblement contribué tendent vers 0, ce qui produit la disparition des composantes Gaussiennes des cibles. Comme GMPHD prend en considération les non-détections, avec le paramètre modélisant la probabilité de détection, le filtre arrive à suivre la cible pendant une itération même si elle n'est pas détectée (itération 458). Après une itération, le filtre perd la cible et il lui affecte une nouvelle identité à la fin de l'occultation (itération 471).

Le changement d'identité dans un système de suivi affecte très fortement les performances du système à cause de la non cohérence des trajectoires des objets. Dans le chapitre 5, nous proposons une solution de faible complexité permettant de résoudre le problème d'occultations entre objets dans le filtre GMPHD.

4.7 Conclusion

Après la description et l'analyse des algorithmes de suivi multi-objet se basant sur la caractéristique de mouvement, nous avons sélectionné le filtre GMPHD. L'étude détaillée de la complexité du filtre a été réalisée afin de démontrer théoriquement la faisabilité d'un suivi en temps réel sur des processeurs embarqués en utilisant GMPHD. Afin d'explorer la possibilité du fonctionnement du filtre sur des architectures très contraintes, nous avons réalisé quelques approximations sur le filtre dans le but de réduire le temps d'exécution et la quantité de mémoire requise. La dégradation de la qualité de suivi causée par les approximations est de l'ordre de 15%. Cependant, le gain en temps d'exécution sur un processeur embarqué sans FPU est d'un facteur 32, comparé à la version originale codée en virgule flottante. Le besoin en mémoire du filtre a été réduit de 128 ko à 37 ko, ce qui signifie un gain de 70%. Finalement, afin de comprendre les limites du filtre pour le suivi d'objets dans un système de vision, nous l'avons couplé à un détecteur à base de la soustraction de fond. Les principales limites du filtre sont les occultations entre objets. Dans le prochain chapitre, une méthode de faible complexité est proposée pour la résolution d'occultation dans le filtre GMPHD.

Chapitre 5

Résolution d'occultations dans le filtre GMPHD

Sommaire

La principale limite du filtre GMPHD, quand il est appliqué pour le suivi visuel d'objets, est la non gestion des occultations qui est un problème classique pour les algorithmes de suivi. Comme vu dans le chapitre précédent, le filtre GMPHD perd les identités des objets quand une occultation se produit et leur affecte de nouvelles identités quand l'occultation est terminée. Le changement d'identité affecte sévèrement les performances du suivi à cause de la perte de la cohérence des trajectoires des objets. Dans ce chapitre nous présentons une méthode de résolution d'occultations qui peut utiliser des caractéristiques différentes (apparence visuelle ou caractéristique de mouvement) et qui s'intègre efficacement dans le filtre GMPHD.

La première section du chapitre présente les travaux existants qui se sont intéressés à la résolution d'occultations pour le filtre GMPHD. La deuxième section explique la méthode proposée. Les performances de la méthode proposée sont présentées dans la dernière section du chapitre en utilisant les bases de données PETS2009 et CAVIAR.

5.1 Travaux existants

Initialement, le filtre GMPHD a été élaboré pour suivre des objets sous forme de points. Dans [Vijverberg et al., 2009], le filtre GMPHD a été étendu aux objets représentés avec des rectangles. Premièrement, les informations de la largeur et la hauteur des rectangles ont été ajoutées au vecteur d'état. La distance de Mahalanobis, dans l'étape de fusion des composantes Gaussiennes du mélange, a été replacée par la distance de Fisher donnée dans l'équation (5.1). La distance de Fisher a pour objectif de fusionner des rectangles proches tout en favorisant ceux dont la taille est petite. La gestion des occultations s'effectue en deux étapes. La première étape est la détection des occultations après la phase de prédiction. Cette étape consiste en le calcul de la distance de Fisher entre les composantes

du mélange. Si la distance entre deux composantes est inférieure à un seuil prédéfini, il y a une grande probabilité que les deux objets associés à ces composantes seront représentés par une seule observation. Dans ce cas, les objets sont considérés en occultation. Un objet *composé* représenté par une Gaussienne qui contient les références des objets en occultation est initialisé. Tant que l'occultation n'est pas finie, les états de l'objet composé et des objets originaux sont prédits et corrigés avec le filtre GMPHD. À chaque itération, après l'étape de fusion des pistes, la fin de l'occultation est détectée. La fin de l'occultation est indiquée par trois cas. Selon le cas, la gestion des objets occultés est différente. Les trois cas indiquant la fin de l'occultation sont les suivants :

1. Cas 1 : un objet j référencé dans l'objet composé est loin de tous les autres objets référencés dans l'objet composé au sens de la distance de Fisher. Dans ce cas, l'objet j n'est plus occulté et sa référence est donc supprimée de l'objet composé.
2. Cas 2 : le nombre d'objets référencés dans l'objet composé est inférieur à 1. Dans ce cas, il n y a plus d'occultation et l'objet composé est supprimé.
3. Cas 3 : le poids de l'objet composé est supérieur à la somme des poids des objets originaux le composant. Dans ce cas, les objets originaux sont supprimés et l'objet composé est gardé, ce qui engendre une fusion irréversible des objets.

$$D_{Fisher}(\mu_1, \mu_2) = \frac{(p_{1x} - p_{2x})^2}{W_1^2/4 + W_2^2/4} + \frac{(p_{1y} - p_{2y})^2}{H_1^2/4 + H_2^2/4} \tag{5.1}$$

Avec $\mu_i = [p_{ix}, p_{iy}, W_i, H_i]$. W_i et H_i sont la largeur et la hauteur de l'objet.

L'avantage de cette méthode est qu'elle ne nécessite pas beaucoup de traitement et de mémoire supplémentaires. En effet, pour un groupe d'objets occultés, une seule Gaussienne est ajoutée. Concernant la mémoire, la Gaussienne est composée d'un état, un poids, une matrice de covariance et les références des objets occultés. Au niveau calcul, le surcoût est dû aux Gaussiennes représentant les objets composés, à la détection des objets en occultation et à la gestion des occultations. En supposant que le nombre d'objets composés est N_c, la complexité supplémentaire se traduit par $J_{max}^{method} = J_{max}^{GMPHD} + N_c$. Chacune de la détection d'occultation et la gestion d'occultation possède le même ordre de complexité que la phase fusion de GMPHD, à l'exception que la distance de Fisher est plus simple que la distance de Mahalanobis (pas d'opérations matricielles). Cependant, la méthode présente plusieurs inconvénients. Les auteurs n'ont pas pris en considération les identités des objets, ce qui rend le 3ème cas ambigu si des identités sont utilisées (quelle identité sera attribuée à l'objet composé ?). L'autre problème est que le poids des Gaussiennes chute rapidement quand une occultation se produit (section 4.6), ce qui impose une fusion irréversible des objets et engendre la perte des objets.

La résolution des occultations, dans [Eiselein et al., 2012], est basée sur la méthode développée dans [Panta et al., 2009] qui consiste en la propagation des pistes comme dans la méthode d'association MHT orienté piste. La cohérence dans l'attribution des identités est assurée par l'utilisation de plusieurs arbres d'identité. Chaque arbre correspond à une seule identité qui représente un objet unique. Chaque branche d'un arbre est une piste possible pour la trajectoire de l'objet. La propagation des pistes s'effectue de la même manière que la méthode MHT orienté piste [Blackman, 2004]. Chaque arbre est classifié

comme confirmé ou tentative selon les poids des Gaussiennes composant ses pistes, et un fenêtrage temporel est utilisé pour limiter le nombre de pistes. Deux cibles i et j ayant la même matrice de covariance P sont considérées en occultation, quand $|m_i - m_j| < 2|P|^{\frac{1}{2}}$ où m_i et m_j sont les vecteurs d'état des cibles. Quand une occultation est détectée, un arbre est défini pour chacune des pistes en occultation. À chaque itération, des hypothèses de pistes sont formulées dans chaque arbre en se basant sur les associations possibles fournies par le filtre GMPHD. Le score de chaque hypothèse est évalué avec un LLR (log-likelihood ratio). Les hypothèses et les scores sont propagés jusqu'à la fin de l'occultation ($|m_i - m_j| > 2|P|^{\frac{1}{2}}$). Une fois l'occultation terminée, la piste hypothèse avec le score le plus élevé est sélectionnée dans chaque arbre.

Le même principe a été adapté dans [Eiselein et al., 2012] pour un système de vision en utilisant la caractéristique de la couleur. Chaque objet est représenté avec un histogramme couleur qui est mis à jour à chaque itération. Si deux objets sont proches, ils sont considérés en occultation et le fenêtrage temporel est désactivé. Les états des deux objets existent dans les deux arbres d'identité. Dans ce cas, les états des objets sont déterminés par la correspondance d'histogrammes. À la fin de l'occultation, l'information couleur permet de distinguer les deux objets. Toutes les branches incohérentes d'un arbre d'identité sont alors supprimées. De plus, afin d'améliorer les résultats de suivi, les auteurs ont intégré deux détecteurs complémentaires (détection à base de soustraction de fond et détecteur de têtes) dans le filtre GMPHD. L'avantage de cette méthode est l'exploitation de l'information visuelle (couleur) qui permet une meilleure discrimination entre les objets en occultation. Cependant, cette méthode possède les mêmes inconvénients que la méthode MHT à cause de la propagation des hypothèses de pistes. En effet, comme le fenêtrage temporel est désactivé durant l'occultation, le nombre d'hypothèses peut exploser si l'occultation dure longtemps. La complexité calculatoire de la méthode s'avère inadéquate pour des architectures embarquées. D'ailleurs, l'implémentation en C++ du filtre GM-PHD amélioré quand les observations sont fournies par deux détecteurs, s'exécute à 35 fps sur une machine core i7 fonctionnant à 2.67 GHz quand le temps des détections utilisées ne sont pas inclus dans le temps de traitement et quand le nombre maximum d'objets égal à 10.

Dans [Lamard et al., 2012b], la résolution d'occultation a été développée pour le filtre GMCPHD et le MHT dans le contexte routier. La même approche peut être aussi utilisée pour le filtre GMPHD. Le suivi d'objets s'effectue dans le repère monde et la probabilité de détection est dépendante de l'état. La gestion des occultations s'effectue en intégrant dans la probabilité de détection une probabilité d'occultation. La probabilité de détecter une cible se trouvant à la position (x, y) est conditionnée par la probabilité de détection du détecteur à cette position et la probabilité qu'aucun objet n'interfère entre la cible et le capteur. La probabilité de détection du détecteur est fixée par la statistique du détecteur. La probabilité qu'aucun objet n'interfère entre la cible et le capteur est calculée en se basant sur la probabilité qu'une partie d'un objet G se trouve entre le capteur et la cible. Afin de calculer cette probabilité, la densité de probabilité de l'état de l'objet G est projetée sur la droite perpendiculaire à l'axe reliant le point x, y au capteur (figure 5.1). Une convolution du résultat de projection avec la densité de probabilité modélisant la largeur de la cible G, permet de déterminer la probabilité d'occultation de la position (x, y) par l'objet G. L'avantage de cette approche est son élégance mathématique qui

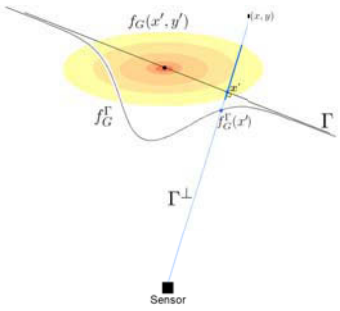

FIGURE 5.1 – Projection de la densité d'état d'une cible sur l'axe perpendiculaire à l'axe reliant le point d'intérêt x, y et la caméra [Lamard et al., 2012b]

lui permet une intégration parfaite dans le filtre. Cependant, quelques inconvénients de la méthode peuvent être notés. Le fait de modéliser que la probabilité de détection en fonction de l'occultation impose que le mouvement de la cible occultée reste fidèle au modèle d'évolution durant l'occultation. En effet, si la cible change de vitesse ou de direction durant l'occultation, la cible ne pourra pas être ré-identifiée. Le deuxième inconvénient est que l'information sur la position de la caméra par rapport aux cibles doit être connue pour pouvoir calculer la probabilité de détection, ce qui n'est pas toujours le cas.

Dans [Yazdian-Dehkordi et al., 2012], l'occultation est modélisée dans la probabilité de détection dépendante de l'état de la cible. En effet la probabilité de détection est donnée par l'équation (5.2).

$$P_D(x^i) = P(D(x^i)|C(x^i))P(C(x^i)) + P(D(x^i)|\bar{C}(x^i))P(\bar{C}(x^i)) \tag{5.2}$$

Avec $P_D(x^i)$ est la probabilité de détecter la cible x^i. $D(x^i)$ signifie que la cible x^i est détectée. $C(x^i)$ signifie que la cible x^i est occultée. $\bar{C}(x^i)$ signifie que la cible x^i n'est pas occultée. Deux cibles sont en occultation si leur pourcentage de chevauchement est supérieur à un seuil prédéfini. La probabilité qu'une cible est détectée sachant qu'elle est occultée par une autre cible dépend du détecteur et de la distance entre les deux cibles. Ce même principe est appliqué dans [Adeli et al., 2012] quand le filtre GMPHD basée sur les arbres d'identités [Panta et al., 2009] est utilisé. Cette méthode intègre bien l'occultation dans le filtre GMPHD mais les caractéristiques visuelles ne peuvent pas être exploitées pour assurer une bonne ré-identification. Concernant la complexité, seul le calcul de la probabilité de détection ajoute un coût supplémentaire.

Dans la prochaine section, nous présentons notre méthode de résolution d'occultation qui peut exploiter tout type de caractéristiques pouvant être fournies par le détecteur.

5.2 Méthode proposée

Le filtre GMPHD décrit dans la section 4.3 est étendu pour prendre en compte les occultations entre objets (figure 5.2). Après la phase de prédiction, un module permettant la détection des occultations est ajouté. Un module de gestion d'occultation est utilisé pour détecter la fin des occultations et assigner l'identité correcte à chaque objet.

Chaque objet est décrit par les coordonnées de position, les coordonnées de vitesse, la largeur et la hauteur du rectangle le délimitant. La détection des occultations s'effectue

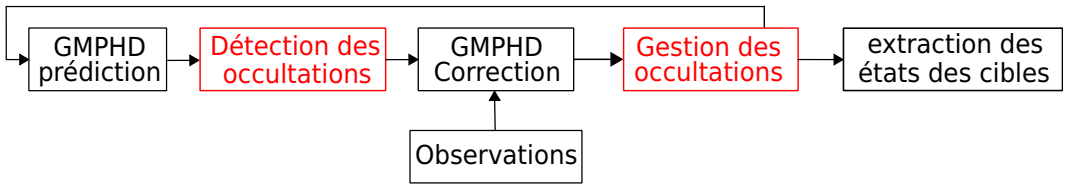

FIGURE 5.2 – Schéma du filtre GMPHD amélioré pour la résolution des occultations entre objets

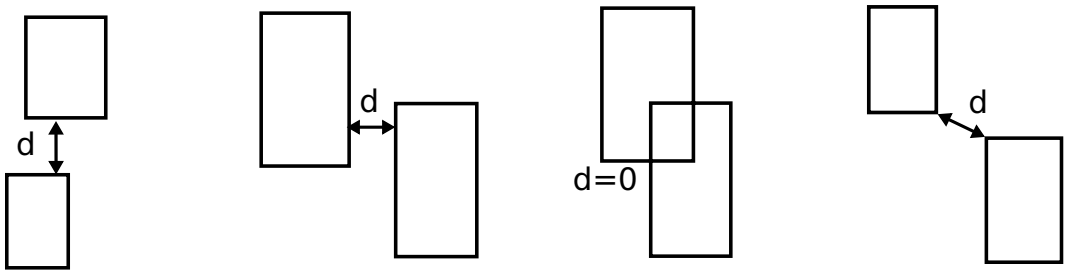

FIGURE 5.3 – Distance entre deux pistes représentées par des rectangles

après la phase de prédiction et ne considère que les objets actifs. Un objet est actif si et seulement s'il est confirmé par le filtre GMPHD (le poids de la composante Gaussienne le représentant est supérieur au seuil d'extraction de cibles) durant N itérations. Le fait de considérer que les objets actifs permet de diminuer le nombre d'objets à traiter et d'éviter que les fausses alarmes non résolues par le filtre soient considérées dans le processus d'occultations. En pratique, détecter si un objet est actif revient à ajouter, par composante Gaussienne, une variable compteur qui s'incrémente à chaque confirmation de la Gaussienne et qui décrémente quand le poids de la Gaussienne est inférieur au seuil d'extraction. La valeur du compteur est bornée par une valeur maximale et une valeur minimale afin de limiter la dynamique et d'assurer qu'un objet confirmé peut devenir non confirmé et vice versa. La détection des occultations consiste à calculer la distance entre les rectangles représentants les objets. La distance entre deux rectangles est la distance euclidienne des deux points les plus proches des rectangles (figure 5.3). Cette distance s'exprime avec l'équation (5.3). Le calcul de cette distance peut être réalisé efficacement en considérant la position d'un rectangle par rapport à un autre, en utilisant l'algorithme 3. Les covariances des cibles ne sont pas prises en compte dans le calcul de cette distance. Le calcul de cette distance est basé sur des opérations simples telles que la comparaison, l'addition, la soustraction et la multiplication. Dans le pire cas, le nombre d'opérations nécessaire est 6 comparaisons, 5 additions, 4 soustractions, 2 multiplications et deux décalages.

$$d(R_i, R_j) = \begin{cases} 0 & \text{si } R_i \text{ et } R_j \text{ se chevauchent} \\ \min\limits_{p_i \in R_i, p_j \in R_j} d_{Eucl}(p_i, p_j) & \text{sinon} \end{cases} \tag{5.3}$$

Où R_i et R_j sont respectivement les rectangles délimitant les objets i et j. p_i et p_j sont respectivement des points des rectangles R_i et R_j. $d_{Eucl}(p_i, p_j)$ est la distance Euclidienne entre les deux points p_i et p_j.

entrée: Les rectangles $R_a = (xa_0, xa_1, ya_0, ya_1)$ et $R_b = (xb_0, xb_1, yb_0, yb_1)$. xi_0, yi_0 sont les coordonnées de la limite haut-gauche du rectangle R_i. xi_1, yi_1 sont les coordonnées de la limite bas-droite du rectangle R_i.
sortie : $d(R_a, R_b)$: la distance entre les rectangles R_a et R_b
\# Déterminer la position relative des centres des rectangles
$x = \frac{xa_1+xa_0}{2} - \frac{xb_1+xb_0}{2}$
$x = \frac{ya_1+ya_0}{2} - \frac{yb_1+yb_0}{2}$
si $x \geq 0$ && $y \geq 0$ **alors**
  \# R_a est à droite et bas de R_b
  $h = xa_0 - xb_1$
  $w = ya_0 - yb_1$
sinon
  si $x \leq 0$ && $y \geq 0$ **alors**
    \# R_a est à gauche et bas de R_b
    $h = xb_0 - xa_1$
    $w = ya_0 - yb_1$
  sinon
    si $x \geq 0$ && $y \leq 0$ **alors**
      \# R_a est à droite et haut de R_b
      $h = xa_0 - xb_1$
      $w = yb_0 - ya_1$
    sinon
      \# R_a est à gauche et haut de R_b
      $h = xb_0 - xa_1$
      $w = yb_0 - ya_1$
    fin
  fin
fin
si $h \leq 0$ && $w \leq 0$ **alors**
  \# chevauchement des deux rectangles
  $d(R_a, R_b) = 0$
sinon
  si $h > 0$ && $w \leq 0$ || $y == 0$ **alors**
    $d(R_a, R_b) = h$
  sinon
    si $h \leq 0$ && $w > 0$ || $x == 0$ **alors**
      $d(R_a, R_b) = w$
    sinon
      $d(R_a, R_b) = h^2 + w^2$
    fin
  fin
fin

Algorithm 3: Calcul de la distance entre deux rectangles

Une occultation entre deux objets i et j est considérée quand la distance $d(R_i, R_j)$ est inférieure à un seuil prédéfini. Dans ce cas, un objet global représenté par une seule Gaussienne remplace les Gaussiennes des objets en occultations. Le poids, la moyenne et la covariance, de l'objet global, sont le résultat de la fusion des Gaussiennes des objets en occultations avec l'algorithme 2. L'objet global contient les identités des objets en occultation ainsi que des caractéristiques permettant de les discriminer à la fin de l'occultation (figure 5.4). Ces caractéristiques peuvent être des caractéristiques visuelles (histogramme de couleur, points d'intérêt, *etc*) ou des caractéristiques spatio-temporelles (direction du mouvement, la taille du rectangle, *etc*). Le choix des caractéristiques à utiliser dépend du scénario de suivi et de la complexité du calcul de la caractéristique. L'objet global peut donc contenir plusieurs objets en occultation (>2). La Gaussienne représentant l'objet est propagée dans le filtre GMPHD (prédiction + correction). Ce raisonnement suppose que les objets en occultation ont le même état et que la correction s'effectue avec la seule observation générée.

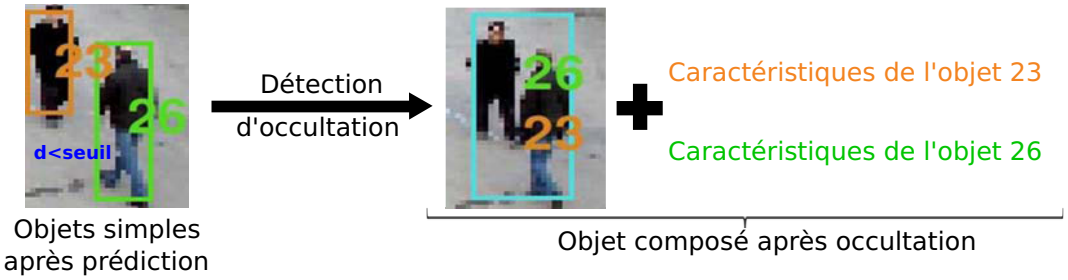

FIGURE 5.4 – Détection des occultations

La gestion des occultations s'effectue après la phase de correction. Durant la phase de correction, le filtre GMPHD autorise qu'une piste à l'itération k 1 se divise en plusieurs pistes à l'itération k. Cette division se produit quand plusieurs observations sont proches de la piste au sens de la distance de Mahalanobis et qu'aucune des autres pistes existantes n'est proche de ces observations. La division est possible car les poids des Gaussiennes sont normalisés par rapport aux observations et non pas par rapport aux pistes, dans l'équation (4.37) (la somme des poids des pistes associées à une observation est proche de 1). Dans [Clark et al., 2006b], quand cette division se produit, la Gaussienne avec le plus fort poids garde son identité et les autres reçoivent des nouvelles identités. En effet, nous gardons ce raisonnement pour le cas des objets simples (nombre d'identité égal à 1). Par contre, dans le cas des objets composés (nombre d'identité supérieur 1), si plusieurs objets confirmés (poids supérieurs au seuil d'extraction) ont les mêmes identités, alors l'objet composé s'est divisé en objets simples, ce qui indique la fin de l'occultation (figure 5.5). Quand la division se produit, une correspondance entre les caractéristiques, fournies par les observations qui ont causées la division, et les caractéristiques sauvegardées au moment où l'occultation est détectée, est effectuée (figure 5.5). L'identité d'un objet est définie par la caractéristique offrant la meilleure correspondance.

Lorsque une occultation dure longtemps au même endroit (deux objets qui s'occultent et arrêtent de se déplacer), la covariance de l'objet global diminue ce qui réduit la région d'apparition des observations prises en compte par le filtre. Ainsi, même si une division se

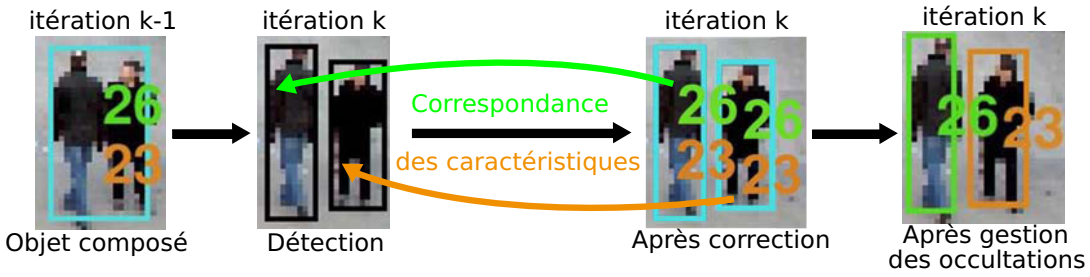

FIGURE 5.5 – Gestion des occultations

produit, l'algorithme GMPHD ne réussit pas à la détecter, car le poids des Gaussiennes résultantes de la division est inférieur au poids des Gaussiennes initialisant les nouvelles pistes. Ce problème est fixé en ajoutant aux covariances de position des objets composés une constante afin d'assurer une région assez large pour la détection des divisions.

Quand l'objet composé, se divisant en deux, contient plus de deux objets simples, il résultera un objet composé et un objet simple ou deux objets composés. Dans notre approche nous considérons toujours le premier cas. Le critère de différentiation entre l'objet simple et l'objet composé est la taille des rectangles les représentant. Le rectangle ayant la plus petite taille est considéré comme objet simple et l'autre comme objet composé. L'identité de l'objet simple est déterminée avec le processus expliqué précédemment. L'identité sélectionnée ainsi que la caractéristique correspondant à l'objet simple sont supprimées de l'objet composé. Le fait de supposer qu'un objet composé se divise toujours en un objet simple et un autre objet (simple ou composé) est la limitation de l'approche. Cependant, toutes les méthodes existantes exposées dans la section 7.5.1 sont concernées par cette limitation.

Comme le processus de fusion est coûteux en opérations, dans notre approche, il est résumé à la fusion des poids des Gaussiennes proches. La moyenne et la covariance de la Gaussienne résultante sont définies par la Gaussienne ayant le poids le plus élevé. Si deux Gaussiennes possèdent des poids supérieurs au seuil d'extraction de cibles, elles ne sont pas fusionnées. De plus, la distance de Mahalanobis est remplacée par la distance Euclidienne sur la position. Le nombre de composantes dans le mélange est limité au J_{max} composantes ayant les plus forts poids. L'extraction des états des cibles n'est pas modifiée. Les Gaussiennes dont le poids est supérieur au seuil d'extraction sont considérées comme des objets confirmés. Sur le plan image un objet est alors représenté avec le rectangle le délimitant et son identité. Dans le cas d'un objet composé, un rectangle peut avoir plusieurs identités (figure 5.5).

La méthode de résolution d'occultations proposée est indépendante de la méthode de détection utilisée et des caractéristiques choisies. Par rapport aux méthodes de l'état de l'art présentées dans la section 7.5.1, la méthode proposée est différente de celle présentée dans [Eiselein et al., 2012] par le fait qu'aucun historique sur les objets est utilisé pour la résolution d'occultation, ce qui permet d'avoir une complexité de calcul et de mémoire inférieure. La méthode proposée se différencie de la méthode [Vijverberg et al., 2009] par sa possibilité d'intégrer différentes caractéristiques notamment des caractéristiques d'ap-

parence qui sont pertinentes pour le suivi visuel. De plus la distance utilisée dans notre méthode pour la détection des occultations est plus réaliste que l'utilisation de la distance de Fisher quand des objets sont représentés avec des rectangles. La différence de notre approche par rapport à [Lamard et al., 2012b, Yazdian-Dehkordi et al., 2012] est que les auteurs traitent les occultations en modélisant la probabilité de détection qui est dépendante de l'état sans utilisation de caractéristique d'apparence ce qui n'est pas le cas dans notre méthode. Comme [Lamard et al., 2012b] effectuent un suivi sur le plan monde, la calibration de la caméra est nécessaire.

Dans la prochaine section, l'évaluation de notre méthode avec deux caractéristiques (histogramme couleur et direction du mouvement) est effectuée sur les séquences des bases de données PETS2009 et CAVIAR.

5.3 Résultats

Dans cette section la méthode présentée est évaluée et comparée au filtre GMPHD original [Clark et al., 2006b]. Pour la détection des objets, nous avons utilisé un détecteur à base de la méthode de soustraction de fond First-Order low pass filter (chapitre 3), appliqué sur les images en niveau de gris. L'état d'un objet à l'instant k est décrit par (x, y) qui sont les coordonnées de position du centre de gravité de l'objet, (v_x, v_y) qui sont les coordonnées de vitesse, w et h sont la largeur et la hauteur de l'objet. La hauteur h et la largeur w prennent les valeurs fournies par les observations et n'interviennent pas dans les phases de prédiction et de correction du filtre GMPHD. Ainsi, w et h ne sont pas mises dans le vecteur d'état. Le modèle d'évolution est un modèle linéaire Gaussien défini par l'équation (5.4).

$$\mathrm{x}_{k|k-1} = F\,\mathrm{x}_k + w_k \tag{5.4}$$

Avec

$$F = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.5}$$

w_k est un bruit blanc de moyenne nulle et de covariance Q.

$$Q = \sigma_e^2 \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{20} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{20} \\ 0 & 0 & \frac{1}{20} & 0 \\ 0 & 0 & 0 & \frac{1}{20} \end{pmatrix} \tag{5.6}$$

Où σ_e est l'écart type modélisant le bruit d'évolution. Le modèle d'observation est linéaire et Gaussien et est défini par l'équation (5.7)

$$\hat{z}_k = H\,\mathrm{x}_{k|k-1} + v_k \tag{5.7}$$

Avec

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{5.8}$$

v_k est un bruit blanc de moyenne nulle et de covariance R

$$R = \sigma_o^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5.9}$$

Où σ_o est l'écart type modélisant le bruit d'observation. Les moyennes des Gaussiennes modélisant la naissance des nouveaux objets à l'instant k sont définies par les observations non associées à l'instant $k-1$.

L'approche proposée est évaluée sur la séquence *S2L1-view1* de PETS2009 et la séquence *Meet_ WalkTogether1* de CAVIAR. Dans la séquence PETS2009, plusieurs personnes (le maximum à la fois est 9) marchent dans la scène et plusieurs occultations se produisent entre elles (3 personnes au maximum sont en occultations). L'évaluation s'effectue de l'image 200 à l'image 794. Les 200 premières images sont utilisées pour initialiser l'arrière plan. Dans la séquence *Meet_ WalkTogether1* deux personnes se rencontrent et marchent ensemble. Le suivi est effectué de l'image 1250 à 1450. Les images 1000 à 1250 sont utilisées pour l'initialisation de l'arrière plan. Les paramètres de GMPHD sont les mêmes pour le filtre original et pour l'approche proposée.

5.3.1 Caractéristiques utilisées pour la résolution d'occultations

Nous avons testé deux caractéristiques pour la résolution d'occultations : histogramme d'apparence et direction de mouvement.

Histogramme : nous utilisons l'histogramme normalisé, défini par les pixels couleurs résultant de la soustraction de fond et qui appartiennent à l'objet. À chaque itération, l'histogramme d'un objet simple est remplacé par le nouvel histogramme fourni par l'observation à laquelle il est associé. Quand une occultation se produit, les histogrammes des objets sont sauvegardés dans l'objet global résultant. À la fin de l'occultation, l'histogramme d'un objet simple, résultant de la division, est comparé à tous les histogrammes contenus dans l'objet composé. La comparaison s'effectue avec la distance euclidienne. L'identité de l'objet est alors définie par l'histogramme présentant la distance minimale. Une fois l'identité sélectionnée, l'objet simple est supprimé de l'objet composé. Suivant le nombre de niveaux utilisés et la nature des pixels (couleur ou niveau de gris), trois versions d'histogrammes sont testées. Une version avec 256 niveaux par canal de couleur. La deuxième version est 8 niveaux par canal de couleur. La troisième version est 8 niveaux extraits de l'image en niveau de gris.

Direction de mouvement : quand une occultation est détectée, les vecteurs d'état des objets en occultation sont sauvegardés comme caractéristiques dans l'objet composé. À chaque itération, les vecteurs sauvegardés subissent le processus de prédiction du filtre GMPHD. À la fin d'occultation, la direction de mouvement est utilisée pour ré-identifier les objets occultés. La direction de mouvement est déduite des signes des coordonnées vitesse de l'objet. Le signe des coordonnées de vitesse d'un objet simple résultant de la fin de l'occultation est comparé aux signes des coordonnées vitesse des caractéristiques présentes dans l'objet composé. Si le nombre de caractéristiques ayant le même signe de vitesse que l'objet est un, alors l'objet prend l'identité correspondant à cette caractéristique. Dans le

cas où plusieurs caractéristiques ont la même direction de mouvement que l'objet simple, la distance euclidienne sur la position est utilisée pour choisir parmi ces caractéristiques. Dans ce cas, la caractéristique dont la distance euclidienne est minimale, définit l'identité de l'objet. Si aucune caractéristique ne possède la même direction de mouvement que l'objet à ré-identifier, alors la distance euclidienne sur la position est utilisée pour la prise de décision. Dans ce cas, la distance euclidienne est calculée pour toutes les caractéristiques présentes dans l'objet composé. La caractéristique présentant la distance minimale définit l'identité de l'objet. Une fois que l'identité d'un objet simple est déterminée, cet objet est supprimé de l'objet composé.

5.3.2 Métriques d'évaluation

Un système de suivi multi-objet est souvent évalué suivant ces trois critères :

1. l'estimation du nombre d'objets
2. l'estimation des positions des objets
3. l'estimation des trajectoires des objets, ce qui se traduit par la cohérence dans l'attribution des identités aux objets estimés.

Les améliorations apportées au filtre GMPHD, avec la méthode proposée, sont concernées par les critères 1 et 3. En effet, la détection d'occultation intervient sur le nombre d'objets estimés. La gestion des occultations améliore la cohérence des identités en diminuant le nombre de fois que l'identité d'un objet change. Ainsi, seuls ces deux critères sont évalués pour montrer l'apport de la méthode proposée par rapport au filtre GMPHD original. Le critère 1 ne dépend pas des caractéristiques utilisées. Il est évalué en comparant le nombre estimé d'objets fourni par notre méthode, le filtre GMPHD original et le module de détection basé sur la soustraction du fond. Le critère 3 est lié à la performance de la ré-identification après occultation. Ce critère est évalué en comptant le nombre de fois que l'identité des objets change au cours de suivi, suivant la caractéristique utilisée. Plus le nombre de changement d'identités est petit, meilleure est la ré-identification et meilleur est le suivi. Quand un objet sort du champ de vision de la caméra et rentre après, il est considéré comme un nouvel objet avec une nouvelle identité.

5.3.3 Évaluation

Les figures 5.6 et 5.7 indiquent le nombre estimé d'objets pour les deux séquences d'évaluation des bases de données PETS2009 et CAVIAR. Le nombre d'objets estimé avec la méthode proposée est plus proche de la vérité terrain que le nombre estimé avec le filtre GMPHD original ou le détecteur.

En effet, même si le nombre d'objets fourni par le détecteur est erroné à cause des occultations, la méthode proposée arrive à l'estimer correctement, contrairement au filtre GMPHD original qui fournit un nombre d'objets estimé proche de celui du détecteur. Les erreurs produites pour les itérations (images) 506 à 540 de la séquence PETS2009 sont causées par le fait que les deux objets entrant dans le champ de vision sont déjà en occultation (figure 5.8). L'estimation est erronée pour les itérations 690 à 790 (PETS2009), car la personne ouvrant le coffre de la voiture est trop petite pour être détectée avec le détecteur utilisé (figure 5.9).

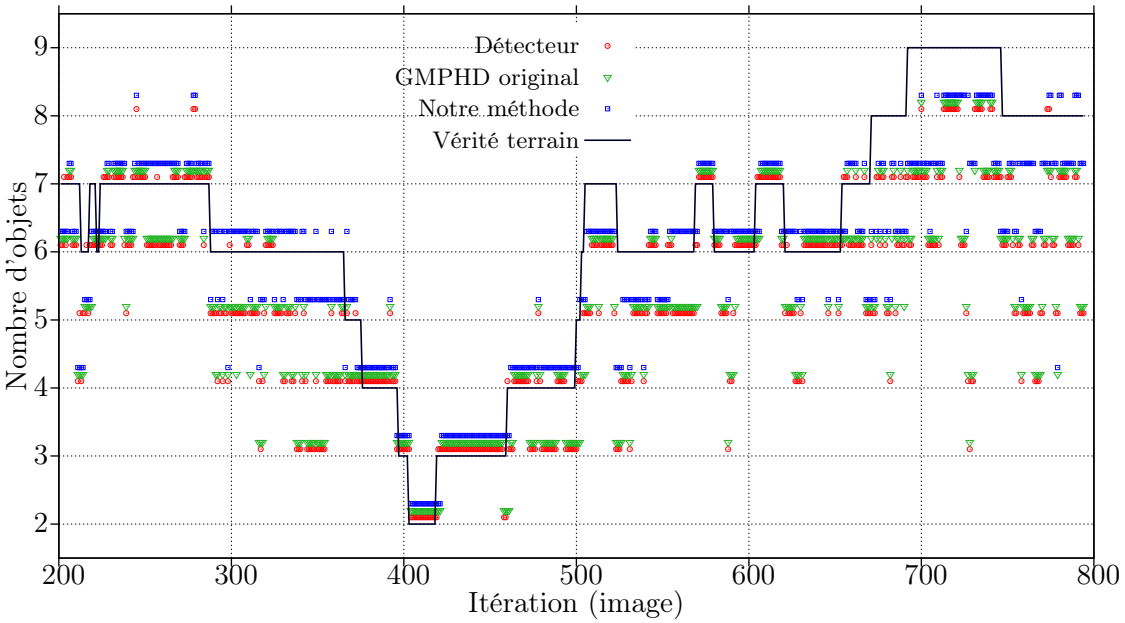

FIGURE 5.6 – Nombre estimé d'objets pour la séquence PETS2009. Le nombre d'objets, qui est un entier, est légèrement décalé verticalement suivant la méthode, pour des raisons de visualisation.

L'erreur moyenne sur l'estimation du nombre d'objets par itération pour la séquence PETS2009 est 0.61 pour notre méthode, 1.1 pour le GMPHD original et 1 pour le détecteur. Ainsi, nous obtenons une amélioration d'un facteur 1.6. Cette erreur pour la séquence CAVIAR est de 0.005 pour notre méthode et 0.49 pour le GMPHD original et le détecteur. Une amélioration d'un facteur 98 est obtenue. Le facteur d'amélioration est différent entre les deux séquences à cause du nombre d'occultations qui se produisent. Dans la séquence CAVIAR, une fois que les personnes sont en occultation, elles ne se séparent pas jusqu'à la fin de la séquence. Durant toute l'occultation, notre méthode estime correctement le nombre d'objets contrairement au filtre GMPHD et au détecteur qui fournissent un nombre erroné.

La figure 5.10 représente le nombre de fois que l'identité d'un objet change au cours de suivi, dans la séquence PETS2009. Ce résultat montre que le changement d'identité est plus faible avec notre méthode comparé au filtre GMPHD original. La caractéristique d'apparence fournie (histogrammes couleurs et niveau de gris) des meilleurs résultats comparés à la direction de mouvement. Les résultats obtenus avec les trois versions d'histogrammes utilisées sont très proches. En effet, comme l'information d'apparence n'est utilisée que quand une occultation se produit, le nombre d'objets est souvent limité ce qui assure une bonne discrimination entre objets même avec des histogrammes de 8 niveaux en niveau de gris. La plupart des changements d'identité dans cette séquence sont causés par le lampadaire se trouvant au milieu de la scène (figure 5.9). En effet, quand un objet est occulté par le lampadaire durant plusieurs itérations, aucune observation n'est générée et notre méthode ainsi que le filtre GMPHD perdent l'objet.

Le nombre de changement d'identité obtenu pour la séquence CAVIAR avec notre

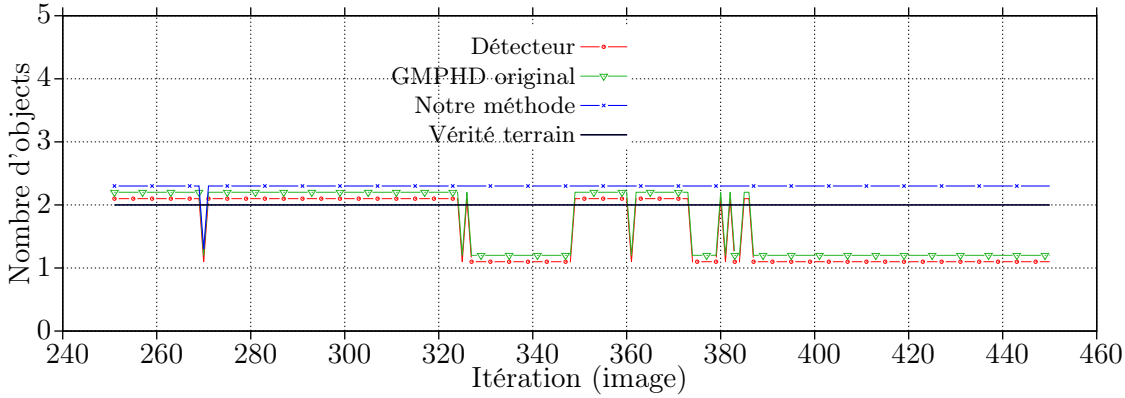

FIGURE 5.7 – Nombre estimé d'objets pour la séquence CAVIAR. Le nombre d'objets, qui est un entier, est légèrement décalé verticalement suivant la méthode, pour des raisons de visualisation.

FIGURE 5.8 – Exemple d'image montrant des personnes qui sont en occultation quand ils entrent dans le champ de vision de la caméra dans la séquence PETS2009.

FIGURE 5.9 – Exemple d'image montrant la personne ouvrant le coffre de la voiture dans la séquence PETS2009 (rectangle bleu) et une occultation causée par le lampadaire (rectangle rouge).

méthode est égal à 0 quelle que soit la caractéristique utilisée (direction de mouvement, histogramme couleur ou histogramme en niveau de gris). En appliquant le filtre GMPHD original, l'identité de l'un des objets change 5 fois et celle de l'autre objet ne change pas.

L'avantage de l'apparence par rapport à la direction de mouvement est son critère discriminatoire qui ne nécessite aucun a priori sur les objets. En effet, la direction de mouvement suppose que le mouvement des objets durant l'occultation reste fidèle au modèle de mouvement (modèle d'évolution du filtre). Cependant, cette supposition n'est pas toujours vérifiée dans la réalité. Un objet peut changer de direction durant l'occultation, ce qui engendrera des erreurs de ré-identification. L'avantage de la direction de mouvement est sa faible complexité. En effet, aucun calcul supplémentaire n'est nécessaire dans la méthode de détection. De plus, la taille mémoire requise par la caractéristique est limitée

à la taille du vecteur d'état qui dans notre cas se compose de 4 éléments. L'utilisation de l'apparence ajoute une complexité supplémentaire à la détection. Cette complexité est dépendante du nombre de pixels constituant les objets et donc de la taille des objets. De plus, la ré-identification est coûteuse quand le nombre d'histogrammes à comparer et le nombre de niveaux par histogramme sont élevés. La complexité liée à la ré-identification est réduite dans notre méthode car cette étape ne s'effectue qu'à la fin de l'occultation. De plus, même avec un faible nombre de niveaux nous pouvons atteindre des performances intéressantes. De ce fait, nous retenons la solution à base de la caractéristique d'histogramme en niveau de gris dont l'implémentation optimisée sur une caméra intelligente embarquée est décrite dans le prochain chapitre.

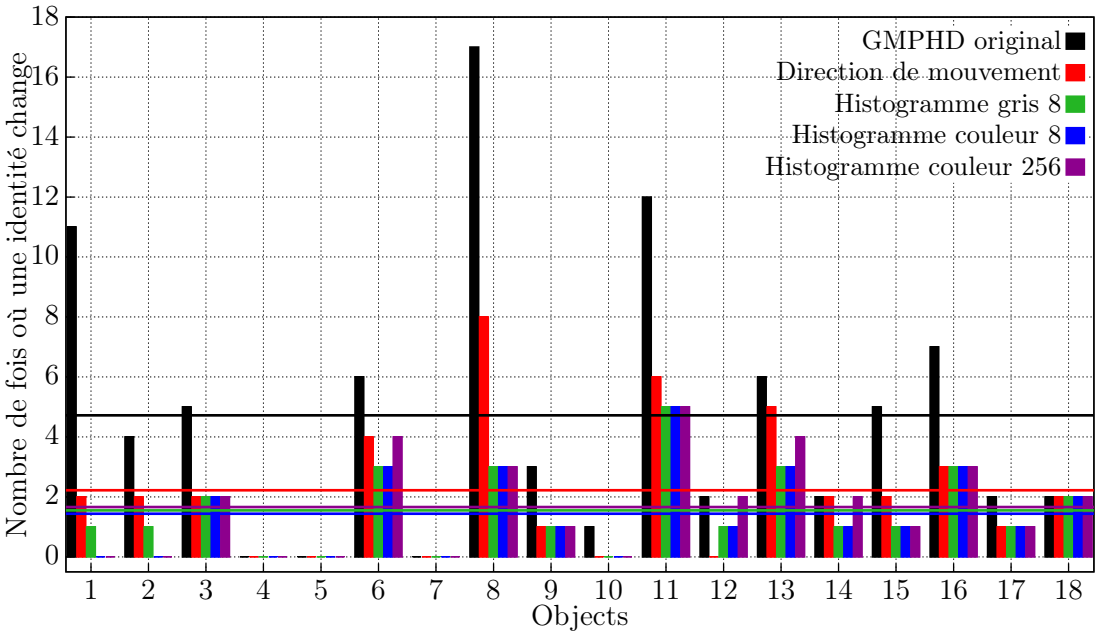

FIGURE 5.10 – Nombre de fois où l'identité d'un objet change dans la séquence PETS2009. Histogramme couleur 8 niveaux : histogramme RGB avec 8 niveaux par canal. Histogramme couleur 256 niveaux : histogramme RGB avec 256 niveaux par canal. Histogramme gris 8 : histogramme en niveau de gris sur 8 niveaux.

Quelques exemples d'images des résultats de suivi sont fournis dans la figure 5.11. La mesure des performances globales du système de suivi et le temps d'exécution de toute la chaîne algorithmique sont présentés dans le prochain chapitre.

5.4 Conclusion

Nous avons développé, dans ce chapitre, une méthode de résolution d'occultation pour le filtre GMPHD. Un premier module est inséré après la phase de prédiction afin de détecter les occultations susceptibles de se produire. Une fois qu'une occultation est détectée, les identités et les caractéristiques des objets en occultation sont sauvegardées dans un objet composé. À la fin de l'occultation, qui est détectée par le module inséré après la phase de correction, les objets sont ré-identifiés à base des caractéristiques sauvegardées. La

particularité de cette approche est qu'elle accepte tout type de caractéristiques. De plus, comme elle n'utilise pas d'historique, la complexité de traitement est uniquement liée à la caractéristique exploitée. Nous avons démontré le fonctionnement de cette méthode en utilisant l'histogramme couleur et la direction de mouvement comme caractéristiques, sur deux bases de données. Les résultats obtenus montrent une amélioration considérable sur l'estimation de nombre d'objets et sur la cohérence des identités des objets. Par rapport aux méthodes de l'état de l'art, notre approche ne nécessite pas d'historique ni de calibration de caméra et peut fonctionner quelles que soient les caractéristiques décrivant les objets. Dans le prochain chapitre nous évaluons les performances de la chaîne complète de suivi intégrant la résolution d'occultation avec la caractéristique d'apparence.

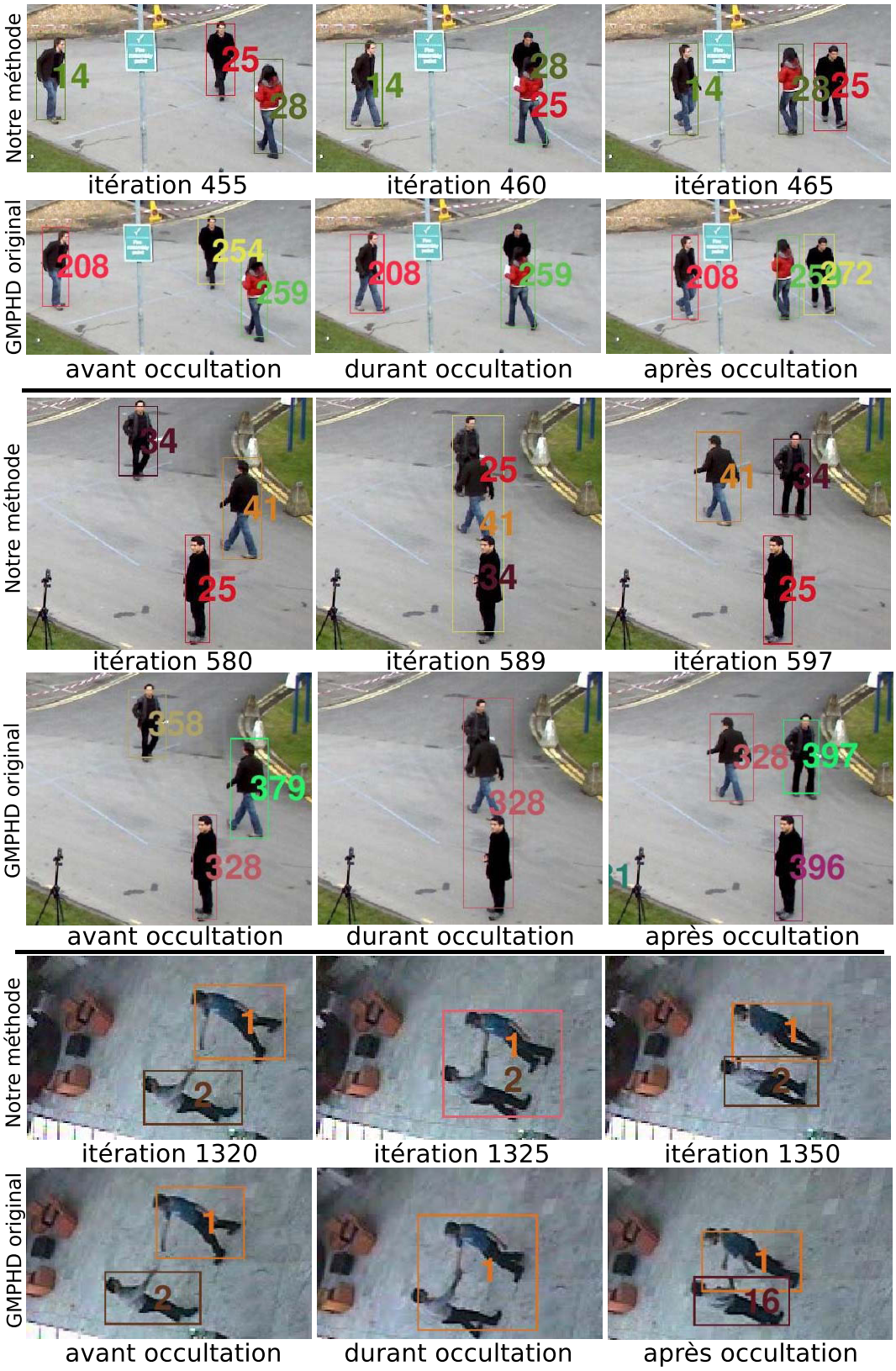

FIGURE 5.11 – Exemples d'images résultantes du suivi.

Chapitre 6

Portage et évaluation de la chaîne de suivi multi-objet

Sommaire

Dans ce chapitre, nous décrivons l'implémentation optimisée de la chaîne de suivi complète, composée du détecteur développé dans le chapitre 3 et du filtre GMPHD amélioré qui traite les occultations entre objets, développé dans le chapitre 5. Suite aux résultats obtenus dans le chapitre 5, nous avons opté pour l'utilisation de la caractéristique d'histogramme en niveau de gris pour la ré-identification des objets à la fin d'occultation. Par la suite, les performances de la chaîne sont évaluées et comparées à quelques méthodes de l'état de l'art. Finalement, des mesures de performances sur une caméra intelligente embarquée sont effectuées.

Dans la première section, les travaux s'intéressant au suivi d'objets sont présentés en insistant sur les travaux effectués sur des caméras intelligentes embarquées. Dans la deuxième section, nous présentons la caméra intelligente embarquée utilisée. La troisième section présente la chaîne complète et explique les optimisations effectuées sur la chaîne algorithmique pour diminuer la quantité de mémoire et de calcul afin d'assurer un traitement en temps réel sur la caméra intelligente. La dernière section du chapitre compare les performances, qualité de suivi et temps d'exécution, de notre chaîne algorithmique optimisée avec des méthodes de l'état de l'art.

6.1 Travaux existants

Les travaux de recherche en suivi multi-objet peuvent être regroupés en deux catégories selon les performances visées : qualité du suivi ou portage sur des caméras intelligentes avec un traitement embarqué. Les travaux s'intéressant à améliorer la qualité du suivi sont souvent basés sur des algorithmes complexes et utilisent des architectures de calcul de type ordinateur de bureau. Ces méthodes présentent souvent les meilleurs résultats de

suivi. De l'autre côté, les travaux visant le portage ou le développement de solutions de suivi multi-objet pour des architectures embarquées de vision s'intéressent à la consommation énergétique des cartes utilisées et au temps d'exécution des algorithmes mais peu de résultats quantifiés sur la qualité de suivi sont présentés. Plus les architectures sont contraintes, plus les algorithmes utilisés sont simples. Dans cette section, nous présentons quelques-uns de ces travaux.

La grande partie des travaux de suivi d'objets en utilisant des caméras intelligentes avec un traitement embarqué s'intéressent au suivi d'un seul objet. Dans [Quaritsch et al., 2007], un seul objet est suivi dans un réseau de caméras intelligentes. Chaque caméra intelligente est composée d'un capteur d'image couleur Eastman Kodak LM9628, deux cartes de traitement Network Video Development Kit (NVDK) de chez ATEME et un processeur réseau pour la communication. Chaque carte NVDK est composée d'un DSP (Digital Signal Processor) TMS320C6416 fonctionannt à 600 MHz avec 264 Mo de mémoire. La chaîne algorithmique de suivi est basée sur l'algorithme CAMSHIFT (section 1.2.3) optimisé pour le DSP. Grâce à la puissance de l'unité de traitement, le temps de traitement d'une image de résolution 352×288 est 10 ms pour l'initialisation de l'histogramme de l'objet et moins de 1 ms pour le suivi. La taille mémoire requise est inférieure de 310 ko. Aucun résultat sur la qualité de suivi n'est fourni. Le traitement temps réel est largement atteint sur la plate-forme embarquée. Étendre le travail au suivi multi-objet avec l'algorithme CAMSHIFT nécessitera une méthode de détection afin d'initialiser les nouveaux objets et une méthode de résolution d'occultation pour assurer la cohérence des identités. Chaque objet est suivi par un CAMSHIFT indépendant comme dans [Hidayatullah and Konik, 2011]. Dans ce cas, le temps de traitement ainsi que le besoin en mémoire vont changer et le traitement en temps réel n'est pas garanti.

Dans [Fleck and Strasser, 2005], le suivi d'un seul objet est effectué à base du filtre à particules utilisant l'histogramme couleur. Le principe de ce filtre est de générer des fenêtres centrées autour des particules prédites et de déterminer l'histogramme de chaque fenêtre. Dans le calcul d'histogramme, la contribution d'un pixel à l'histogramme est pondérée par sa distance spatiale par rapport au centre de la fenêtre. La mise à jour des poids des particules est basée sur la ressemblance de son histogramme à celui de l'objet. La caméra intelligente utilisée est une mvBlueLYNX 420CX de Matrix Vision. Elle est composée d'un capteur d'image CCD d'une résolution 640×480, d'un FPGA Xilinx Spartan-IIE et d'un processeur PowerPC avec FPU fonctionnant à 200 MHz dont le rôle est d'exécuter un Linux embarqué. Cette caméra possède 32 Mo de memoire SDRAM, 32 Mo de mémoire NOR-FLASH et 4 Mo de mémoire NAND-FLASH et elle consomme 7 W. Les résultats obtenus, sous forme d'images de sortie de l'algorithme de suivi, montrent que l'objet est bien suivi au cours de temps et qu'une cadence d'image de 15 fps est atteinte pour la résolution du capteur. L'inconvénient de la méthode est que l'objet est représenté par un ensemble de points (les particules) mais aucune délimitation de l'objet n'est effectuée. De plus, l'extension de la méthode pour le suivi multi-objet va être coûteuse en calcul. En effet, pour la même architecture de caméra intelligente, la cadence d'image va diminuer quand le nombre d'objets augmente.

Peu de travaux se sont intéressés au suivi de plusieurs objets en utilisant des caméras intelligentes embarquées. Dans [Litzenberger et al., 2006], une architecture de vision composée d'un capteur d'évènements relié, via une mémoire FIFO (First-In First-Out), à un

DSP Blackfin BF537 dont la fréquence d'horloge peut atteindre 600 MHz. La mémoire interne est de 128 ko et la mémoire externe est de 32 Mo. L'intérêt du capteur événementiel est sa capacité à coder dans chaque pixel le changement de l'intensité de lumière au lieu de la luminosité de la scène, ce qui permet une soustraction de deux images successives sans aucun traitement. La transmission de capteur au DSP s'effectue seulement quand des évènements sont détectés, ce qui permet de diminuer la consommation de la caméra intelligente. L'algorithme de suivi est basé sur la segmentation en cluster de l'information fournie par le capteur. Chaque cluster représente un objet. Chaque pixel reçu est testé pour savoir s'il appartient aux clusters existants. Dans ce cas, la position du cluster est mise à jour. Sinon un nouveau cluster est créé. Cet algorithme est appliqué pour le suivi et l'estimation des vitesses de véhicules et de piétons. L'avantage de cet algorithme est qu'il traite les pixels à la volée, ce qui assure un faible besoin en mémoire. Le traitement en temps réel est assuré et les résultats obtenus, sous forme d'images, montrent un bon suivi. Cependant, le scénario du test est très simple car la caméra est placée en vue de haut et aucune occultation ne se produit entre les objets.

Dans [Wang et al., 2010], une méthode de suivi multi-objet traitant les occultations entre objets a été développée pour fonctionner en temps réel dans un réseau de caméras intelligentes embarquées. Chaque caméra intelligente effectue son propre suivi d'objet et échange des informations avec les caméras observant le même objet afin de rectifier les erreurs d'identités. Nous ne nous intéressons qu'à la description de suivi effectué au sein d'une caméra. La chaîne algorithmique est la même que celle présentée dans [Velipasalar et al., 2008]. Cette chaîne est composée d'un détecteur à base de la soustraction de fond utilisant l'algorithme Light-Weight [Casares et al., 2010] et d'un module de suivi basé sur l'intersection des boîtes englobantes et la correspondance d'histogrammes. Chaque piste est décrite avec l'histogramme et la boîte délimitant l'objet. Si la boîte de la piste est en intersection avec une boîte fournie pas le détecteur, la correspondance entre leur histogramme est calculée en utilisant le coefficient de Bhattacharyya. La piste est associée à l'observation dont le coefficient de Bhattacharyya est le plus élevé. L'histogramme de l'objet est mis à jour quand le coefficient est supérieur à un seuil prédéfini. Si plusieurs pistes sont associées à une seule détection, elles sont considérées en occultation et leur histogramme n'est pas mis à jour. Une fois l'occultation terminée, les deux pistes peuvent être ré-identifiées en se basant sur l'histogramme. Cette méthode a été implémentée dans la caméra intelligente embarquée CITRIC (table 1.1). Pour des résolutions 320×240, le temps de traitement sur la caméra embarquée est entre 87 ms à 99 ms selon le nombre d'objets et la taille des objets dans l'image. Aucun résultat quantitatif concernant la qualité de suivi n'est fourni dans [Wang et al., 2010]. Les résultats obtenus dans [Velipasalar et al., 2008] montrent que la méthode permet de résoudre correctement 75% des occultations se produisant dans la base de donnée PETS2001. Dans [Casares and Velipasalar, 2010], la même méthode, mais sans tenir compte des occultations, a été optimisée. En effet, les auteurs ont utilisé l'information fournie par l'algorithme de suivi pour limiter la région où s'effectue la détection. Ainsi, la vitesse d'exécution passe de 47.2 ms à 38.5 ms quand trois objets sont suivis. La consommation en énergie de la caméra est réduite de 8.5% grâce à cette méthode.

Comme les travaux de suivi multi-objet sur des caméras intelligentes embarquées ne présentent pas suffisamment de résultats quantifiés pour la qualité de suivi, nous nous

comparons aux méthodes de l'état de l'art utilisant des algorithmes performants pour des machines de calcul de haute performance. Dans [Poiesi and Cavallaro, 2015], pour le suivi, les auteurs utilisent l'algorithme Hongrois pour l'association des observations successives afin de construire un ensemble de pistes courtes. À partir de l'ensemble des pistes courtes générées $\mathfrak{T}$ au cours d'un intervalle de temps, les pistes globales (pistes des cibles) $\mathcal{T} = \{T_a\}_{a=1}^{A}$ (pour $a > b$, la piste T_b commence avant la piste T_a) sont déterminées en maximisant la probabilité $p = [p(T_1|\mathfrak{T})p(T_2|\mathfrak{T}\backslash T_1)\dots p(T_A|\mathfrak{T}\backslash T_A, \dots, T_a, \dots, T_1)]$ avec la méthode Greedy Graph Based (GGB). Le problème est formulé sous forme de graphe où chaque piste courte représente un noeud avec un père et un fils. Deux pistes courtes t_b et $t_{b'}$ sont liées avec une probabilité d'association qui diminue exponentiellement avec la distance entre l'état initial de la piste $t_{b'}$ et l'état final de la piste t_b, et la différence de temps entre la fin de la piste t_b et le début de la piste $t_{b'}$. La solution du problème est basée sur une méthode itérative, qui à chaque itération relie deux pistes courtes. Le test de la méthode est effectué sur des bases de données d'insectes en mouvement et sur des bases de données contenant des humains. Les résultats obtenus montrent des performances équivalentes aux méthodes de l'état de l'art quand la fenêtre temporelle contient 25 images et seulement la position est utilisée comme caractéristique (les informations de vitesse et de couleur ne sont pas utilisées). Les résultats obtenus avec cette méthode sont comparés aux résultats de notre méthode dans la section 6.4. Le temps d'exécution de la méthode quand la détection n'est pas incluse est de 15 fps pour une implémentation en Matlab sur une machine Core i5 fonctionnant à 2.4 GHz, en utilisant une base de données contenant en moyenne 31 cibles par image.

Dans [Conte et al., 2010], un suivi multi-objet avec une gestion d'occultations a été développé. La chaîne est composée d'un module de détection et d'un module de suivi. La détection utilise pour la soustraction de fond l'algorithme First-Order low pass filter avec un seuil de segmentation adaptatif avec l'intensité lumineuse moyenne de l'image. Une analyse en composante connectée permet d'extraire les régions en mouvement de l'image. Le problème de détection de plusieurs fragments appartenant à un même objet est résolu en fusionnant les détections proches tout en respectant un rectangle modèle prédéfini représentant l'objet. Les régions détectées sont filtrées suivant leur taille et leur surface. Une élimination d'ombre est effectuée sur le résultat du filtrage. La dernière étape de détection consiste à éliminer la réflexion de lumière, généralement causée par la nature du sol, en utilisant les propriétés chromatiques de la réflexion. Les résultats de détection constituent les observations du module de suivi. Dans le module de suivi, les distance entre les caractéristiques des pistes et celles des détections sont calculés et comparées à un seuil. Suivant le nombre de détections associées à une piste et le nombre de pistes associées à une détection, les occultations sont traitées. La caractéristique de chaque objet est un MultiView Appearance Model (modèle d'apparence de plusieurs points de vue). Le nombre de points de vue est fixé à 8 selon l'orientation du vecteur de mouvement de l'objet. Chaque point de vue est décrit avec deux blocs de pixels. Le premier bloc contient l'apparence visuelle de l'objet. Le second bloc contient l'information sur le nombre de fois qu'un pixel du premier bloc est considéré comme appartenant à l'objet en se basant sur le résultat de détection. La méthode est testée sur les séquences de la base de données PETS2009. Les résultats obtenus sont comparés aux résultats de notre méthode dans la section 6.4.

Cet état de l'art montre que les systèmes de suivi multi-objet réalisés sur des caméras intelligentes avec un traitement embarqué utilisent des méthodes de suivi classiques pour des scénarios simples pour la plupart. Ces travaux s'intéressent au temps de traitement sur la carte, au besoin en mémoire et à la consommation en puissance. Les résultats sur la qualité de suivi sont rarement quantifiés. D'un autre côté, les travaux développant des algorithmes de suivi pour des architectures de type PC traitent des scénarios de suivi plus complexes. Ils s'intéressent plus à la qualité de suivi qu'aux besoins en mémoire et en calcul. Dans ces travaux un traitement en temps réel sur un PC est satisfaisant. Dans la prochaine section, nous présentons la caméra intelligente utilisée pour nos expérimentations.

6.2 Caméra intelligente utilisée

La caméra intelligente embarquée que nous avons choisie d'utiliser se compose de la caméra RaspiCam utilisée comme capteur et de la carte Raspberry-Pi modèle B avec 512 Mo de mémoire utilisée comme unité de traitement. La plate-forme est indiquée dans la figure 6.1. La caméra RaspiCam peut acquérir des images de résolution allant jusqu'à 2592×1944 avec une cadence de 15 fps dans l'espace couleur YUV. La capture d'images de faible résolution permet une cadence plus élevée. La carte Raspberry Pi version B est composée du circuit Broadcom BCM2835 qui contient un processeur ARM1176JZF-S avec FPU fonctionnant à 700 MHz et un GPU Broadcom VideoCore4. Différents périphériques peuvent être connectés à la carte à travers les ports USB ou les entrées-sorties pour utilisation générale (general purpose input-output - GPIO) : wifi USB, clavier, souris, *etc.* La carte possède une sortie HDMI pour pouvoir connecter un écran. La caméra RaspiCam est connectée à la carte en utilisant le port Camera Serial Interface (CSI). La carte exécute un système d'exploitation Raspbian (dérivé de Linux) qui est installé sur une carte mémoire SD. Grâce aux librairies du système d'exploitation et aux compilateurs, des codes en C/C++ peuvent être compilés et exécutés sur la carte.

L'accès à la caméra s'effectue avec la librairie userland [1]. Cette librairie capture l'image et la renvoie à la carte sous forme d'une structure contenant les données des pixels. Elle permet aussi de contrôler les différents paramètres de capture comme le contraste, la résolution et la cadence d'images.

6.3 Optimisation de la chaîne algorithmique

La chaîne de suivi multi-objet complète est indiquée dans la figure 6.2. Le module détection est celui présenté dans le chapitre 3 et le module de suivi est celui présenté dans le chapitre 5. Nous rappelons brièvement comment fonctionnent les deux modules et nous expliquons comment s'effectue l'extraction d'histogramme en niveau de gris avec le module de détection.

Le module de détection applique la soustraction de fond sur l'image d'entrée. La soustraction de fond est basée sur l'algorithme Zipfian Sigma-Delta fusionné avec l'information du gradient de l'image d'entrée. Chaque pixel considéré comme appartenant à un objet

1. https ://github.com/raspberrypi/userland.

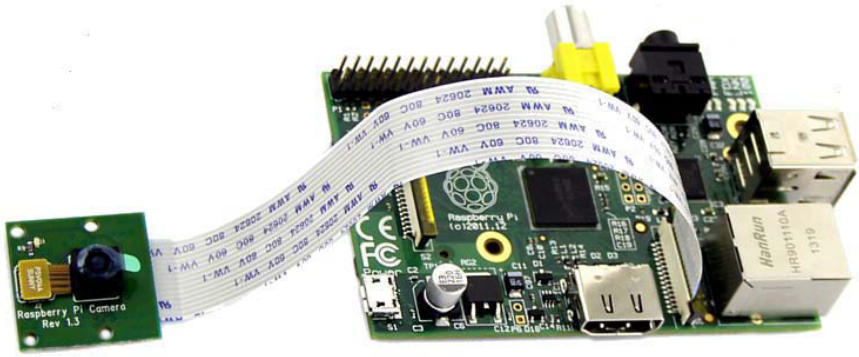

FIGURE 6.1 – Caméra intelligente embarquée composée de la carte Raspberry Pi et de la caméra RapiCam

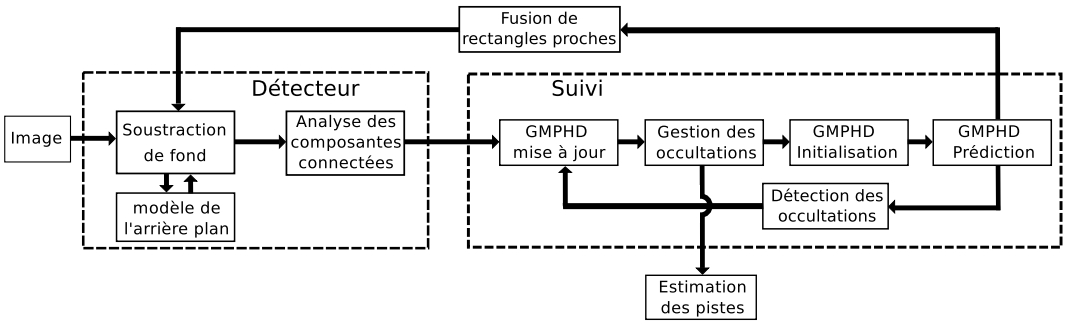

FIGURE 6.2 – Chaîne de suivi multi-objets

est traité à la volée par l'algorithme d'analyse des composantes connectées *Single Pass labeling*. Pour pouvoir extraire l'histogramme des objets détectés, chaque pixel classifié par Zipfian Sigma-Delta (avant d'appliquer la fusion avec le gradient) comme appartenant au premier plan (objet) voit sa valeur en niveau de gris sauvegardée dans une mémoire image. Une fois que les objets sont détectés et les coordonnées des boîtes englobantes sont déterminées. Les histogrammes m-niveaux des objets sont calculés en prenant en compte que les pixels se trouvant à l'intérieur de la boîte délimitant l'objet. Le module de détection ne nécessite pas l'utilisation d'une FPU et une implémentation en virgule fixe est réalisée.

Pour l'implémentation du module de suivi, nous avons optimisé celui présenté dans le chapitre 5. Le vecteur d'état comprend les coordonnées de position (x, y), les coordonnées de la vitesse (v_x, v_y) ainsi que la largeur w et la hauteur h de l'objet, ce qui engendre des matrices de covariance P de taille 6×6 . Chaque composante du mélange Gaussien contient l'histogramme de l'objet qu'elle représente. Afin d'optimiser l'algorithme, nous exploitons l'indépendance statistique des éléments composant le vecteur d'état. La seule dépendance existante est entre les coordonnées de position et les coordonnées de vitesse d'un axe de l'image, i.e. x dépend de v_x et y dépend de v_y. Dans ce cas, la matrice de covariance est une matrice creuse (figure 6.3). De plus, nous imposons que les valeurs des variances des coordonnées de positions et des coordonnées de vitesse soient identiques suivant les axes x et y pour le modèle d'état et le modèle d'initialisation de cibles. Dans ce

$$\begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xv_x} & \sigma_{xv_y} & \sigma_{xw} & \sigma_{xh} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yv_x} & \sigma_{yv_y} & \sigma_{yw} & \sigma_{yh} \\ \sigma_{v_xx} & \sigma_{v_xy} & \sigma_{v_xv_x} & \sigma_{v_xv_y} & \sigma_{v_xw} & \sigma_{v_xh} \\ \sigma_{v_yx} & \sigma_{v_yy} & \sigma_{v_yv_x} & \sigma_{v_yv_y} & \sigma_{v_yw} & \sigma_{v_yh} \\ \sigma_{wx} & \sigma_{wy} & \sigma_{wv_x} & \sigma_{wv_y} & \sigma_{ww} & \sigma_{wh} \\ \sigma_{hx} & \sigma_{hy} & \sigma_{hv_x} & \sigma_{hv_y} & \sigma_{hw} & \sigma_{hh} \end{pmatrix} \underset{\text{statistique}}{\overset{\text{independance}}{\Longrightarrow}} \begin{pmatrix} \sigma_{xx} & 0 & \sigma_{xv_x} & 0 & 0 & 0 \\ 0 & \sigma_{yy} & 0 & \sigma_{yv_y} & 0 & 0 \\ \sigma_{xv_x} & 0 & \sigma_{v_xv_x} & 0 & 0 & 0 \\ 0 & \sigma_{v_yy} & 0 & \sigma_{v_yv_y} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{ww} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{hh} \end{pmatrix}$$

FIGURE 6.3 – Impact de l'indépendance statistique sur les éléments du vecteur d'état

cas, les variances $\sigma_{xx} = \sigma_{yy}$ et $\sigma_{v_xv_x} = \sigma_{v_yv_y}$ et les covariances $\sigma_{xv_x} = \sigma_{v_xx} = \sigma_{v_yy} = \sigma_{yv_y}$. Ainsi, les seuls éléments non nuls dans la matrice de covariance sont : σ_{xx}, σ_{xv_x}, $\sigma_{v_xv_x}$, σ_{ww}, σ_{hh}. Dans ce cas, le nombre d'éléments représentant une matrice de covariance est réduit de 36 à 5. L'indépendance entre les éléments du vecteur d'état permet d'exploser la formulation matricielle du filtre GMPHD en un système linéaire d'équations avec peu d'équations (seuls les éléments non nuls sont propagés). Cette nouvelle formulation permet de diminuer le nombre d'opérations et d'éviter les opérations d'inversion de matrice et le calcul de déterminant lors de la mise à jour des poids des Gaussiennes. Comme la caméra intelligente utilisée possède une FPU, l'implémentation de l'algorithme de suivi est réalisée en virgule flottante.

Dans le but d'améliorer le temps d'exécution de tout le système, nous exploitons le résultat de suivi pour prédire les régions de l'image où les objets sont susceptibles d'être détectés. Ainsi, les moyennes (vecteurs d'état) des composantes du mélange Gaussien après prédiction ainsi que leurs écarts-types sur la hauteur et la largeur des objets sont utilisés pour définir un rectangle de l'image dans lequel s'effectue la détection. Afin d'éviter plusieurs détections dans la même région de l'image, les rectangles prédits qui se chevauchent sont fusionnés pour construire un seul rectangle où s'effectue la détection. Une détection s'effectue sur l'image entière quand aucun objet n'est présent dans la scène ou chaque N images dans le cas contraire.

6.4 Résultats

L'évaluation de la chaîne est effectuée en deux étapes. Dans la première étape nous nous intéressons à la qualité de suivi en se comparant aux méthodes de l'état de l'art [Poiesi and Cavallaro, 2015, Conte et al., 2010] dont les fonctionnements sont décrits dans la section 6.1, sur la séquence vidéo PETS2009-S2L1. La deuxième étape consiste à mesurer les temps d'exécution de la chaîne avec les bases de données utilisées sur un ordinateur personnel ayant comme processeur un core 2 duo fonctionnant à 3 GHz et sur la carte Raspberry Pi constituant l'unité de traitement la caméra intelligente. Par la suite, la cadence d'images est mesurée quand la caméra intelligente est utilisée pour le suivi, i.e. les images sont capturées avec la RaspiCam. Tous les traitements s'effectuent sur des images en niveau de gris capturées à la cadence maximale correspondant à la résolution utilisée.

Les paramètres de la chaîne incluant les seuils du détecteur et les paramètres du filtre GMPHD (comme les covariances du bruit du modèle d'état, la probabilité du survie, la densité des fausses alarmes, ...) sont fixés expérimentalement pour assurer un bon suivi selon la séquence utilisée. Le modèle d'état du filtre GMPHD, dans les expérimentations, considère l'évolution de tous les paramètres du vecteur d'état, contrairement à ce qui est présenté dans le chapitre 5. En effet, les écarts-types de la hauteur et de la largeur des objets sont utilisés pour limiter la région de l'image où s'effectue la détection en se basant sur les états prédits avec GMPHD. Le vecteur d'observation contient donc les coordonnées de position du centre de gravité, la hauteur et la largeur de l'objet. Pour la mise à jour des poids des Gaussiennes du mélange, seules les coordonnées de position sont considérées et la largeur et la hauteur ne sont pas utilisées. La largeur et la hauteur sont des paramètres moins stables quand la soustraction de fond est utilisée et souvent sont identiques pour tous les objets quand des personnes sont considérées.

6.4.1 Qualité de suivi

Pour la mesure de la qualité de la chaîne de suivi sur la séquence PETS20019-S2L1 [2], nous avons construit à partir des images de la séquence une image d'arrière plan qui ne contient aucun objet (figure 6.4). Cette image est utilisée pour initialiser le modèle d'arrière plan de l'algorithme Zipfian sigma-delta. Les images de 1 à 795 sont utilisées pour l'évaluation. Notre approche est aussi testée sur la séquence MeetSplit3rdGuy de CAVIAR [3]. Pour cette séquence, le suivi est considéré de l'image 150 à l'image 900 de la séquence. L'initialisation de l'arrière plan s'effectue avec les images 1 à 150. Dans les deux base de données, la détection est effectuée sur toute l'image avec une cadence d'une détection chaque 2 images successives. Le reste du temps la détection s'effectue sur les régions prédites par la filtre GMPHD.

La vérité terrain utilisée pour l'évaluation de la qualité de suivi sur la séquence PETS2009 est celle fournie par les auteurs [Poiesi and Cavallaro, 2015] [4]. La vérité terrain pour la séquence CAVIAR est celle fournie dans le site de la base de données.

Les métriques utilisées pour l'évaluation de la qualité de suivi du système sont : Précision (Pr) (équation 6.1), Rappel (Re) (équation 6.2), Multiple Object Tracking Precision (MOTP) (équation 6.3), Multiple Object Tracking Accurancy (MOTA) (équation 6.4) et ID switch (IDS). Ces métriques sont basées sur les faux positifs (FP) et les faux négatifs (FN). FP se produisent quand l'algorithme suit des objets qui ne sont pas présents dans la scène. FN sont les objets qui existent en réalité mais qui ne sont pas suivis par l'algorithme. MOTP traduit la précision de l'algorithme de suivi à délimiter les objets en mesurant le taux d'intersection des rectangles estimés avec ceux de la vérité terrain. MOTA traduit la fidélité du suivi en considérant les FP, FN et IDS. IDS représente le nombre de fois que les identités des objets changent au cours de suivi.

$$Pr = \frac{TP}{TP + FP} \tag{6.1}$$

2. http ://www.cvg.reading.ac.uk/PETS2009/a.html
3. http ://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
4. http ://www.eecs.qmul.ac.uk/∼andrea/thdt.html

FIGURE 6.4 – L'image utilisée pour l'initialisation de l'arrière plan pour la séquence PETS2009-S2L1

$$Re = \frac{TP}{TP + FN} \tag{6.2}$$

$$MOTP = \frac{\sum_{k=1}^{N_{frame}} \sum_{i=1}^{N_{mapped}(k)} \frac{|G_i(k) \bigcap T_i(k)|}{|G_i(k) \bigcup T_i(k)|}}{\sum_{k=1}^{N_{frame}} N_{mapped}(k)} \tag{6.3}$$

$$MOTA = 1 - \frac{\sum_{k=1}^{N_{frame}} FP(k) + FN(k) + IDS(k)}{\sum_{k=1}^{N_{frame}} N_g(k)} \tag{6.4}$$

$N_g(k) = TP(k) + FN(k)$ est le nombre réel d'objets dans l'image à l'itération k. $N_{mapped}(k)$ est le nombre d'objets estimés qui sont associés à des objets de la vérité terrain à l'itération k . G_i est la boîte englobant l'objet i dans la vérité terrain. T_i est la boîte fournie par l'algorithme de suivi qui englobe l'objet i.

Le calcul de ces métriques nécessite une association entre la vérité terrain et le résultat de suivi fourni par la chaîne. Le critère d'association est basée sur l'intersection des boîtes estimées et celles de la vérité terrain. Le coût d'association de deux boîtes est la surface du rectangle de la vérité terrain sur le taux d'intersection des deux rectangles. Plus l'intersection est importante, plus le coût est faible. Quand les boîtes ne s'intersectent pas, le coût est fixé à l'infini. À partir de ces coûts, une matrice de coût est construite et utilisée comme entrée de l'algorithme Hongrois qui permet de déterminer l'association optimale entre la vérité terrain et l'estimation de l'algorithme de suivi. Dans notre méthode, un rectangle peut avoir plusieurs identités quand des occultations se produisent. Plusieurs rectangles superposés avec des identités différentes sont considérés dans l'association. Sans l'ajout d'une contrainte, le coût d'association sera le même pour tous les rectangles se superposant, ce qui engendre une association aléatoire avec l'algorithme Hongrois et augmente la valeur IDS. Afin d'éviter ce problème, nous imposons la contrainte de favoriser l'association qui ne produit pas de changement d'identité. Cela se traduit par l'ajout d'un coût ε au coût d'une association dans la matrice de coût quand cette association produit un changement d'identité.

Le tableau 6.1 montre l'application de ces métriques aux résultats de suivi pour la séquence PETS2009-S2L1. Nos résultats sont comparés aux résultats de [Poiesi and Cavallaro, 2015] en utilisant les mêmes métriques et la même vérité terrain. Notre méthode est aussi comparée à la méthode présentée dans [Conte et al., 2010] dont les valeurs des métriques sont celles fournies dans le papier et la vérité terrain est définie manuellement par les auteurs.

TABLE 6.1 – Résultats du suivi multi-objet. MOTA, MOTP, Pr and Re : une valeur élevée traduit une meilleure qualité, IDS : une valeur faible traduit une bonne qualité.

Séquence	Méthode	Pr	Re	MOTA	MOTP	IDS
PETS2009-S2L1	Notre méthode	0.92	0.96	0.86	0.59	78
	[Poiesi and Cavallaro, 2015]	0.95	0.97	0.91	0.87	46
	[Conte et al., 2010]	-	-	0.83	0.64	-
CAVIAR MeetSplit3rdGuy	Notre méthode	0.91	0.78	0.70	0.50	3

Sur la séquence PETS2009, les métriques Pr et Re obtenues avec notre méthode sont proches, mais inférieures, à la méthode [Poiesi and Cavallaro, 2015]. La valeur de la métrique MOTA de la méthode [Poiesi and Cavallaro, 2015] surpasse notre méthode ce qui se traduit par une meilleure fidélité dans le suivi. Cette différence dans la valeur de MOTA est liée à l'écart entre les IDS obtenus. Étant donnée que la valeur IDS résultante de notre méthode est supérieure à celle de la méthode [Poiesi and Cavallaro, 2015], la valeur MOTA diminue considérablement. La différence dans les valeurs IDS n'est pas seulement liée à la qualité de l'algorithme de suivi, mais aussi à la qualité de détection. En effet, plus le détecteur est robuste aux occultations, plus la valeur IDS diminue. Dans [Poiesi and Cavallaro, 2015], les détections fournies en entrée à l'algorithme de suivi sont issues du détecteur utilisé dans [Yang and Nevatia, 2012] [5]. Ce détecteur se base sur une méthode d'apprentissage qui arrive à détecter des objets même quand une occultation partielle se produit. Avoir des détections précises permet d'augmenter la valeur de la métrique MOTP. La valeur MOTP obtenue avec notre méthode est faible à cause de l'approche de fusion regroupant les objets occultés dans le même rectangle.

En se comparant à la méthode [Conte et al., 2010], notre méthode fournit des meilleurs résultats par rapport à la métrique MOTA. La métrique MOTP traduisant la précision des rectangles issus du suivi reste faible comparée à celle de la méthode [Conte et al., 2010]. En effet, les heuristiques ajoutées à la soustraction de fond dans [Conte et al., 2010] permet d'améliorer la précision des boîtes détectées ce qui se traduit par l'augmentation de la valeur MOTP.

Dans [Ellis and Ferryman, 2010], plusieurs algorithmes de suivi ont été comparés en utilisant la séquence PETS2009-S2L1 (figure 6.5). Malheureusement, la vérité terrain qu'ils utilisent pour la comparaison n'est pas disponible. L'utilisation des vérités terrain différentes ne permet pas une comparaison correcte. Cependant, si nous considérons que la

5. http ://iris.usc.edu/people/yangbo/downloads.html

différence des vérités terrain n'influence pas les métriques MOTA et MOTP, la comparaison de notre méthode à ces méthodes peut s'effectuer. Dans ce cas, notre méthode surpasse 11 sur les 12 méthodes présentées pour la métrique MOTA. La valeur MOTP obtenue avec notre méthode est proche de la meilleure valeur MOTP des méthodes.

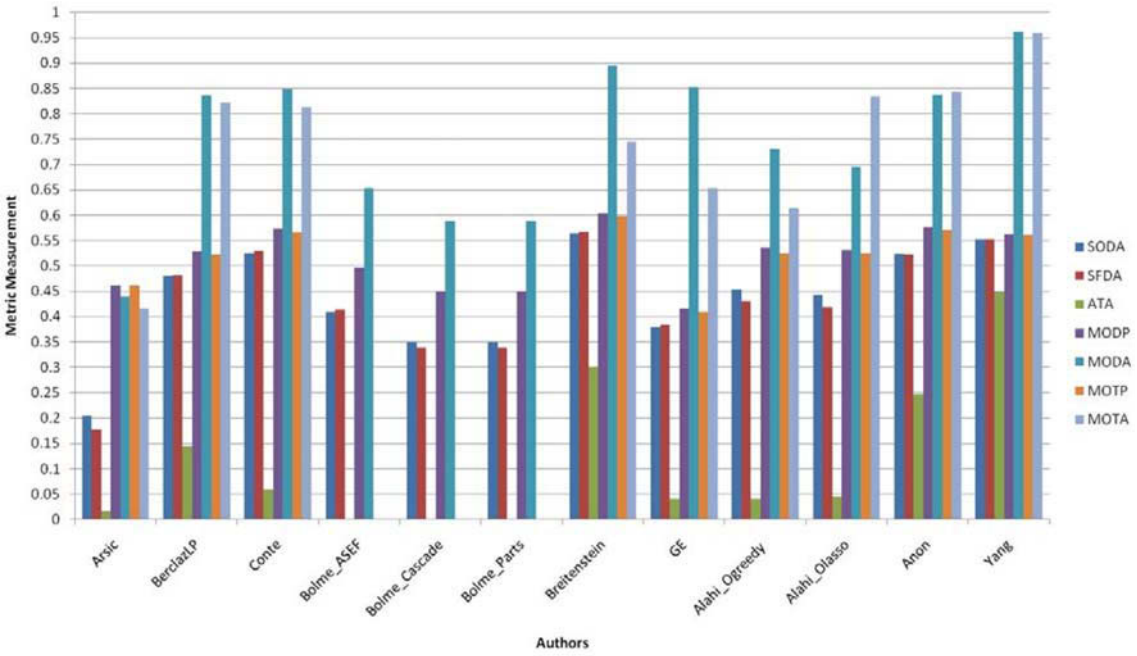

FIGURE 6.5 – Comparaison de différents algorithmes de suivi multi-objet sur la séquence PETS2009-S2L1 [Ellis and Ferryman, 2010]. Seules les valeurs MOTA et MOTP nous intéressent.

Les résultats obtenus pour la base de données CAVIAR montrent un faible changement dans l'identité des objets. Le peu de changement dans les identités n'est pas causé par des occultations mais par les non détections successives d'objets à cause de leur faible taille quand ils rentrent dans la scène. Des échantillons d'images montrant les résultats de suivi sont donnés dans les figures 6.6.

6.4.2 Vitesse d'exécution

Le tableau 6.2 donne les cadences d'images obtenues en exécutant la chaîne optimisée sur un ordinateur de bureau et sur la carte Raspberry Pi avec les séquences PETS2009 et CAVIAR. L'implémentation sur la carte Raspberry Pi n'exploite pas le GPU présent dans le circuit Broadcom BCM2835. Dans cette expérimentation, la module RaspiCam de la caméra intelligente n'est pas utilisé. La lecture des images des bases de données s'effectue à partir de la mémoire.

Les résultats montrent que la cadence de traitement est élevée sur l'ordinateur de bureau. Cette cadence d'image dépend de la résolution d'image d'entrée. La mesure de la proportion du temps de calcul de chaque module de la chaîne en utilisant l'outil *callgrind* [Nethercote et al., 2006], montre que détection occupe 99% de temps de calcul. La

proportion de la détection montre que le temps d'exécution est faiblement lié au nombre d'objets à suivre.

L'exécution de la chaîne sur la carte Raspberry Pi satisfait les contraintes de temps réel même pour la résolution de la séquence PETS2009 qui est de 768×576.

TABLE 6.2 – Cadence d'image issue de l'exécution de la chaîne de suivi multi-objet sur les séquences PETS2009 et CAVIAR sur un ordinateur personnel et sur la carte Raspberry Pi

Dataset	Image resolution	PC (fps)	RaspberryPi (fps)
PETS2009	768 x 576	300	14
CAVIAR	384 x 288	600	40

Le cadence d'exécution de la méthode présentée dans [Poiesi and Cavallaro, 2015], sans considérant la détection, est de 15 fps pour une implémentation en Matlab sur une machine Core i5 fonctionnant à 2.4 GHz. L'algorithme de suivi est exécuté sur une base de données contenant en moyenne 31 cibles par image. En effet, le temps d'exécution de cette méthode diminue avec la diminution de nombre de cibles. Pour un nombre de cibles s'élevant à 9 au maximum dans la scène, notre méthode fournit la même performance d'exécution sur une calculateur embarquée avec le module de détection inclus. Dans [Conte et al., 2010] aucune vitesse d'exécution n'est fournie.

6.4.3 Exécution de la chaîne sur la caméra intelligente

Dans cette expérimentation, la chaîne de suivi multi-objet est exécutée sur la caméra intelligente. En effet, les images son capturée par la RaspiCam. L'objectif de cette expérimentation est de mesurer la vitesse d'exécution sur une caméra intelligente réelle où la capture de l'image a un coût et où la bande passante entre l'imageur et l'unité de traitement joue un rôle. La mesure de la cadence d'image s'est effectuée avec la librairie *time* qui compte le nombre d'images traitées pendant un laps de temps. Nous avons effectué une expérimentation pour le suivi de personne dans le laboratoire.

Le temps d'exécution est de 30 fps pour une résolution de 320×240 et de 15 fps pour une résolution 640×480. Le temps traitement d'une image de résolution 320×240 sur la caméra intelligente CITRIC, avec la méthode développée dans [Wang et al., 2010] varie de 87 ms à 99 ms selon le nombre d'objets et la taille des objets. La cadence d'image résultante dans ce cas varie entre 10 et 11 fps pour un nombre d'objets maximum de 3. La vitesse d'exécution de notre chaîne de traitement surpasse celle de la méthode [Wang et al., 2010], même quand une résolution plus élevée est considérée dans notre méthode.

La méthode développée dans [Casares and Velipasalar, 2010] et qui ne tient pas compte des occultations, assure une cadence d'image variable entre 25 fps quand 3 objets sont considérés, et 50 fps quand un seul objet est suivi. La résolution des images utilisées n'est pas fournie par les auteurs.

La méthode que nous avons proposée assure des performances équivalentes mais avec l'avantage que le nombre d'objets n'influence pas beaucoup sur le temps de traitement. Ce résultat démontre la fonctionnalité de notre approche sur une vraie caméra intelligente dont le prix s'élève à quelque euros et dont la consommation en puissance est modérée.

Le traitement en temps réel dans un système de suivi peut se caractériser avec une dizaine d'images par seconde selon la vitesse des objets et l'emplacement de la caméra. Ainsi, les résultats obtenus sont prometteurs pour que la chaîne puisse atteindre le temps réel sur des caméras intelligentes disposant de moins de ressources de calcul. Les faibles ressources de calcul permettent de diminuer le prix et la consommation des caméras intelligentes ce qui est bénéfique quand des réseaux de caméras sont déployés.

6.5 Conclusion

Nous avons présenté les résultats de la chaîne complète de suivi multi-objet de point de vue de la qualité de suivi et de la vitesse d'exécution. Les résultats sur la qualité de suivi ne sont pas loins de ceux des méthodes de l'état de l'art qui s'intéressent à la qualité des algorithmes plus qu'à leur portage.

La cadence d'image est mesurée sur un ordinateur personnel et sur la caméra intelligente composée de la Raspberry Pi version 1 et du module RaspiCam. Les résultats obtenus montrent une cadence élevée sur l'ordinateur personnel qui de l'ordre de 300 fps pour une résolution de 768×576. La cadence d'images sur la caméra intelligente est de 15 fps pour la résolution 640×480 et 30 fps pour une résolution de 320×240. Cette vitesse d'exécution sur la caméra intelligente est équivalente et surpasse dans certains cas les vitesses d'exécution obtenues dans la littérature. De plus, ces résultats sont prometteurs pour un fonctionnement en temps de la chaîne sur des caméras intelligentes avec des ressources de calcul plus faibles.

Nous avons démontré dans ce chapitre la possibilité de suivi multi-objet en temps réel sur une caméra embarquée intelligente de faible coût et de consommation modérée. La consommation et le coût de cette caméra la rend pertinente pour être déployer dans un réseau de caméras. Dans le prochaine chapitre, nous nous intéresserons à l'extension de la méthode pour un réseau de caméras avec des champs de vision non recouvrants.

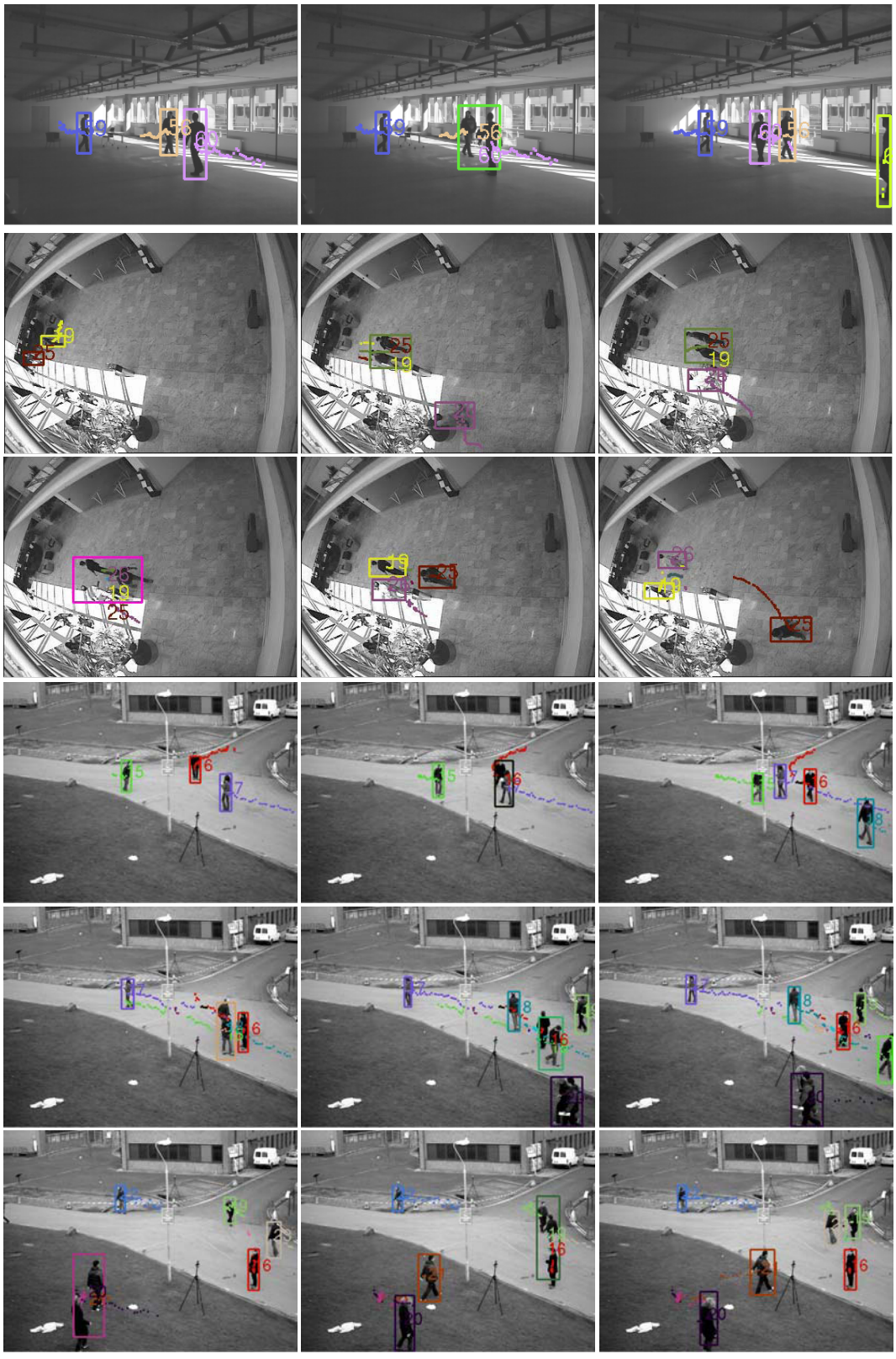

FIGURE 6.6 – Échantillon d'images montrant la qualité de suivi obtenue avec notre méthode sur les bases de données. La première ligne représente le résultat du suivi avec la caméra intelligente. La seconde et la troisième lignes représentent des images de la séquence CAVIAR. Les trois dernières lignes représentent des images de la séquence PETS2009.

Chapitre 7

Suivi multi-objet dans un réseau caméras avec champs non recouvrants

Sommaire

Dans ce chapitre, la chaîne de suivi multi-objet développée dans les chapitres précédents, est étendue au cas de plusieurs caméras avec des champs de vision non recouvrants. Le suivi dans ce type de réseaux de caméras introduit des nouveaux challenges par rapport au suivi mono-caméra. En effet, la continuité spatio-temporelle et visuelle des observations générées par les objets n'est plus assurée. Cette discontinuité est causée par les régions non visibles par les caméras du réseau. La problématique du suivi multi-objet dans ce cas est une problématique de ré-identification d'objets quand ils quittent une caméra pour ré-apparaître dans une autre caméra. Les caméras à champs non recouvrant ne nécessitent pas une grande précision sur la calibration spatiale, comme c'est le cas pour les caméras avec des champs recouvrants. Cependant, d'autres types de calibrations peuvent s'avérer nécessaires, selon les approches utilisées.

Le challenge dans les réseaux de caméras est la consommation énergétique. Des systèmes fonctionnant à base de batteries sont plus favorables pour le déploiement. La consommation énergétique d'un réseau de caméras dépend principalement des architectures électroniques composant les noeuds, de la bande passante et du protocole de communication. L'utilisation des architectures légères comme des caméras intelligentes composées de capteurs et d'unités de traitement à base de processeurs embarqués assurent une faible

consommation et réduit la bande passante nécessaire. En effet, ces architectures effectuent des traitements proches capteur, ce qui permet de réduire l'information communiquée entre les noeuds et donc le besoin en bande passante. Un autre facteur intervenant sur la bande passante du système est la nature du traitement. Un traitement distribué où les communications s'effectuent entre les noeuds voisins consomme moins de bande passante qu'un système centralisé où tous les noeuds communiquent simultanément avec le centre de traitement. Afin de rester dans ce contexte, nous proposons une solution distribuée où le suivi dans chaque caméra est réalisé avec le filtre GMPHD. Notre approche utilise les caractéristiques spatio-temporelle et d'apparence pour ré-identifier les objets. Le résultat de ré-identification est par la suite intégré dans le filtre GMPHD d'une manière distribuée afin de limiter les communications entre les noeuds

Dans la première section du chapitre, les approches existantes de suivi multi-objet dans les réseaux de caméras avec des champs non recouvrants sont présentées en se focalisant sur les travaux proches de la solution proposée. La seconde section explique notre approche. La troisième section détaille le scénario de simulation réalisé pour tester l'approche proposée. Dans quatrième section, les résultats obtenus sont présentés. La dernière section analyse la complexité de l'approche proposée.

7.1 Travaux existants

Plusieurs approches ont été développées pour la ré-identification d'objets se déplaçant dans un réseau de caméras avec des champs de vision non recouvrants. Certaines approches ne nécessitent aucune connaissance a priori sur le réseau de caméras pour assurer la ré-identification [Truong Cong et al., 2009, Alahi et al., 2010, Cong et al., 2010]. La ré-identification, dans ce cas, est réalisée avec des caractéristiques visuelles qui sont robustes aux paramètres variables entre caméras, comme le point de vue, la qualité du capteur, l'éclairage, *etc.* L'inconvénient de ces approches est la complexité calculatoire nécessaire pour l'extraction des caractéristiques ainsi que les besoins en mémoire de ces caractéristiques. Les caractéristiques sont souvent de grandes tailles ce qui implique une bande passante de communication importante. D'autre approches exploitent la connaissance a priori du réseau afin de réaliser une ré-identification correcte tout en utilisant des caractéristiques simples comme, les histogrammes de couleur, le temps de transition, *etc.* Dans la thèse nous nous sommes intéressés à ces approches, car elles permettent de bons résultats avec peu de complexité et de bande passante. Le seul inconvénient est qu'une phase d'apprentissage est souvent nécessaire pour la construction de la connaissance a priori. Cependant, cet apprentissage peut s'effectuer automatiquement [Javed et al., 2003, Javed et al., 2008] ce qui garantit un déploiement facile et flexible du réseau de caméras.

Dans [Huang and Russell, 1997, Kettnaker and Zabih, 1999, Javed et al., 2003, Dick and Brooks, 2005], des modèles probabilistes ont été utilisés pour la ré-identification. L'information d'apparence et l'information spatio-temporelle constituent les caractéristiques des objets. Dans [Chen et al., 2013b] la ré-identification est formulée comme la reconnaissance multi-classe d'objets avec le classifieur Adaboost. L'information a priori consiste en la fonction de transfert de couleur entre caméras, le temps de transition d'un objet entre deux caméras, la probabilité de transition, les régions d'entrées-sorties

d'objets, *etc.* Une formulation probabiliste de suivi multi-caméra s'avère plus pertinente dans notre cas, vu que la méthode de suivi mono-caméra est basée sur le filtre GMPHD qui est une méthode probabiliste. Les méthodes probabilistes proches de notre travail sont étudiées dans le reste de cette section.

Dans [Huang and Russell, 1997] la ré-identification est étudiée dans un contexte de la surveillance routière avec deux caméras. Le modèle probabiliste développé modélise une voiture par sa couleur moyenne, son temps de transition, sa vitesse, sa taille, et la voie de l'autoroute où elle circule. Le temps de transition est représenté par une loi normale dont la moyenne et la covariance sont supposées être connues. Cette approche est limitée par le fait qu'elle soit dédiée au contexte routier qui considère que la direction d'une voiture est connue dans l'autoroute. Le fait de considérer que deux caméras ne permet pas de traiter les problèmes de prédire dans quelle caméra la voiture est susceptible d'apparaître.

Dans [Javed et al., 2003], le problème de ré-identification d'objets est modélisé comme un problème de maximisation de la probabilité a posteriori. La probabilité que l'objet b observé dans la caméra a correspond à l'objet d observé dans la caméra c est composée des probabilités de correspondances spatio-temporelles et d'apparence, comme indiqué dans l'équation (7.1).

$$\mathrm{P}(k_{i,b}^{j,d}|O_{i,b},O_{j,d}) = \frac{\mathrm{P}(O_{i,b}(app),O_{j,d}(app)|k_{i,b}^{j,d})\mathrm{P}(O_{i,b}(st),O_{j,d}(st)|k_{i,b}^{j,d})\mathrm{P}(k_{i,b}^{j,d})}{\mathrm{P}(O_{i,b},O_{j,d})} \tag{7.1}$$

- $k_{i,b}^{j,d}$ signifie que les deux observations (pistes) $O_{i,b}$ et $O_{j,d}$ appartiennent à un même objet. Autrement dit, les objets b et d sont le même objet.
- $\mathrm{P}(O_{i,b}(app),O_{j,d}(app)|k_{i,b}^{j,d})$ est la probabilité a priori de la correspondance entre les apparences des objets b et d. Elle modélise la distance entre l'apparence de b et l'apparence de d.
- $\mathrm{P}(O_{i,b}(st),O_{j,d}(st)|k_{i,b}^{j,d})$ est la probabilité a priori de correspondance entre les caractéristiques spatio-temporelles des objets b et d.
- $\mathrm{P}(O_{i,b},O_{j,d})$ est une constante de normalisation.
- $\mathrm{P}(k_{i,b}^{j,d}) = p(c_i,c_j)$ est la probabilité de transition de la caméra c_i à la caméra c_j.

Les probabilités a priori $\mathrm{P}(O_{i,b}(app),O_{j,d}(app)|k_{i,b}^{j,d})$, $\mathrm{P}(O_{i,b}(st),O_{j,d}(st)|k_{i,b}^{j,d})$ et $p(c_i,c_j)$ sont définies grâce à une phase d'apprentissage. L'apprentissage consiste à estimer ces probabilités à partir de correspondances connues. Dans la phase d'apprentissage, les correspondances sont déterminées par l'appariement des apparences des objets se déplaçant dans le réseau. Seuls les meilleurs appariements sont considérés durant l'apprentissage. Les correspondances générées dans cette étape constituent les entrées utilisées pour l'estimation des probabilités a priori. La probabilité spatio-temporelle est modélisée par une densité non paramétrique estimée avec la méthode des fenêtres de Parzen (*Parzen windows*). La méthode des fenêtres de Parzen prend comme entrée les vecteurs de caractéristiques spatio-temporelle des correspondances connues et estime la probabilité a priori sous forme non paramétrique. Le vecteur de caractéristiques contient la position de ré-apparition de l'objet, la position de sa sortie, la vitesse au moment de sortie et le temps de transition entre l'entrée et la sortie. Les régions d'entrée-sortie des caméras sont

définies par les coordonnées de position des objets à l'entrée et à la sortie d'une caméra. L'apprentissage de la probabilité d'apparence a montré que la distance (coefficient modifié de Bhattacharyya) entre les histogrammes d'un objet dans des caméras différentes suit une loi proche de la loi normale. La probabilité de transition entre les caméras c_i et c_j est définie par le ratio des personnes quittant la caméra i et ré-apparaissant dans la caméra j. Les probabilité a priori spatio-temporelle et d'apparence sont mises à jour à chaque instant en intégrant les observations des correspondances déterminées par le système.

La ré-identification d'objets consiste à trouver l'ensemble de correspondances K' maximisant la probabilité a posteriori $p(K|O)$ donnée par l'équation (7.2). $K = \{k_{i,b}^{j,d}\}$ est l'ensemble de toutes les correspondances et O est l'ensemble de toutes les observations (pistes). La solution K' est déterminée en trouvant le meilleur chemin d'un graphe orienté où les observations O constituent les noeuds du graphe et le logarithme de la probabilité $p(k_{i,b}^{j,d}|O_{i,b}, O_{j,d})$ représente le poids de l'arête reliant $O_{i,b}$ à $O_{j,d}$. La complexité de cette solution dépend du nombre d'observations (nombre d'éléments de O). Afin de limiter le nombre d'observations, seules les observations à l'intérieur d'une fenêtre de temps sont considérées. La taille de la fenêtre est estimée, à chaque instant, à partir de la densité de probabilité spatio-temporelle des pistes non-visibles. L'estimation consiste à choisir l'intervalle de temps à partir duquel la probabilité de ré-apparition de ces pistes dans n'importe quelle caméra du réseau est inférieure à un seuil prédéfini.

$$p(K|O) = \prod_{k_{i,b}^{j,d} \in K} p(k_{i,b}^{j,d}|O_{i,b}, O_{j,d}) \tag{7.2}$$

Le test de la méthode est effectué avec un réseau composé de trois caméras et avec plusieurs personnes se déplaçant dans le réseau. Les résultats obtenus démontrent la robustesse de la méthode. Le point faible de la méthode est son aspect centralisé pour la détermination de l'ensemble de correspondances K' maximisant la probabilité a posteriori. L'aspect centralisé est causé par le traitement simultané de toutes les correspondances dans la probabilité a posteriori. Un autre inconvénient de la méthode est le besoin en mémoire élevé pour le stockage de la probabilité spatio-temporelle. En effet, l'utilisation d'une fonction non paramétrique pour la représentation de la probabilité engendre un grand nombre d'éléments à stocker, en particulier quand la dimension du vecteur d'entrée est élevée (dimension 7 dans ce cas). L'avantage de l'approche est son universalité ce qui permet son utilisation quel que soit l'algorithme de suivi utilisé dans chaque caméra et quel que soit le type d'objets à suivre.

Dans [Dick and Brooks, 2005], seule la caractéristique spatio-temporelle est utilisée pour la ré-identification. Le problème de suivi inter-caméra est modélisé par un modèle de Markov. Chaque caméra exécute une détection basée sur la soustraction de fond et un suivi basé sur le filtre de Kalman. Le modèle de Markov est utilisé au sein d'une même caméra pour tenir compte des mouvements irréguliers des objets. L'image de chaque caméra est divisée en plusieurs régions et chaque région constitue un état du modèle de Markov. Chaque état est défini par sa probabilité initiale qui représente la probabilité de la naissance d'un nouvel objet. La transition d'un état à un autre est définie par une matrice

contenant les probabilités de transition. La transition peut s'effectuer entre deux états appartenant à une même caméra ou à des caméras différentes. Les probabilités initiales et les probabilités de transition sont déterminées grâce à une phase d'apprentissage. L'apprentissage consiste en une personne se déplaçant naturellement dans le réseau en portant un ballon de couleur distinctive par rapport au fond. La couleur distinctive permet de suivre facilement la personne et de compter le nombre de fois qu'elle passe d'un état à un autre. Chaque élément t_{ij} de la matrice de transition correspond au résultat du nombre de fois que la personne est passée de l'état j à l'état i. Les probabilités de transition sont déduites en normalisant chaque colonne de la matrice.

Quand un objet détecté dans l'état i n'est associé à aucune des pistes existantes, l'identité de l'objet est définie en se basant sur le modèle du Markov. Toutes les pistes disparues dans les 2 à K images précédentes sont considérées comme candidates pour la ré-identification. La piste disparue à l'état e qui maximise la probabilité de transition t_{ix} définit l'identité de l'objet si t_{ie} est supérieure à la probabilité d'initialiser une nouvelle piste dans l'état i. Dans le cas contraire, l'objet reçoit une nouvelle identité. Les résultats obtenus avec l'approche montrent une bonne ré-identification quand le déplacement des objets respecte les paramètres du modèle. Dans le cas contraire la ré-identification est erronée. L'inconvénient de la méthode est que les paramètres du modèle de Markov sont figés et n'évoluent pas dans le temps, ce qui limite l'utilisation de l'approche aux déplacements appris. Un autre inconvénient est que la méthode n'exploite pas l'apparence visuelle des objets, ce qui rend la discrimination difficile entre deux objets qui quittent au même instant le même état. De plus aucune contrainte temporelle n'est imposée sur le déplacement des objets. En effet, si la probabilité de transition entre deux états qui sont loin est élevée, un objet peut ré-apparaître dans le nouvel état même si sa disparition s'est produite dans les deux images précédentes. L'avantage de cette méthode est la complémentarité entre le filtre de Kalman et le modèle de Markov. Le fait d'utiliser des états de Markov définis par des blocs de pixels permet de limiter la taille de la matrice de transition et le temps de calcul pour la prise de décision.

Dans la prochaine section, nous développons un formalisme probabiliste basé sur l'apparence et l'information spatio-temporelle. Notre approche se base sur un traitement distribué ce qui permet de réduire les communications entre les noeuds. La caractéristique spatio-temporelle considérée dans notre cas se résume au temps et à la probabilité des transitions entre les régions entrées-sorties. Dans notre cas, une région d'entrée-sortie est définie par un bloc de pixels ce qui a pour but de limiter la mémoire et le calcul. La probabilité de correspondance résultante s'intègre naturellement dans le filtre GMPHD.

7.2 Notre approche

7.2.1 Principe

Dans le réseau de caméras, chaque caméra effectue le suivi multi-objet dans son plan image avec l'algorithme GMPHD. Le suivi dans le plan image permet d'éviter la calibration spatiale des caméras et de rendre le système flexible et facilement déployable. Quand un objet quitte le champ de vision d'une caméra, il faut déterminer sa probabilité d'apparition dans les autres caméras. Tout d'abord, les régions d'entrées et de sortie des

objets (E/S), dans une caméra, sont définies en se basant sur la topologie du réseau. Cette définition peut être effectuée manuellement par l'utilisateur ou automatiquement par un apprentissage. Une matrice reliant les entrées et les sorties des caméras est utilisée pour prédire dans quelles régions un objet est susceptible d'apparaître quand il quitte la région d'une caméra. Cette matrice peut être construite par une phase d'apprentissage comme expliqué dans [Dick and Brooks, 2005]. La matrice de transition permet de définir les communications entre les caméras du réseau. Une fois que la région de sortie s de l'objet est déterminée, ses caractéristiques sont communiquées par la caméra aux autres caméras ayants des régions E/S liées à la sortie s. La matrice des probabilités de transition définit pour chaque caméra la région où s'attendre à l'apparition de l'objet. Cette région est appelées la *région d'attente*. En se basant sur les caractéristiques reçues, chaque caméra évalue la probabilité qu'un objet détecté dans la région d'attente, soit l'objet attendu. Le calcul de la probabilité ne s'effectue que quand un objet est détecté dans la région, ce qui permet de réduire la complexité de calcul.

Une fois que l'objet est ré-identifié dans une des caméras, cette dernière communique l'information aux autres caméras attendant l'objet afin de supprimer ces caractéristiques et d'arrêter l'attente. Quand la ré-identification d'un objet est confirmée simultanément dans deux caméras différentes, la caméra présentant la probabilité de correspondance la plus élevée garde l'identité de l'objet. La ré-identification dans l'autre caméra est considérée comme une fausse alarme et la piste de l'objet est supprimée.

7.2.2 Formalisation

La formalisation mathématique de la probabilité d'apparition d'un objet dans une région après sa disparition est calculée en se basant sur l'apparence de l'objet, le temps de transition et la probabilité de transition entre les régions E/S. Supposons que le réseau est composé de deux caméras c_1 et c_2. Chaque objet dans une caméra est représenté par une composante Gaussienne dans le filtre GMPHD et par un modèle d'apparence comme l'histogramme. La probabilité que l'objet o', détecté dans la région r_{j,c_2} de la caméra c_2 avec l'apparence $App_{c_2,o'}$ à l'instant t_1, soit le même objet o ayant quitté la caméra c_1 à la région r_{i,c_1} avec l'apparence $App_{c_1,o}$ à l'instant t_0 est :

$$\mathrm{P}\left(o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\middle|o_{(r_{i,c_1},App_{c_1,o},t_0)}\right)$$

L'application de la formule de Bayes permet d'obtenir l'équation (7.3).

$$\mathrm{P}\left(o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\middle|o_{(r_{i,c_1},App_{c_1,o},t_0)}\right) = \frac{\mathrm{P}\left(o_{(r_{i,c_1},App_{c_1,o},t_0)}\middle|o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\right)\mathrm{P}\left(o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\right)}{\mathrm{P}\left(o_{(r_{i,c_1},App_{c_1,o},t_0)}\right)} \quad (7.3)$$

Supposons que l'apparence d'un objet est indépendante de la région de l'image où il se trouve. En appliquant cette hypothèse au terme $\mathrm{P}\left(o_{(r_{i,c_1},App_{c_1,o},t_0)}\middle|o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\right)$, nous obtenons l'équation (7.4).

$$\mathrm{P}\left(o_{(r_{i,c_1},App_{c_1,o},t_0)}\middle|o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\right) = \mathrm{P}\left(o_{(App_{c_1,o})}\middle|o'_{(App_{c_2,o'})}\right)\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\middle|o'_{(r_{j,c_2},t_1)}\right) \quad (7.4)$$

— $\mathrm{P}\left(o_{(App_{c_1,o})}\middle|o'_{(App_{c_2,o'})}\right)$ est la probabilité de correspondance entre les apparences des objets o et o'. Cette probabilité modélise le changement de l'apparence d'un objet ayant quitté la caméra c_1 et qui ré-apparaît dans la caméra c_2. Elle est supposée inclure toutes les variations que l'apparence d'un objet subisse quand il transite de c_1 à c_2. Ces variations peuvent être dues aux changements d'éclairage, à la différence entre les capteurs des caméras ou à la différence des points de vue.

— $\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\middle|o'_{(r_{j,c_2},t_1)}\right)$ est la probabilité de correspondance spatio-temporelle. Cette probabilité modélise l'information spatio-temporelle quant un objet quitte la région r_{i,c_1} à l'instant t_0 et ré-apparaît dans la région r_{j,c_2} à l'instant t_1. Autrement dit, cette probabilité modélise le temps nécessaire pour un objet de ré-apparaître dans la région r_{j,c_2} quand il quitte la région r_{i,c_1}.

Comme le temps de transition d'un objet de la région r_{i,c_1} à la région r_{j,c_2} est défini par $\Delta t = t_1 - t_2$, la probabilité $\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\middle|o'_{(r_{j,c_2},t_1)}\right)$ peut être écrite comme indiqué dans l'équation (7.5).

$$\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\middle|o'_{(r_{j,c_2},t_1)}\right) = \mathrm{P}\left(o_{(r_{i,c_1})}\middle|o'_{(r_{j,c_2})}\right)\mathrm{P}\left(\Delta t\middle|o'_{(r_{j,c_2})},o_{(r_{i,c_1})}\right) \tag{7.5}$$

En replaçant tous les termes développés dans l'équation (7.3), nous obtenons l'équation (7.6).

$$\mathrm{P}\left(o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\middle|o_{(r_{i,c_1},App_{c_1,o},t_0)}\right) = \frac{\mathrm{P}\left(o_{(App_{c_1,o})}\middle|o'_{(App_{c_2,o'})}\right)\mathrm{P}\left(o_{(r_{i,c_1})}\middle|o'_{(r_{j,c_2})}\right)\mathrm{P}\left(\Delta t\middle|o'_{(r_{j,c_2})},o_{(r_{i,c_1})}\right)\mathrm{P}\left(o'_{(r_{j,c_2},t_1)}\right)\mathrm{P}\left(o'_{(App_{c_2,o'})}\right)}{\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\right)\mathrm{P}\left(o_{(App_{c_1,o})}\right)} \tag{7.6}$$

— $\mathrm{P}\left(o_{(App_{c_1,o})}\right)$ et $\mathrm{P}\left(o'_{(App_{c_2,o'})}\right)$ sont, respectivement, la probabilité que l'apparence de l'objet o est $App_{c_1,o}$ dans la caméra c_1 et que l'apparence de l'objet o' est $App_{c_2,o'}$ dans la caméra c_2.

— $\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\right)$ et $\mathrm{P}\left(o'_{(r_{j,c_2},t_1)}\right)$ sont respectivement la probabilité que l'objet o se trouve dans la région r_{i,c_1} à l'instant t_0 et la probabilité que l'objet o' se trouve dans la région r_{j,c_2} à l'instant t_1.

Si nous considérons que $\mathrm{P}\left(o_{(App_{c_1,o})}\right) = \mathrm{P}\left(o'_{(App_{c_2,o'})}\right)$, alors l'équation (7.6) se transforme à l'équation (7.7). Cette hypothèse est valide car l'apparence de l'objet dépend de la méthode d'extraction qui est supposée être la même dans toutes les caméras du réseau.

$$\mathrm{P}\left(o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\middle|o_{(r_{i,c_1},App_{c_1,o},t_0)}\right) = \frac{\mathrm{P}\left(o_{(App_{c_1,o})}\middle|o'_{(App_{c_2,o'})}\right)\mathrm{P}\left(o_{(r_{i,c_1})}\middle|o'_{(r_{j,c_2})}\right)\mathrm{P}\left(\Delta t\middle|o'_{(r_{j,c_2})},o_{(r_{i,c_1})}\right)\mathrm{P}\left(o'_{(r_{j,c_2},t_1)}\right)}{\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\right)} \tag{7.7}$$

En considérant que l'algorithme de suivi est basé sur le filtre GMPHD, nous aurons :

- $\mathrm{P}\left(o'_{(r_{j,c_2},App_{c_2,o'},t_1)}\middle|o_{(r_{i,c_1},App_{c_1,o},t_0)}\right)$ représente le poids de la Gaussienne initialisant l'objet détecté
- $\mathrm{P}\left(o_{(r_{i,c_1},t_0)}\right)$ représente le poids de la Gaussienne de l'objet au moment où il quitte la région r_{i,c_1}.
- $\mathrm{P}\left(o'_{(r_{j,c_2},t_1)}\right)$ représente la probabilité de détection de l'objet o' dans la région r_{j,c_2}. Autrement dit, c'est la confiance dans l'observation fournie par le détecteur.
- $\mathrm{P}\left(o_{(r_{i,c_1})}\middle|o'_{(r_{j,c_2})}\right)$ la probabilité de transition de la région r_{i,c_1} à la région r_{j,c_2} qui est définie par une matrice de transition. Un exemple de matrice de probabilité de transition est fourni dans l'équation (7.8).
- $\mathrm{P}\left(\Delta t\middle|o'_{(r_{j,c_2})},o_{(r_{i,c_1})}\right)$ est la probabilité modélisant le temps de transition de l'objet de la région r_{i,c_i} à la région r_{j,c_2}. Le temps de transition est une fonction de la distance entre les régions et de la vitesse des objets. Cette probabilité peut être estimée manuellement dans le cas où la vitesse des objets est relativement constante et le chemin entre deux régions est relativement le même. Dans le cas général, un apprentissage permet de définir cette probabilité.
- $\mathrm{P}\left(o_{(App_{c_1,o})}\middle|o'_{(App_{c_2,o'})}\right)$ est la probabilité modélisant la correspondance d'apparence entre objets, comme défini précédemment. Un apprentissage et une mise à jour automatique de cette probabilité sont souvent nécessaires. L'apprentissage peut s'effectuer comme indiqué dans [Javed et al., 2003, Javed et al., 2008].

Quand un objet o quitte une région r_{n,c_i}, la caméra c_i informe toutes les caméras $C = \{c_1, \ldots, c_p\}$ où il est susceptible d'apparaître. Le message envoyé à chaque caméra $c \in C$ contient :

- la région de sortie r_{n,c_i}
- l'identité de l'objet
- le temps de sortie t_0
- l'apparence de l'objet *App*
- le poids de la Gaussienne au moment de sortie

L'organigramme fourni dans la figure 7.2 explique ce mécanisme quand un objet o quitte la région r_{n,c_1} pour apparaître dans la région r_{m,c_2}, pour la configuration du réseau de caméras indiquée dans la figure 7.1.

Quand la caméra c détecte un objet o' qui n'est associé à aucune des Gaussiennes du mélange après le suivi, une nouvelle Gaussienne est initialisée. Les paramètres de la Gaussienne dépendent si l'objet détecté est un nouvel objet où correspond à l'objet o transmis par la caméra c_i. Si la caméra ne s'attend à aucun objet en transition dans la région d'image où l'objet o' est détecté, o' est considéré comme un nouvel objet. La Gaussienne d'un nouvel objet est caractérisée par un poids d'initialisation w_0 qui est fixe et une nouvelle identité. Si la région où l'objet o' est détecté correspond à la région $r_{m,c}$ d'attente de l'objet o, la probabilité w que o' et o représentent le même objet est calculée avec l'équation (7.6). Si w est inférieur à w_0 alors o' est considéré comme un nouvel objet.

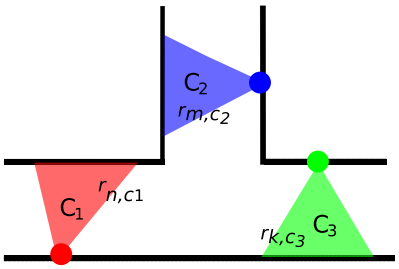

FIGURE 7.1 – Exemple du réseau de caméras avec des champs de vision non recouvrants utilisé pour l'explication des communications entre les caméras quand un objet quitte la région r_{n,c_1} pour apparaître dans la région r_{m,c_2}. L'entrée-sortie r_{n,c_1} est reliée aux deux caméras c_2 et c_3.

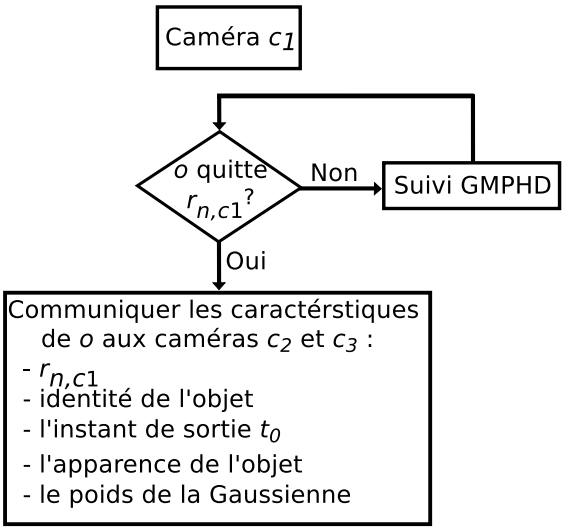

FIGURE 7.2 – Organigramme expliquant la transmission des caractéristiques de l'objet o ayant quitté la région r_{n,c_1} de la caméra c_1. L'organigramme est fourni pour la configuration du réseau de caméras présentée dans la figure 7.1.

Si w est supérieur à w_0 alors o et o' représentent le même objet. Dans ce cas, le poids de la Gaussienne est égal à w et l'identité de la Gaussienne est celle de l'objet o. Cette étape s'effectue dans le module "GMPHD Initialisation" de la figure 6.2. Tous ce traitement est expliqué dans l'organigramme de la figure 7.3.

Quand plusieurs objets sont attendus dans la même région de la caméra, l'identité de celui ayant la probabilité w la plus élevée est assignée à la Gaussienne initialisée. Le suivi multi-objet se poursuit normalement. Si à l'instant suivant, la piste de l'objet o' est confirmée, un message est envoyé aux autres caméras où l'objet o est susceptible d'apparaître, afin qu'elles arrêtent d'attendre. Le message envoyé contient l'identité de l'objet et la probabilité w (figure 7.3).

Si plusieurs caméras ré-identifient et confirment simultanément l'objet o, la caméra ayant la probabilité w la plus élevée garde l'identité de l'objet o. Dans les autres caméras, la Gaussienne correspondant à l'objet est supprimée afin qu'il reçoive à l'instant suivant une nouvelle identité qui peut être celle d'un autre objet en transition ou celle d'un nouvel objet. L'organigramme de la figure 7.4 explique les procédures effectuées à chaque instant dans chaque caméra pour éviter que plusieurs objets se retrouvent avec la même identité. Dans la prochaine section, nous expliquons la simulation utilisée pour le test de l'approche.

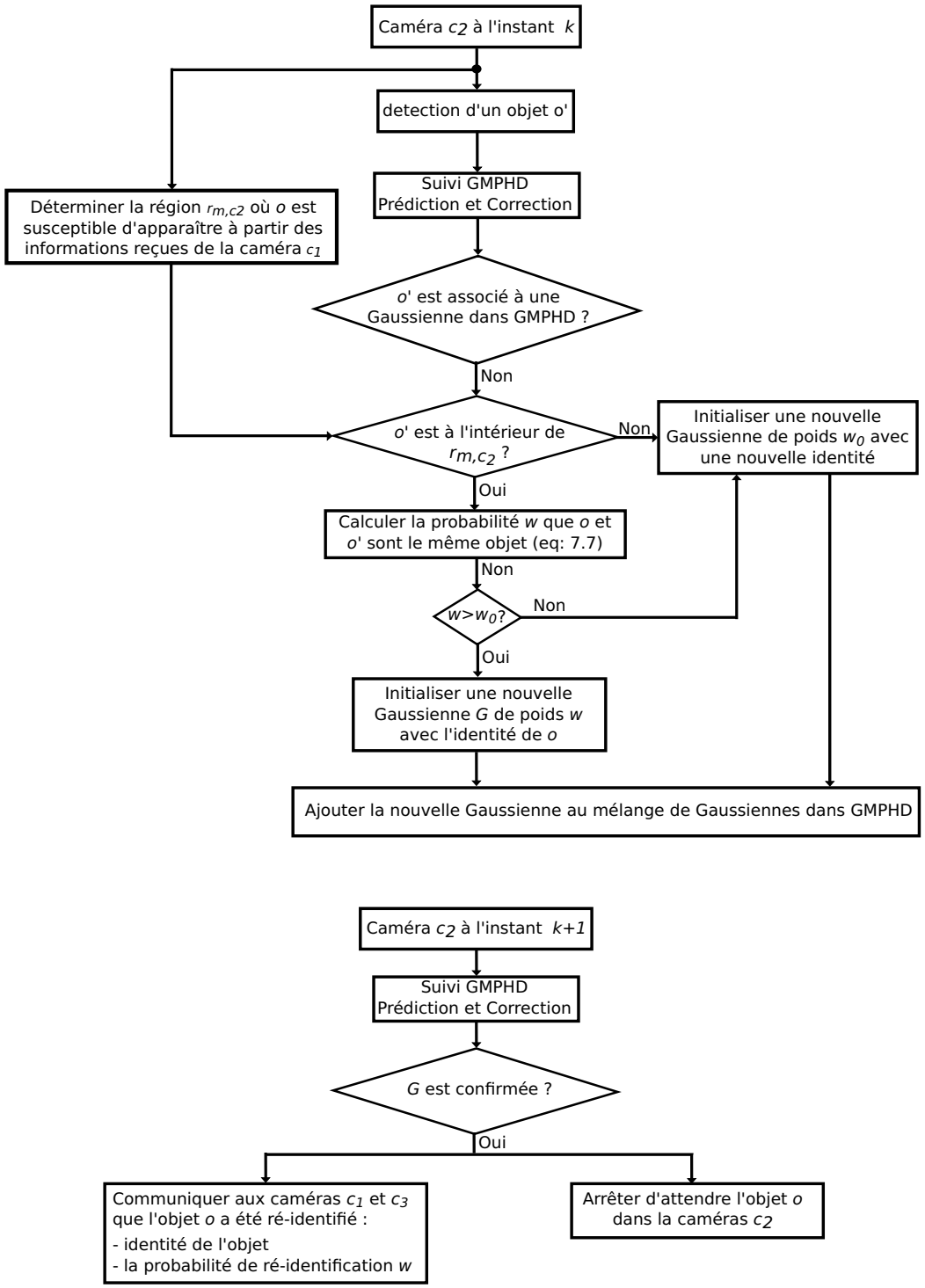

FIGURE 7.3 – Organigramme expliquant l'attente de l'objet o dans la caméra c_2. L'organigramme est basé sur la configuration du réseau de caméras présentée dans la figure 7.1.

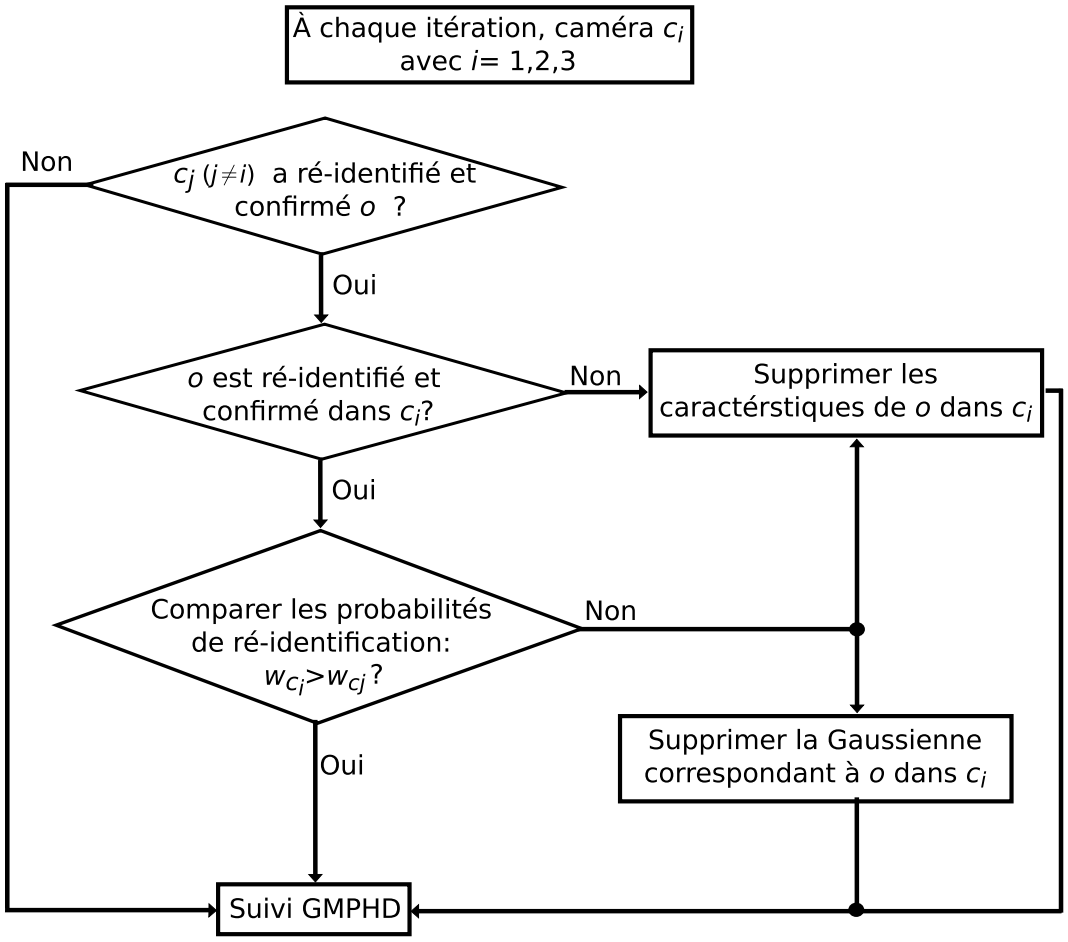

FIGURE 7.4 – Organigramme expliquant l'heuristique utilisée pour garantir la cohérence dans les identités afin d'éviter que des objets se trouvant dans différentes caméras aient la même identité. L'organigramme est basé sur la configuration du réseau de caméras présentée dans la figure 7.1.

7.3 Simulation

Pour tester l'approche, un réseau de caméras dont la topologie est fournie par la figure 7.5 est simulé. Le scénario est un couloir en forme de T surveillé avec 4 caméras. Chaque caméra possède deux régions d'entrée/sortie. Par exemple, quand un objet o quitte la caméra c_1 de la région $r_{2,1}$, son apparition est susceptible dans les régions $r_{2,1}$, $r_{3,2}$ et $r_{7,4}$. En effet, un objet est susceptible de ré-apparaître dans la région où il a disparu. Quand un objet quitte le couloir par les régions $r_{1,1}$, $r_{5,3}$ ou $r_{8,4}$, l'objet peut ne pas revenir dans le couloir. Certaines transitions entre régions ne sont pas possibles. Par exemple, un objet qui a quitté la région $r_{2,1}$ ne peut pas apparaître dans la région $r_{4,2}$ ou $r_{5,3}$.

Pour réaliser la simulation, les probabilités de transition entre régions sont définies par la matrice de transition T_{tran} indiquée dans l'équation (7.8). La matrice de transition est une matrice creuse ce qui permet un traitement distribué. Les caméras C_1 et C_4 n'effectuent aucune communication avec la caméra C_3. Pour la caméra C_2, selon la région

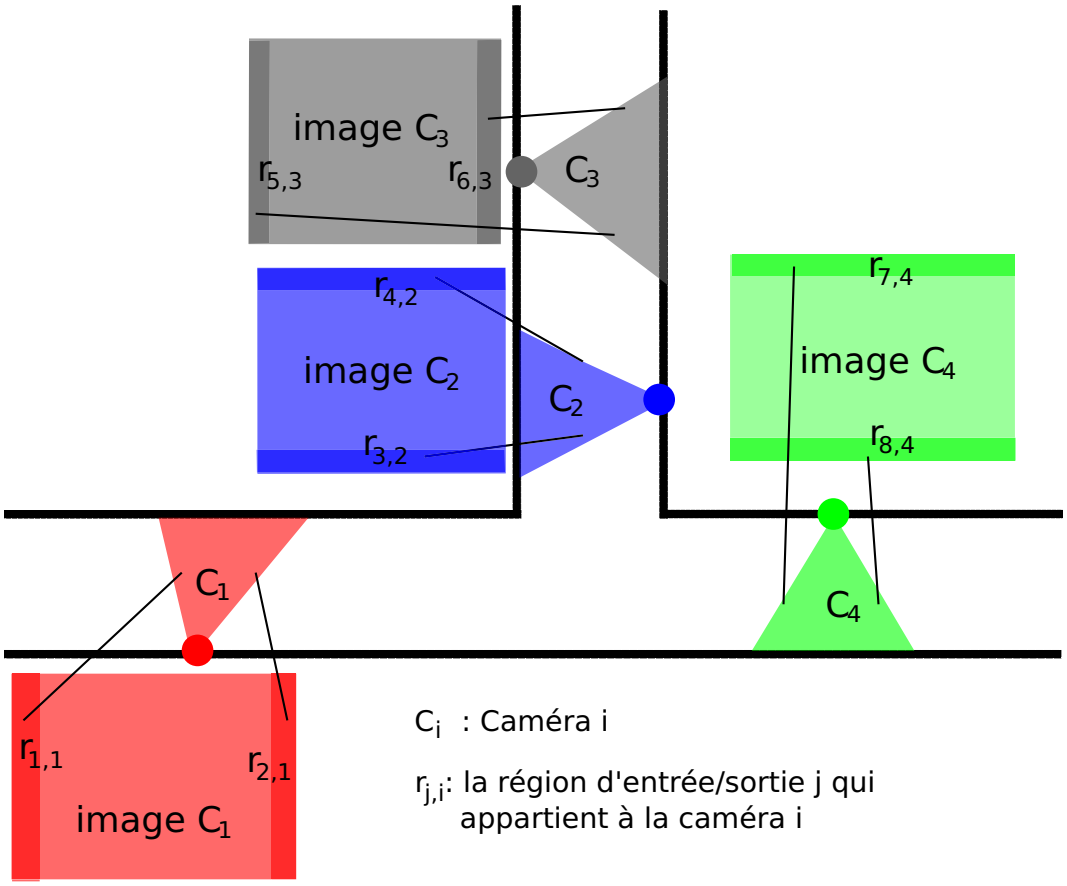

FIGURE 7.5 – Réseau de caméras simulé. Un couloir en forme de T où les segments de droite noir représentent les mures. Le couloir est surveillé avec 4 caméras où chacune possède deux régions d'entrée/sortie.

E/S sollicitée, elle peut communiquer soit avec C_1 et C_4 soit avec C_3.

$$T_{tran} = \begin{pmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0.45 & 0 & 0 & 0 & 0.45 & 0 \\ 0 & 0.45 & 0.1 & 0 & 0 & 0 & 0.45 & 0 \\ 0 & 0 & 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0.45 & 0.45 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0.9 & 0 & 0 & 0 & 0 & 0.9 & 0 & 0.9 \end{pmatrix} \tag{7.8}$$

Pour les huit premières lignes et les huit colonnes de la matrice T_{tran}, chaque élément t_{ij} représente la probabilité de transition de la région $r_{j,.}$ à la région $r_{i,.}$. La dernière ligne modélise la sortie définitive des objets du couloir. En effet, un objet quittant la région $r_{1,1}$ a une probabilité de 0.1 de ré-apparaître dans la même région après un certain temps et une probabilité de 0.9 de sortir du couloir. Dans la réalité, cette matrice peut être générée

grâce à une phase d'apprentissage qui analyse le mouvement des objets dans le réseau, comme dans [Dick and Brooks, 2005].

L'apparence d'un objet est décrite par un vecteur normalisé de m éléments positifs qui peut représenter un histogramme couleur dans la réalité. Le vecteur d'apparence est généré aléatoirement quand l'objet est initialisé. L'apprentissage réalisé dans [Javed et al., 2003] a montré que la distance entre deux histogrammes d'un même objet observé par deux caméras différentes suit une loi proche de la loi normale quand le coefficient de Bhattacharyya modifié (équation (7.9)) est utilisé pour le calcul de la distance. Dans un scénario réel, la moyenne de cette loi normale est non nulle et elle traduit le changement d'apparence dû aux variations colorimétriques entre les caméras, aux changements du point de vue et d'échelles des caméras. La même loi est utilisée dans nos simulations. Simuler le changement d'apparence revient à affecter le coefficient de Bhattacharyya modifié par un bruit Gaussien. Pour faciliter la simulation, nous supposons que la variance de la loi normale est la même quel que soit le couple de caméras. La formule de la probabilité d'apparence est alors donnée par l'équation (7.10).

$$D_{Bhatt}(App_{c_1,o}, App_{c_2,o'}) = \sqrt{1 - \sum_{i=1}^{m} \sqrt{App_{c_1,o}(i) App_{c_2,o'}(i)}} \tag{7.9}$$

$$\mathrm{P}\left(o_{(App_{c_1,o})} \middle| o'_{(App_{c_2,o'})}\right) = \frac{1}{\sqrt{\sigma^2_{App(c_1,c_2)}(2\pi)}} e^{-\frac{1}{2}\frac{(D_{Bhatt}(App_{c_1,o}, App_{c_2,o'} - \mu_{App(c_1,c_2)})^2}{\sigma^2_{App(c_1,c_2)}}} \tag{7.10}$$

$\mu_{App(c_i,c_j)}$ et $\sigma^2_{App(c_i,c_j)}$ sont respectivement la moyenne et la variance affectant la distance entre deux vecteurs d'apparence lorsqu'un objet disparaît de la caméra c_i et ré-apparaît dans la caméra c_j. Dans nos simulations, $\mu_{App(c_i,c_j)} = 0$ et $\sigma^2_{App(c_i,c_j)} = \sigma^2_{App}$ quel que soit le couple c_i et c_j.

Le temps de transition entre deux régions est modélisé par la loi normale donnée dans l'équation (7.11). Le temps moyen de transition entre régions est donné par la matrice $Temps_{trans}$ indiqué par l'équation (7.12). Chaque élément $Temps_{trans}(i,j)$ de la matrice $Temps_{trans}$ représente le temps moyen nécessaire pour qu'un objet transite de la région $r_{j,.}$ à la région $r_{i,.}$. La représentation de temps de transition par une loi normale est réaliste quand la vitesse des objets est peu variable et le chemin parcouru par les objets entre deux régions est relativement le même. Ces hypothèses sont souvent respectées dans un couloir.

$$\mathrm{P}\left(\Delta t \middle| o'_{(r_{j,c_2})}, o_{(r_{i,c_1})}\right) = \frac{1}{\sqrt{2\pi\sigma^2_{ji}}} e^{-\frac{1}{2}\frac{(\Delta t - Temps_{tran}(j,i))^2}{\sigma^2_{ji}}} \tag{7.11}$$

σ^2_{ji} est la variance de la loi normale modélisant le temps de transition de la région r_{i,c_1} à la région r_{j,c_2}. Cette variance est un paramètre contrôlable à la simulation et est défini par un pourcentage du temps de transition moyen $Temps_{tran}(j,i)$.

$$Temps_{tran} = \begin{pmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 50 & 0 & 0 & 0 & 50 & 0 \\ 0 & 50 & 10 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 10 & 30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 30 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 50 & 50 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{pmatrix} \tag{7.12}$$

En se basant sur ces informations, des transitions d'objets entre régions sont simulées. Quand un objet entre le champ de vision d'une caméra, il peut la quitter par n'importe quelle région. La probabilité de quitter la caméra par la région d'où il est entré est de 0.2 et la probabilité de la quitter par l'autre région est de 0.8. Le rôle du simulateur est la génération des observations des objets. Le vecteur d'observation contient les coordonnées de position x, y de l'objet dans l'image ainsi que son modèle d'apparence. La probabilité qu'un objet réel est détectable (génère une observation) est p_D. Des fausses alarmes sont aussi considérées dans le simulateur. Les observations sont utilisées comme entrées pour l'approche de suivi multi-caméra proposée. Dans la prochaine section, notre approche est testée en utilisant les données fournies par le simulateur et les résultats obtenus sont présentés.

7.4 Résultats

Pour la validation de l'approche multi-caméra, nous supposons qu'un filtre GMPHD est utilisé dans chaque caméra pour le suivi multi-objet. Le GMPHD prend en entrée les observations générées par le simulateur. Pour pouvoir tester notre approche, les différentes probabilités intervenant dans le calcul de l'équation (7.7) sont définies comme suit :

- $\mathrm{P}\left(o_{(r_{i,c_1})} \middle| o'_{(r_{j,c_2})}\right)$ est définie par la matrice de transition entre régions T_{tran} donnée par l'équation (7.8).
- $\mathrm{P}\left(o'_{(r_{j,c_2},t_1)}\right) = p_D$ qui est la probabilité de détection utilisée dans le simulateur pour la génération des observations et qui correspond au paramètre p_D du filtre GMPHD. Dans les simulations cette probabilité est fixée à $p_D = 0.9$.
- La nombre de fausses alarmes dans une image suit une loi de poisson de moyenne égale à 1. La distribution des fausses alarmes suit une loi uniforme sur toute l'image. Les fausses alarmes dans deux images successives sont indépendantes. La probabilité qu'une image contient des fausses alarmes est fixée à 0.1 dans les simulations réalisées.
- La probabilité de temps de transition est modélisée par la loi normale donnée par l'équation (7.11).
- La probabilité de changement d'apparence est modélisée par la loi normale donnée par l'équation (7.9).
- Dans le filtre GMPHD, la probabilité qu'une observation non associée aux pistes existantes soit générée par un nouvel objet est fixée dans toutes les simulations à

$w_0 = 0.0001$.

Afin d'éviter qu'une caméra attende indéfiniment un objet, dans le cas où celui-ci est mal ré-identitfié ailleurs, nous fixons une contrainte sur le temps d'attente. En effet, si après $t_{moyen} + 5t_\sigma$ l'objet n'est pas ré-identifié, la caméra ne l'attend plus. Dans ce cas, ses caractéristiques sont supprimées de la caméra. t_{moyen} est le temps de transition moyen et t_σ est l'écart-type du temps de transition.

7.4.1 Métriques et scénario

L'évaluation de l'approche s'effectue en déterminant le pourcentage de ré-identifications correctes. Quatre expérimentations sont réalisées. La première consiste à déterminer le pourcentage de ré-identifications correctes en fonction du nombre maximum d'objets se déplaçant dans le réseau. Dans la deuxième, le pourcentage de ré-identifications est déterminé en fonction du nombre d'éléments dans le vecteur d'apparence. La troisième et la quatrième étudient le pourcentage de ré-identification en fonction de la variance de la loi normale modélisant la correspondance d'apparence et la variance de la loi normale modélisant le temps de transition.

Expérimentation 1 : Le nombre maximum d'objets présents dans le réseau varie de 1 à 10. Les paramètres de simulation sont fixés comme suit :

- Le vecteur d'apparence contient 128 éléments
- $\sigma_{App} = 0.1$. Dans les estimations par apprentissage réalisées dans [Javed et al., 2003], l'écart-type extrait des courbes fournies est de l'ordre de 0.1.
- $\sigma_{ij} = Temps_{trans}(i,j)\frac{10}{100}$.

Expérimentation 2 : Le nombre d'éléments constituant le vecteur d'apparence varie de 2 à 256 en considérant que les puissances de 2.

- Le nombre maximum d'objets dans le réseau est fixé à 5
- $\sigma_{App} = 0.1$.
- $\sigma_{ij} = Temps_{trans}(i,j)\frac{10}{100}$.

Expérimentation 3 : L'écart-type σ_{App} la loi normale modélisant la correspondance d'apparence varie de 0.1 à 10.

- Le nombre maximum d'objets dans le réseau est fixé à 5
- Le vecteur d'apparence contient 128 éléments
- $\sigma_{ij} = Temps_{trans}(i,j)\frac{10}{100}$.

Expérimentation 4 : Les écarts-types σ_{ij} des lois normales modélisant le temps de transition varient de 10% à 1000% de temps moyen de transition

- Le vecteur d'apparence contient 128 éléments
- Le nombre maximum d'objets dans le réseau est fixé à 5
- $\sigma_{App} = 0.1$.

7.4.2 Discussion

Les résultats de la première expérimentation sont présentés dans la figure 7.6. Le pourcentage de ré-identifications correctes diminue linéairement avec le nombre maximum d'objets dans le réseau. Plus le nombre d'objets est élevé, plus il y a de possibilités que deux ou plusieurs objets différents quittent la même région d'E/S au même instant ou à des instants proches. Dans ce cas, la probabilité spatio-temporelle devient non pertinente pour la discrimination des objets. Si les modèles d'apparence de ces objets sont proches, la probabilité que la ré-identification soit erronée est élevée. Quel que soit le nombre d'objets dans la scène, le taux de ré-identifications correctes est supérieur à 75% ce qui représente un bon résultat.

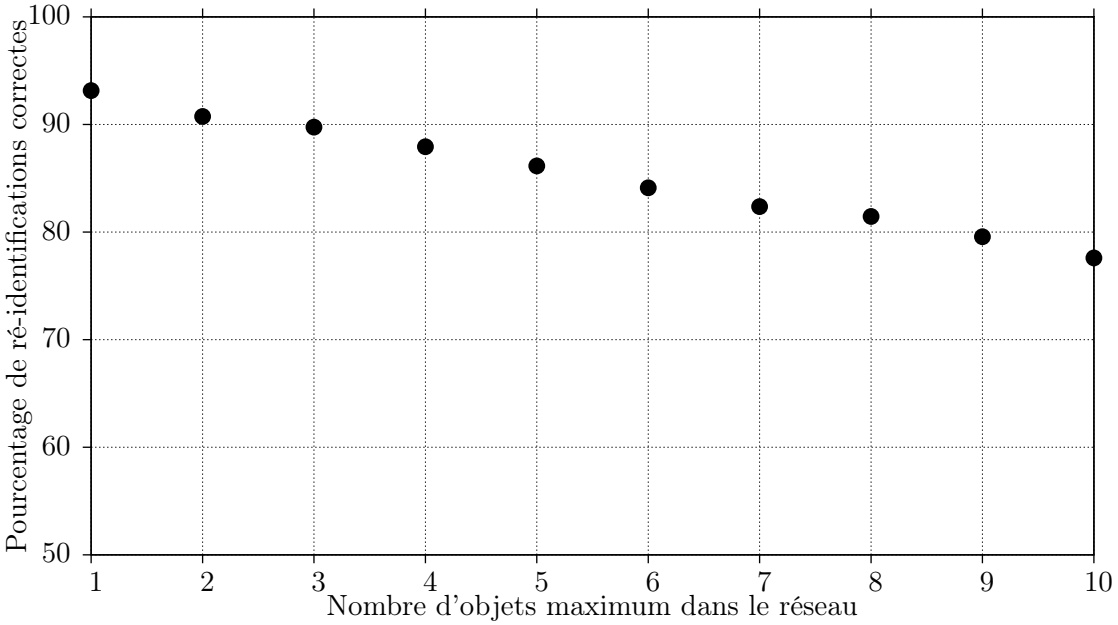

FIGURE 7.6 – Pourcentage de ré-identifications correctes en fonction du nombre d'objets présents dans le réseau de caméras (figure 7.5). 100 différentes simulations sont réalisées. Chaque simulation est stoppée après 100 transitions.

Les résultats de la deuxième expérimentation sont présentés dans la figure 7.7. Le pourcentage de ré-identifications correctes varie peu avec le nombre d'éléments constituant le vecteur d'apparence. Cela peut s'expliquer par la probabilité de correspondance d'apparence qui est une loi normale sur la distance entre deux vecteurs d'apparence. En simulation, cela revient à affecter la distance entre les vecteurs d'apparence par un bruit Gaussien additif. Ainsi, aucun bruit n'affecte les éléments composant le vecteur d'apparence. Suite à cela, le fait d'utiliser une variance constante quel que soit le nombre d'éléments composant le vecteur d'apparence, le taux de ré-identification devient indépendant de la taille du vecteur d'apparence. En effet, dans la réalité la variance du bruit dépend du nombre d'éléments composant l'histogramme utilisé. Plus le nombre d'éléments est réduit, plus la variance de la loi normale est grande. Donc pour un nombre d'éléments donné, la variance du bruit correspondant doit être estimée par une phase d'apprentissage.

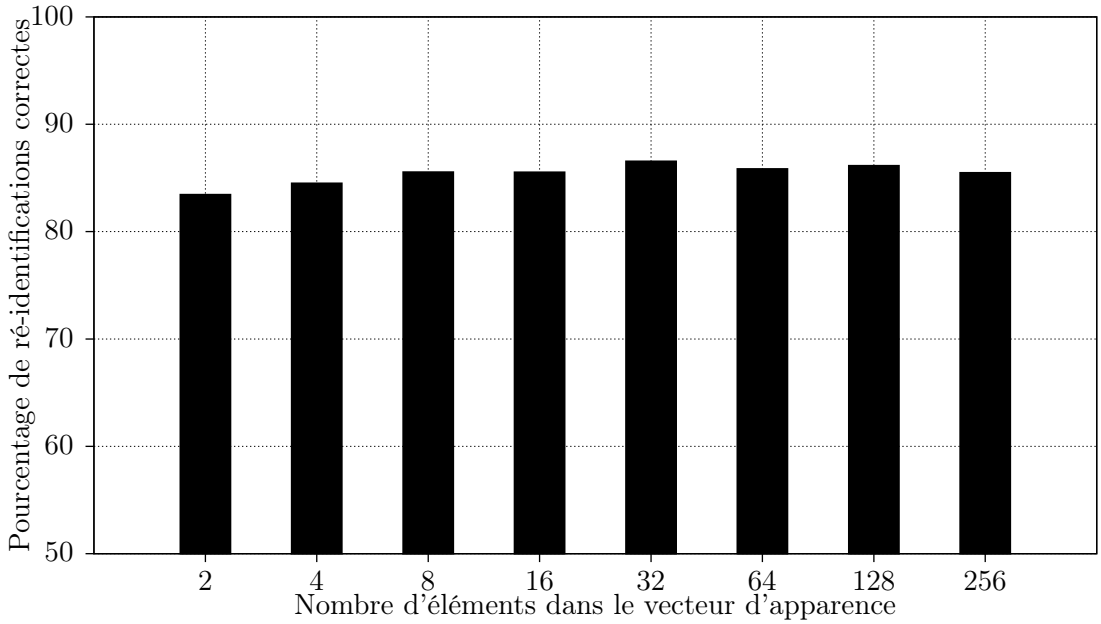

FIGURE 7.7 – Pourcentage de ré-identifications correctes en fonction du nombre d'éléments constituant le vecteur d'apparence. 100 différentes simulations sont réalisées. Chaque simulation est stoppée après 100 transitions.

Les résultats de la troisième expérimentation sont présentés dans la figure 7.8. Le pourcentage de ré-identifications correctes est élevé pour les faibles σ_{App}. Le taux de ré-identification est supérieur à 80% pour un écart-type inférieur à 1. Plus l'écart-type augmente, plus le taux de ré-identification diminue progressivement. La diminution progressive est due à l'intervention de la probabilité spatio-temporelle. En effet, comme l'écart-type de la loi normale modélisant le temps de transition est de 10% du temps de transition moyen, la correspondance spatio-temporelle augmente la probabilité de correspondance, ce qui permet d'assurer des ré-identifications correctes. Afin de tester cette hypothèse, nous avons effectué une simulation avec $\sigma_{App} = 0.5$ et $\sigma_{ij} = Temps_{tran}(i,j)\frac{50}{100}$ pour 5 objets dans le réseau. Le taux de ré-identifications correctes obtenu n'est que de 63%. Comme la distance entre deux vecteurs d'apparence (équation (7.9)) est toujours comprise entre 0 et 1, dès que l'écart-type est supérieur à 1, il peut influencer considérablement la valeur de la distance et engendrer des ré-identifications erronées. Le taux de ré-identification obtenus reste toujours supérieur 75% pour les valeurs simulées de l'écart-type.

Les résultats de la quatrième expérimentation sont présentés dans la figure 7.9. Le pourcentage de ré-identifications correctes est élevé pour les faibles σ_{ij}. Tant que l'écart-type σ_{ij} est inférieur à 50% du temps de transition moyen, le taux de ré-identification est supérieur à 80%. Plus l'écart-type augmente, plus le taux de ré-identifications correctes diminue jusqu'à atteindre 68% pour $\sigma_{ij} = Temps_{tran}(i,j)$. Au delà de cette valeur, le taux de ré-identifications correctes diminue rapidement pour atteindre 41% pour $\sigma_{ij} = 10Temps_{tran}(i,j)$. Un écart-type $\sigma_{ij} = Temps_{tran}(i,j)\frac{50}{100}$ signifie que le temps de transition d'un objet entre la région $r_{j,.}$ et $r_{i,.}$ est compris entre $\frac{1}{2}Temps_{tran}(i,j)$ et

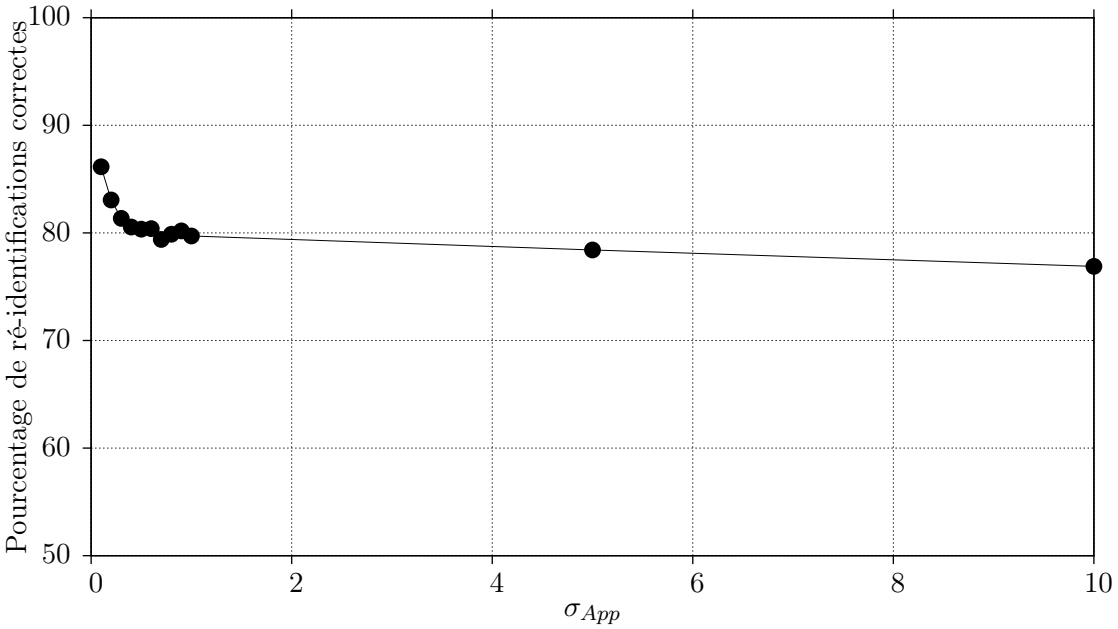

FIGURE 7.8 – Pourcentage de ré-identifications correctes en fonction de l'écart-type de bruit modélisant le changement d'apparence. 100 différentes simulations sont réalisées. Chaque simulation est stoppée après 100 transitions.

$\frac{3}{2}Temps_{tran}(i,j)$ en suivant une loi normale. Dans le scénario étudié qui un couloir en forme de T, cette valeur d'écart-type peut être considérée comme élevée. Donc même pour cette valeur élevée, le taux de ré-identifications correctes est supérieur à 80%.

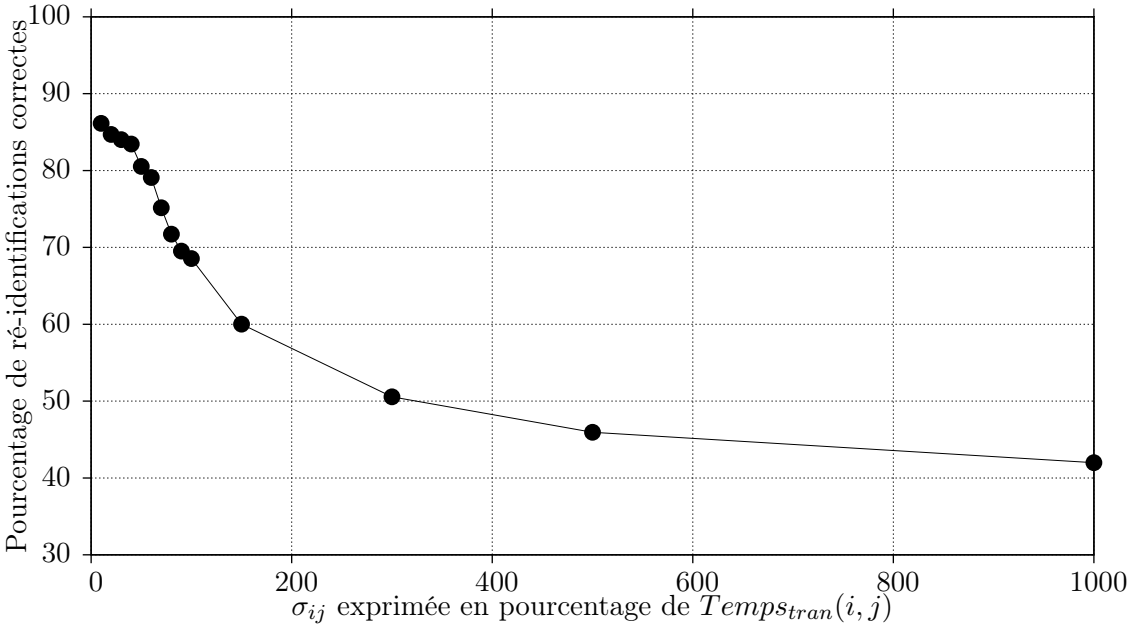

FIGURE 7.9 – Pourcentage de ré-identifications correctes en fonction de l'écart-type de temps de transition. L'écart-type est un pourcentage du temps moyen de transition. 100 différentes simulations sont réalisées. Chaque simulation est stoppée après 100 transitions.

Les taux de ré-identifications obtenus dans les différentes expérimentations sont prometteurs pour un fonctionnement dans un réseau de caméra réel. Afin d'avoir une idée sur le coût de l'approche proposée, une analyse de complexité est effectuée dans la section suivante.

7.5 Analyse de complexité

Dans cette section, nous nous intéressons à la complexité liée à la taille d'un message communiqué entre deux caméras, aux variables à stocker dans chaque caméra pour réaliser la ré-identification et à la complexité de calcul.

7.5.1 Communication

Comme expliqué dans la section 7.2.2, le message à transmettre aux caméras voisines doit contenir, l'identité de l'objet, la région de sortie, le vecteur d'apparence, le temps de sortie et le poids de la Gaussienne à la sortie. Si nous traduisons ce message par sa taille mémoire, nous obtenons les tailles suivantes pour les différents éléments :

- l'identité : entier sur 32 bits
- la région de sortie : la dynamique de cette variable dépend du nombre de caméras et du nombre d'entrées-sorties par caméra. Dans notre exemple de simulation, 8 bits sont suffisants.
- le vecteur d'apparence : si un histogramme de 256 niveaux est utilisé pour représenter l'apparence de l'objet et si une dynamique de 8 bits est utilisée pour le codage de la valeur d'un niveau, la taille du vecteur d'apparence sera de 256 octets. La taille du vecteur augmente avec le nombre d'éléments et la dynamique de chaque élément. De manière générale pour m éléments où chaque élément est codé sur n bits, la taille mémoire nécessaire est mn bits.
- le temps de sortie : dans un réseau de caméra réel, ce temps doit être transféré sous forme de structure contenant l'heure, les minutes et les secondes. Une variable de 24 bits est suffisante pour représenter cette structure.
- le poids de la Gaussienne au moment de sortie : dans notre cas, le poids est codé en flottant sur 32 bits.

La taille totale du message est 268 octets, ce qui permet un fonctionnement dans des réseaux avec une faible bande passante. Quand une caméra ré-identifie un objet elle transmet aux caméras voisines la probabilité de correspondance et l'identité de l'objet, ce qui revient à envoyer un message de 64 bits. La bande passante requise par le système dépend de la taille du message et du nombre de caméras auxquelles il faut le transmettre. L'intérêt d'une approche distribuée est que le nombre de caméras auxquelles le message est communiqué se restreint aux caméras voisines.

7.5.2 Mémoire

Afin de ré-identifier les objets, chaque caméra doit sauvegarder des variables contenant les informations des caméras voisines. L'aspect distribué et le traitement local d'information fournie par notre approche permet donc d'assurer une dispersion

des variables à stocker par le système sur les noeuds du réseau contrairement à une architecture centralisée où le centre de traitement doit stocker toutes les variables. Le nombre de variables à stocker dans une caméra c dépend du nombre de régions E/S dans la caméra c, le nombre de régions E/S liées à la caméra c et le nombre de caméras voisines. Supposons que c possède l régions E/S et que chaque région est liée à k régions dans les p caméras voisines. Dans le pire cas, la caméra c doit avoir en mémoire :

- les délimitations des l régions de c. Dans le cas où les régions sont représentées par des rectangles, la délimitation d'une région est un vecteur de 4 coordonnées de n bits. n dépend de la taille de la région. Une taille mémoire de $4ln$ bits est nécessaire.
- les identifiants des lk régions E/S des caméras voisines. Si chaque identifiant est codé sur 8 bits, la taille mémoire est $8lk$ bits.
- les $lk + l$ probabilités de transition. Chaque probabilité de transition est codée par un flottant sur 32 bits. Une mémoire de $32l(k + 1)$ bits est nécessaire.
- les $lk + l$ probabilités de temps de transition. La modélisation de la probabilité de temps de transition par une loi normale revient donc à stocker la moyenne et la variance de cette loi. Si la moyenne et la variance possèdent des dynamiques de 16 bits, la taille mémoire pour stocker une probabilité de temps de transition est de 32 bits. Une moyenne sur 16 bits peut contenir un temps de transition maximum de 65535 secondes. La taille mémoire nécessaire est $32l(k+1)$ bits. La modélisation de la probabilité par une loi paramétrique permet de gagner en mémoire par rapport à une loi non-paramétrique.
- les p probabilités de correspondance d'apparence. La modélisation de la probabilité de correspondance d'apparence par une loi normale nécessite le stockage de la moyenne et de la variance. Si nous supposons que la moyenne et la variance sont codées avec des flottants sur 32 bits, la taille mémoire pour stocker une probabilité de correspondance d'apparence est 64 bits. La taille nécessaire est $64p$ bits.

La mémoire totale nécessaire pour la caméra c est $(4n + 64)l + 72lk + 64p$ bits. Si nous considérons l'exemple de la figure 7.5, la caméra c_2 doit avoir en mémoire :

- les délimitations des régions $r_{3,2}$ et $r_{4,2}$. Dans notre cas, $n = 8$ est suffisant, car les images des caméras sont supposées avoir une résolution 320×240. La taille mémoire, dans ce cas, est de 64 bits.
- les identifiants des régions $r_{2,1}$, $r_{5,3}$ et $r_{7,4}$. Une mémoire de 24 bits est nécessaire dans ce cas.
- les probabilités de transitions : $\mathrm{P}\left(o_{(r_{2,1})}\middle|o'_{(r_{3,2})}\right)$, $\mathrm{P}\left(o_{(r_{7,4})}\middle|o'_{(r_{3,2})}\right)$, $\mathrm{P}\left(o_{(r_{5,3})}\middle|o'_{(r_{4,2})}\right)$, $\mathrm{P}\left(o_{(r_{3,2})}\middle|o'_{(r_{3,2})}\right)$ et $\mathrm{P}\left(o_{(r_{4,2})}\middle|o'_{(r_{4,2})}\right)$. Une mémoire de 160 bits est nécessaire.
- les lois modélisant les temps de transition : $\mathrm{P}\left(\Delta t\middle|o'_{(r_{3,2})}, o_{(r_{2,1})}\right)$, $\mathrm{P}\left(\Delta t\middle|o'_{(r_{3,2})}, o_{(r_{7,4})}\right)$, $\mathrm{P}\left(\Delta t\middle|o'_{(r_{4,2})}, o_{(r_{5,3})}\right)$, $\mathrm{P}\left(\Delta t\middle|o'_{(r_{3,2})}, o_{(r_{3,2})}\right)$ et $\mathrm{P}\left(\Delta t\middle|o'_{(r_{4,2})}, o_{(r_{4,2})}\right)$. La mémoire totale est 160 bits.

— les lois modélisant la correspondance d'apparence : $\mathrm{P}\left(o_{(App_{1,o})}\middle|o'_{(App_{2,o'})}\right)$, $\mathrm{P}\left(o_{(App_{3,o})}\middle|o'_{(App_{2,o'})}\right)$ $\mathrm{P}\left(o_{(App_{4,o})}\middle|o'_{(App_{2,o'})}\right)$. La mémoire totale est 192 bits.

Ce qui fait une mémoire totale de 75 octets et qui reste faible même pour un calculateur embarqué. Chaque caméra doit stocker les messages reçus des caméras voisines tant que les objets attendus ne sont pas ré-identifiés. Comme montrer dans la section 7.5.1, la taille de ce message est inférieur à 300 octets.

7.5.3 Calcul

Une fois le message d'une caméra voisine est reçu, la région d'attente est défini à partir de la région d'où l'objet est sortie. Cette opération s'effectue en accédant à l'élément d'un tableau. Chaque observation non associée aux pistes présentes dans la caméra est testée si elle appartient à une région E/S. Comme les régions E/S sont des rectangles, le test revient à comparer les coordonnées de l'observation aux coordonnées délimitant les régions. La complexité de cette étape dépend du nombre de régions E/S de la caméra.

Le calcul de la probabilité de correspondance entre deux objets, revient à calculer la distance de Bhattacharyya entre deux histogrammes et à évaluer deux lois normales de dimension égale à un. Le nombre des probabilités de correspondance à calculer dépend du nombre et de mouvement des objets dans le réseau. Le calcul ne s'effectue qu'une fois une observation non associée est détectée dans la région d'attente de l'objet en transition, ce qui permet d'éviter des calculs sans intérêt. Enfin, la complexité de la méthode dépend particulièrement de la taille du vecteur d'apparence, du nombre de régions E/S et du nombre et du mouvement des objets, mais elle reste tout de même acceptable pour des architectures embarquées de type Raspberry Pi.

La taille du message communiqué entre les caméras, la mémoire requise par chaque caméra et la complexité d'évaluation de la probabilité de correspondance sont en concordance avec les contraintes des réseaux de caméras intelligentes embarquées caractérisées par des calculateurs de faible puissance de calcul, faible capacité mémoire et faible bande passante pour les communications entre les noeuds.

7.6 Conclusion

Dans ce chapitre, nous avons présenté une approche distribuée s'intégrant avec le filtre GMPHD pour le suivi multi-objet dans un réseau de caméras avec des champs de vision non recouvrants. Après l'analyse des méthodes de la littérature, nous avons détaillé notre approche. L'approche développée est basée sur le calcul de la probabilité de correspondance d'un objet ayant disparu d'une région d'une caméra pour ré-apparaître dans une région d'une autre caméra. La probabilité de correspondance est calculée en utilisant la caractéristique d'apparence de l'objet, le temps de transition entre régions et la probabilité de transition entre régions. Notre approche est testée en simulation. Le scénario de simulation utilisé consiste en un couloir en forme de T qui contient quatre caméras. Les résultats obtenus sont prometteurs pour assurer un fonctionnement correct

sur des bases de données d'images et dans un réseau de caméras réel. Enfin, la complexité de la méthode a été estimée en analysant la taille des messages communiqués, la mémoire nécessaire et le calcul de la probabilité de correspondance. Les estimations montrent que la taille des messages est faible et peu de mémoire est nécessaire pour chaque caméra pour réaliser la ré-identification. Ces estimations sont en concordance pour un fonctionnement sur des plates-formes embarquées et dans un réseau de caméras avec une faible bande passante.

À présent, nous sommes confiants sur la faisabilité du déploiement d'un réseau de caméras avec des champs de vision non recouvrants où chaque noeud est un caméra intelligente de faible coût et de basse consommation et où la communication entre caméras s'effectue avec une faible bande passante.

Chapitre 8

Conclusion et perspectives

8.1 Conclusion

Compte tenu du nombre d'applications exploitant le suivi multi-objet, et des contraintes en terme de performances, du coût de déploiement et de consommation en énergie imposées par ces applications, l'utilisation d'architectures embarquées de vision devient primordial. Les architectures de vision embarquées sont efficaces en énergie et en encombrement mais possèdent des puissances de calcul et des capacités de mémoire réduites, ce qui introduit des challenges supplémentaires pour assurer un suivi en temps réel. Dans ce contexte, l'objectif de la thèse a été de concevoir un système de suivi multi-objet fonctionnant en temps réel sur des caméras avec un traitement embarqué et pouvant être déployé dans un réseau de caméras avec des champs de vision non recouvrants. Le but du système est d'assurer un bonne qualité de suivi avec une complexité de calcul adaptée à une architecture de calcul embarquée caractérisée par une consommation modérée et un faible coût. Nous avons donc développé une chaîne algorithmique de suivi multi-objet pour une seule caméra permettant la résolution d'occultations entre objets. Cette chaîne a été optimisée et portée sur une caméra intelligente embarquée composée de la carte de traitement Raspberry Pi version 1 et de la caméra RaspiCam. La chaîne s'exécute en temps réel sur la caméra intelligente tout en assurant une qualité de suivi proche des algorithmes de l'état de l'art. Nous avons proposé une méthode distribuée pour l'extension de suivi aux réseaux de caméras avec des champs de vision non recouvrants. Nous avons démontré la faisabilité de la méthode par simulation.

La chaîne algorithmique est composée de deux blocs : la détection basée sur la soustraction de fond et sur l'analyse des composantes connectées, et le suivi en utilisant une méthode probabiliste. Les deux blocs ont été sélectionnés après une analyse de l'état de l'art. La soustraction de fond développée est une fusion hiérarchique du résultat de segmentation fourni par l'algorithme Zipfian Sigma-Delta et de l'information du gradient de l'image d'entrée. Cette fusion a montré une amélioration de la qualité de détection au coût d'une faible complexité de calcul et de mémoire. Après l'étude et l'analyse des algorithmes de suivi multi-objet nous avons choisi le filtre probabiliste GMPHD qui présente un bon compromis entre la qualité de suivi et la complexité de calcul. Une étude de dégradation du suivi pour une implémentation embarquée du filtre GMPHD a été réalisée. Le but de cette expérimentation a été d'estimer le gain en vitesse d'exécution sur un processeur embarqué et de connaître les limites du filtre.

Le test du filtre GMPHD dans un scénario de suivi visuel d'objets a montré une limitation aux occultations. Nous avons donc amélioré le filtre pour résoudre les occultations se produisant entre les objets. La méthode de résolutions des occultations a été développée en tenant compte des contraintes des systèmes embarqués. En effet, aucun historique sur

les objets n'est nécessaire et seulement des caractéristiques simples comme l'histogramme en niveau de gris ont été utilisées pour la discrimination entre les objets quand des occultations se produisent. Nous avons optimisé toute la chaîne de suivi pour assurer un traitement en temps réel (15-30 fps) sur la caméra intelligente à traitement embarqué.

L'extension de la méthode de suivi aux réseaux de caméras avec des champs de vision non recouvrants s'est effectuée en proposant un système probabiliste distribué permettant la ré-identification des objets se déplaçant dans le réseau. La ré-identification s'effectue en calculant la probabilité de correspondance entre les caractéristiques des objets. La probabilité de correspondance se compose de la probabilité spatio-temporelle et la probabilité d'apparence. La formulation de ces probabilités exploitent la connaissance a priori du réseau ce qui permet d'assurer des bonnes performances de ré-identification en utilisant des caractéristiques simples comme le temps de transition et l'histogramme d'apparence. La nature probabiliste de l'approche développée assure une bonne intégration dans le filtre GMPHD. Le test de notre approche est réalisé en simulation, en supposant que les formes des probabilités spatio-temporelle et d'apparence sont connues. Cependant, ces probabilités peuvent être obtenues par apprentissage. Les résultats obtenus sont prometteurs pour une future réalisation réelle.

8.2 Perspectives

Les résultats obtenus dans la thèse ouvrent la porte à deux perspectives : algorithmique et architecture de calcul.

8.2.1 Perspectives algorithmiques

Dans le but d'améliorer la qualité de suivi, nous envisageons l'utilisation d'un étage de classification entre l'étage de détection et l'étage de suivi. L'objectif est d'arriver à séparer les différentes classes d'objets tout en améliorant la qualité de détection. Afin de respecter les contraintes des systèmes embarqués, la classification s'effectuera sur le résultat de soustraction de fond, ce qui permettra de diminuer la complexité de calcul engendrée par le parcours de l'image avec une fenêtre glissante à différentes échelles. Les travaux réalisés dans [Kristof et al., 2015] ont montré qu'un classifieur par apprentissage de type cascaded DPM (Deformable Part Models) [Felzenszwalb et al., 2010a] appliqué sur le résultat de la soustraction de fond fourni des bons résultats de détection tout en assurant un fonctionnement en temps réel sur un ordinateur personnel. Une analyse des classifieurs de la littérature permettra de choisir un classifieur qui est adapté aux systèmes embarqués et qui permet d'assurer un bon compromis entre la complexité de calcul et la qualité de classification.

Sur la partie multi-caméra, une étude des protocoles de communication entre les noeuds du réseau peut s'avérer utile pour diminuer la consommation énergétique. De plus, il est intéressent d'étudier l'extension de l'approche aux réseaux de caméras avec des champs de vision recouvrants. La nature probabiliste du filtre GMPHD permettra d'assurer une fusion synchrone distribuée en utilisant des filtres de *Consensus* [Olfati-Saber, 2005] ou une fusion asynchrone centralisée en traitant à la volée les observations.

8.2.2 Perspective d'architecture de calcul

A court terme, il est intéressant de tester la chaîne algorithmique de suivi multi-objet sur des caméras dont l'unité de traitement est plus contraignante : faible capacité de mémoire (<1 Mo), faible fréquence d'horloge (< 500 MHz) et pas d'extensions matérielles comme une FPU ce qui nous permettra d'utiliser les travaux d'approximation du filtre GMPHD.

Sur le long terme, la conception d'un co-processeur pour accélérer l'étage de détection qui consomme une grande partie du temps de traitement permettra de diminuer le temps de calcul. L'objectif de cette étude sera d'arriver à concevoir une caméra intelligente dédiée à la chaîne algorithmique proposée en utilisant un co-processeur pour l'étage de détection et un petit micro-contrôleur pour l'étage de suivi. Une conception dédiée permettra de diminuer le coût et la consommation énergétique d'un noeud du réseau de caméras sans dégradation des performances.

Bibliographie

[Adeli et al., 2012] Adeli, A., Yazdian-Dehkordi, M., Azimifar, Z., and Rojhani, O. (2012). Occluded targets tracking using improved gm-phd tracker. In *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, volume 2, pages 1071–1075. (Cité en pages 82 et 88.)

[Ahonen et al., 2006] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns : Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12) :2037–2041. (Cité en page 12.)

[Alahi et al., 2011] Alahi, A., Jacques, L., Boursier, Y., and Vandergheynst, P. (2011). Sparsity driven people localization with a heterogeneous network of cameras. *Journal of Mathematical Imaging and Vision*, 41(1-2) :39–58. (Cité en page 19.)

[Alahi et al., 2012] Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak : Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. Ieee. (Cité en page 12.)

[Alahi et al., 2010] Alahi, A., Vandergheynst, P., Bierlaire, M., and Kunt, M. (2010). Cascade of descriptors to detect and track objects across any network of cameras. *Computer Vision and Image Understanding*, 114(6) :624–640. (Cité en pages 22 et 116.)

[Amditis et al., 2012] Amditis, A., Thomaidis, G., Karaseitanidis, G., Lytrivis, P., and Maroudis, P. (2012). *Multiple Hypothesis Tracking Implementation*. INTECH Open Access Publisher. (Cité en page 60.)

[An et al., 2009] An, S., Peursum, P., Liu, W., and Venkatesh, S. (2009). Efficient algorithms for subwindow search in object detection and localization. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 264–271. (Cité en page 13.)

[Andriluka et al., 2009] Andriluka, M., Roth, S., and Schiele, B. (2009). Pictorial structures revisited : People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. (Cité en page 14.)

[Araki et al., 2000] Araki, S., Matsuoka, T., Yokoya, N., and Takemura, H. (2000). Real-time tracking of multiple moving object contours in a moving camera image sequence. *IEICE TRANSACTIONS on Information and Systems*, 83(7) :1583–1591. (Cité en page 18.)

[Arth et al., 2006] Arth, C., Bischof, H., and Leistner, C. (2006). Tricam - an embedded platform for remote traffic surveillance. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 125–125. (Cité en page 24.)

[Arth et al., 2007] Arth, C., Leistner, C., and Bischof, H. (2007). Object reacquisition and tracking in large-scale smart camera networks. In *Distributed Smart Cameras, 2007. ICDSC'07. First ACM/IEEE International Conference on*, pages 156–163. IEEE. (Cité en page 22.)

[Arulampalam et al., 2002] Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2) :174–188. (Cité en pages 15 et 17.)

[Aslani and Mahdavi-Nasab, 2013] Aslani, S. and Mahdavi-Nasab, H. (2013). Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical, Computer, Electronics and Communication Engineering*, 7(9) :761 – 765. (Cité en pages 11 et 12.)

[Balan and Black, 2006] Balan, A. O. and Black, M. J. (2006). An adaptive appearance model approach for model-based articulated object tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 758–765. IEEE. (Cité en page 14.)

[Bar-Shalom et al., 2009] Bar-Shalom, Y., Daum, F., and Huang, J. (2009). The probabilistic data association filter. *Control Systems, IEEE*, 29(6) :82 –100. (Cité en page 56.)

[Bar-Shalom and Li, 1995] Bar-Shalom, Y. and Li, X.-R. (1995). *Multitarget-multisensor tracking : Principles and techniques*. YBS publishing, Storrs, CT. (Cité en pages 16 et 56.)

[Barnich and Van Droogenbroeck, 2011] Barnich, O. and Van Droogenbroeck, M. (2011). Vibe : A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Transactions on*, 20(6) :1709–1724. (Cité en pages 8, 28, 36, 37 et 38.)

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf : Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer. (Cité en page 12.)

[Beauchemin and Barron, 1995] Beauchemin, S. S. and Barron, J. L. (1995). The computation of optical flow. *ACM Comput. Surv.*, 27(3) :433–466. (Cité en pages 9 et 10.)

[Benenson et al., 2012] Benenson, R., Mathias, M., Timofte, R., and Van Gool, L. (2012). Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910. IEEE. (Cité en page 13.)

[Benezeth et al., 2008] Benezeth, Y., Jodoin, P.-M., Emile, B., Laurent, H., and Rosenberger, C. (2008). Review and evaluation of commonly-implemented background subtraction algorithms. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. (Cité en pages 7, 8, 9 et 35.)

[Blackman, 2004] Blackman, S. (2004). Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1) :5–18. (Cité en pages 16, 58 et 86.)

[Blackman and Popoli, 1999] Blackman, S. and Popoli, R. (1999). *Design and Analysis of Modern Tracking Systems*. Artech House, Norwood, MA. (Cité en pages 16, 54, 55 et 63.)

[Bourgeois and Lassalle, 1971] Bourgeois, F. and Lassalle, J.-C. (1971). An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Commun. ACM*, 14(12) :802–804. (Cité en pages 16 et 55.)

[Bouwmans and El Baf, 2009] Bouwmans, T. and El Baf, F. (2009). Modeling of dynamic backgrounds by type-2 fuzzy gaussians mixture models. *MASAUM Journal of of Basic and Applied Sciences*, 1(2) :265–276. (Cité en page 35.)

[Bouwmans et al., 2008] Bouwmans, T., El Baf, F., and Vachon, B. (2008). Background modeling using mixture of gaussians for foreground detection-a survey. *Recent Patents on Computer Science*, 1(3) :219–237. (Cité en page 35.)

[Bramberger et al., 2006] Bramberger, M., Doblander, A., Maier, A., Rinner, B., and Schwabach, H. (2006). Distributed embedded smart cameras for surveillance applications. *Computer*, 39(2) :68–75. (Cité en pages 23 et 24.)

[Burton and Radford, 1978] Burton, A. and Radford, J. (1978). *Thinking in Perspective : Critical Essays in the Study of Thought Processes*. Psychology in progress. Methuen. (Cité en page 9.)

[Calderara et al., 2006] Calderara, S., Melli, R., Prati, A., and Cucchiara, R. (2006). Reliable background suppression for complex scenes. In *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, VSSN '06, pages 211–214, New York, NY, USA. ACM. (Cité en page 35.)

[Casares, 2014] Casares, M. (2014). *Energy-efficient lightweight algorithms for embedded smart cameras : design, implementation and performance analysis*. PhD thesis, Syracuse University. (Cité en page 38.)

[Casares and Velipasalar, 2010] Casares, M. and Velipasalar, S. (2010). Resource-efficient salient foreground detection for embedded smart cameras br tracking feedback. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 369–375. IEEE. (Cité en pages 7, 103 et 112.)

[Casares et al., 2010] Casares, M., Velipasalar, S., and Pinto, A. (2010). Light-weight salient foreground detection for embedded smart cameras. *Computer Vision and Image Understanding*, 114(11) :1223–1237. (Cité en pages 7, 38 et 103.)

[Chase et al., 2008] Chase, J., Nelson, B., Bodily, J., Wei, Z., and Lee, D.-J. (2008). Real-time optical flow calculations on fpga and gpu architectures : a comparison study. In *Field-Programmable Custom Computing Machines, 2008. FCCM'08. 16th International Symposium on*, pages 173–182. IEEE. (Cité en page 28.)

[Chen et al., 2008] Chen, P., Ahammad, P., Boyer, C., Huang, S.-I., Lin, L., Lobaton, E., Meingast, M., Oh, S., Wang, S., Yan, P., Yang, A., Yeo, C., Chang, L.-C., Tygar, J., and Sastry, S. (2008). Citric : A low-bandwidth wireless camera network platform. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–10. (Cité en page 24.)

[Chen et al., 2013a] Chen, X., An, L., and Bhanu, B. (2013a). Reference set based appearance model for tracking across non-overlapping cameras. In *Distributed Smart Cameras (ICDSC), 2013 Seventh International Conference on*, pages 1–6. IEEE. (Cité en pages 21 et 22.)

[Chen et al., 2013b] Chen, X., Huang, K., and Tan, T. (2013b). Object tracking across non-overlapping cameras using adaptive models. In *Computer Vision-ACCV 2012 Workshops*, pages 464–477. Springer. (Cité en pages 21, 22 et 116.)

[Clark et al., 2006a] Clark, D., Panta, K., and Vo, B.-N. (2006a). The gm-phd filter multiple target tracker. In *Information Fusion, 2006 9th International Conference on*, pages 1–8. (Cité en pages 68, 69 et 73.)

[Clark et al., 2006b] Clark, D., Vo, B.-N., and Bell, J. (2006b). Gm-phd filter multitarget tracking in sonar images. In *Defense and Security Symposium*, pages 62350R–62350R. International Society for Optics and Photonics. (Cité en pages 73, 91 et 93.)

[Clark and Bell, 2007] Clark, D. E. and Bell, J. (2007). Multi-target state estimation and track continuity for the particle phd filter. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(4) :1441–1453. (Cité en page 62.)

[Coifman et al., 1998] Coifman, B., Beymer, D., McLauchlan, P., and Malik, J. (1998). A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C : Emerging Technologies*, 6(4) :271–288. (Cité en page 14.)

[Comaniciu et al., 2000] Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149 vol.2. (Cité en pages 14 et 15.)

[Cong et al., 2010] Cong, D.-N. T., Khoudour, L., and Achard, C. (2010). People reacquisition across multiple cameras with disjoint views. In *Image and Signal Processing*, pages 488–495. Springer. (Cité en pages 22 et 116.)

[Conte et al., 2010] Conte, D., Foggia, P., Percannella, G., and Vento, M. (2010). Performance evaluation of a people tracking system on pets2009 database. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 119–126. IEEE. (Cité en pages 104, 107, 110 et 112.)

[Cox and Hingorani, 1996] Cox, I. J. and Hingorani, S. L. (1996). An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(2) :138–150. (Cité en pages 14, 16 et 58.)

[Cucchiara et al., 2001] Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2001). Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, pages 360–365. (Cité en page 7.)

[Cucchiara et al., 2004] Cucchiara, R., Grana, C., Tardini, G., and Vezzani, R. (2004). Probabilistic people tracking for occlusion handling. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 132–135 Vol.1. (Cité en page 19.)

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE. (Cité en page 12.)

[Dash et al., 2014] Dash, P. P., Patra, D., and Mishra, S. K. (2014). Local binary pattern as a texture feature descriptor in object tracking algorithm. In *Intelligent Computing, Networking, and Informatics*, pages 541–548. Springer. (Cité en page 14.)

[Dias et al., 2007] Dias, F., Berry, F., Serot, J., and Marmoiton, F. (2007). Hardware, design and implementation issues on a fpga-based smart camera. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 20–26. (Cité en pages 22 et 24.)

[Dick and Brooks, 2005] Dick, A. R. and Brooks, M. J. (2005). A stochastic approach to tracking objects across multiple cameras. In *AI 2004 : Advances in Artificial Intelligence*, pages 160–170. Springer. (Cité en pages 21, 116, 118, 120 et 127.)

[Dollár et al., 2010] Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *BMVC*, volume 2, page 7. Citeseer. (Cité en pages 13 et 28.)

[Dollar et al., 2012] Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection : An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4) :743–761. (Cité en page 13.)

[Dziri et al., 2014] Dziri, A., Duranton, M., and Chapuis, R. (2014). Low complexity multi-target tracking for embedded systems. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–8. (Cité en page 2.)

[Dziri et al., 2015a] Dziri, A., Duranton, M., and Chapuis, R. (2015a). Low complexity multi-object tracking system dealing with occlusions. In *Computer Vision Theory and Applications (VISAPP), 2015 10th International Conference on*, pages 1–8. (Cité en page 2.)

[Dziri et al., 2015b] Dziri, A., Duranton, M., and Chapuis, R. (2015b). Reliable multi-object tracking dealing with occlusions for a smart camera. In *Distributed Smart Cameras (ICDSC), 2015 Fifth ACM/IEEE International Conference on*, pages 1–6. (Cité en page 2.)

[Eiselein et al., 2012] Eiselein, V., Arp, D., Pätzold, M., and Sikora, T. (2012). Real-time multi-human tracking using a probability hypothesis density filter and multiple detectors. In *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, pages 325–330. IEEE. (Cité en pages 82, 86, 87 et 92.)

[El Baf et al., 2008a] El Baf, F., Bouwmans, T., and Vachon, B. (2008a). Fuzzy integral for moving object detection. In *Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on*, pages 1729–1736. IEEE. (Cité en page 35.)

[El Baf et al., 2008b] El Baf, F., Bouwmans, T., and Vachon, B. (2008b). Type-2 fuzzy mixture of gaussians model : Application to background modeling. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y., Rhyne, T.-M., and Monroe, L., editors, *Advances in Visual Computing*, volume 5358 of *Lecture Notes in Computer Science*, pages 772–781. Springer Berlin Heidelberg. (Cité en pages 8 et 35.)

[El Baf et al., 2009] El Baf, F., Bouwmans, T., and Vachon, B. (2009). Fuzzy statistical modeling of dynamic backgrounds for moving object detection in infrared videos. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 60–65. IEEE. (Cité en page 35.)

[Elgammal et al., 2000] Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. In Vernon, D., editor, *Computer Vision - ECCV 2000*, volume 1843 of *Lecture Notes in Computer Science*, pages 751–767. Springer Berlin Heidelberg. (Cité en page 8.)

[Ellis and Ferryman, 2010] Ellis, A. and Ferryman, J. (2010). Pets2010 and pets2009 evaluation of results using individual ground truthed single views. In *Advanced Video*

and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on, pages 135–142. (Cité en pages 110 et 111.)

[Enzweiler and Gavrila, 2009] Enzweiler, M. and Gavrila, D. M. (2009). Monocular pedestrian detection : Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12) :2179–2195. (Cité en pages 6 et 13.)

[Felzenszwalb et al., 2010a] Felzenszwalb, P. F., Girshick, R. B., and McAllester, D. (2010a). Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE. (Cité en page 138.)

[Felzenszwalb et al., 2010b] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010b). Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9) :1627–1645. (Cité en page 12.)

[Fleck et al., 2006] Fleck, S., Busch, F., Biber, P., and Straber, W. (2006). 3d surveillance a distributed network of smart cameras for real-time tracking and its visualization in 3d. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 118–118. (Cité en pages 22 et 23.)

[Fleck and Strasser, 2005] Fleck, S. and Strasser, W. (2005). Adaptive probabilistic tracking embedded in a smart camera. In *In Proc IEEE Embedded Computer Vision Workshop (ECVW '05) in conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pages 134–134. (Cité en page 102.)

[Fleet and Weiss, 2006] Fleet, D. and Weiss, Y. (2006). Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*, pages 237–257. Springer. (Cité en pages 9 et 10.)

[Fleuret et al., 2008] Fleuret, F., Berclaz, J., Lengagne, R., and Fua, P. (2008). Multicamera people tracking with a probabilistic occupancy map. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2) :267–282. (Cité en pages 18 et 19.)

[Gabriel et al., 2003] Gabriel, P. F., Verly, J. G., Piater, J. H., and Genon, A. (2003). The state of the art in multiple object tracking under occlusion in video sequences. In *Advanced Concepts for Intelligent Vision Systems*, pages 166–173. (Cité en page 18.)

[Gall et al., 2012] Gall, J., Razavi, N., and Van Gool, L. (2012). An introduction to random forests for multi-class object detection. In Dellaert, F., Frahm, J.-M., Pollefeys, M., Leal-Taixé, L., and Rosenhahn, B., editors, *Outdoor and Large-Scale Real-World Scene Analysis*, volume 7474 of *Lecture Notes in Computer Science*, pages 243–263. Springer Berlin Heidelberg. (Cité en page 13.)

[Gao et al., 2011] Gao, T., Packer, B., and Koller, D. (2011). A segmentation-aware object detection model with occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1361–1368. (Cité en page 18.)

[Gavrila and Munder, 2007] Gavrila, D. M. and Munder, S. (2007). Multi-cue pedestrian detection and tracking from a moving vehicle. *International journal of computer vision*, 73(1) :41–59. (Cité en page 12.)

[Godbehere et al., 2012] Godbehere, A. B., Matsukawa, A., and Goldberg, K. (2012). Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *American Control Conference (ACC), 2012*, pages 4305–4312. IEEE. (Cité en pages 8 et 35.)

[Golbabaee et al., 2014] Golbabaee, M., Alahi, A., and Vandergheynst, P. (2014). Scoop : a real-time sparsity driven people localization algorithm. *Journal of mathematical imaging and vision*, 48(1) :160–175. (Cité en page 19.)

[Goyat et al., 2006] Goyat, Y., Chateau, T., Malaterre, L., and Trassoudaine, L. (2006). Vehicle trajectories evaluation by static video sensors. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 864–869. (Cité en pages 8 et 35.)

[Haritaoglu et al., 1999] Haritaoglu, I., Harwood, D., and Davis, L. (1999). Hydra : multiple people detection and tracking using silhouettes. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 280–285. (Cité en page 13.)

[Harzallah et al., 2009] Harzallah, H., Jurie, F., and Schmid, C. (2009). Combining efficient object localization and image classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 237–244. IEEE. (Cité en page 12.)

[He et al., 2011] He, L., Wang, Y., Velipasalar, S., and Gursoy, M. (2011). Human detection using mobile embedded smart cameras. In *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, pages 1–6. (Cité en page 28.)

[Hengstler et al., 2007] Hengstler, S., Prashanth, D., Fong, S., and Aghajan, H. (2007). Mesheye : A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 360–369. (Cité en page 22.)

[Hidayatullah and Konik, 2011] Hidayatullah, P. and Konik, H. (2011). Camshift improvement on multi-hue object and multi-object tracking. In *Visual Information Processing (EUVIP), 2011 3rd European Workshop on*, pages 143–148. (Cité en pages 17 et 102.)

[Hofmann et al., 2012] Hofmann, M., Tiefenbacher, P., and Rigoll, G. (2012). Background segmentation with feedback : The pixel-based adaptive segmenter. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 38–43. IEEE. (Cité en page 35.)

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics. (Cité en page 10.)

[Hoseinezhad et al., 2009] Hoseinezhad, R., Vo, B.-N., and Suter, D. (2009). Fast single-view people tracking. In *Cognitive Systems with Interactive Sensors (COGIS2009)*. (Cité en page 82.)

[Hu et al., 2006] Hu, W., Hu, M., Zhou, X., Tan, T., Lou, J., and Maybank, S. (2006). Principal axis-based correspondence between multiple cameras for people tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4) :663–671. (Cité en page 20.)

[Hu et al., 2004] Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C :*

Applications and Reviews, IEEE Transactions on, 34(3) :334–352. (Cité en pages 5 et 6.)

[Huang and Russell, 1997] Huang, T. and Russell, S. (1997). Object identification in a bayesian context. In *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence - Volume 2*, IJCAI'97, pages 1276–1282, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (Cité en pages 21, 116 et 117.)

[Jain and Jain, 1981] Jain, J. and Jain, A. (1981). Displacement measurement and its application in interframe image coding. *Communications, IEEE Transactions on*, 29(12) :1799–1808. (Cité en page 11.)

[Jakubowski and Pastuszak, 2013] Jakubowski, M. and Pastuszak, G. (2013). Block-based motion estimation algorithms : a survey. *Opto-Electronics Review*, 21(1) :86–102. (Cité en page 11.)

[Javed et al., 2003] Javed, O., Rasheed, Z., Shafique, K., and Shah, M. (2003). Tracking across multiple cameras with disjoint views. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 952–, Washington, DC, USA. IEEE Computer Society. (Cité en pages 21, 116, 117, 122, 127 et 129.)

[Javed et al., 2008] Javed, O., Shafique, K., Rasheed, Z., and Shah, M. (2008). Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. *Computer Vision and Image Understanding*, 109(2) :146–162. (Cité en pages 116 et 122.)

[Javed et al., 2002] Javed, O., Shafique, K., and Shah, M. (2002). A hierarchical approach to robust background subtraction using color and gradient information. In *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 22–27. (Cité en pages 8, 38 et 39.)

[Jaward et al., 2006] Jaward, M., Mihaylova, L., Canagarajah, N., and Bull, D. (2006). A data association algorithm for multiple object tracking in video sequences. In *Target Tracking : Algorithms and Applications, 2006. The IEE Seminar on (Ref. No. 2006/11359)*, pages 129–136. (Cité en page 16.)

[KaewTraKulPong and Bowden, 2002] KaewTraKulPong, P. and Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer. (Cité en pages 8 et 35.)

[Kalal et al., 2012] Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7) :1409–1422. (Cité en page 18.)

[Kamal et al., 2012] Kamal, A., Farrell, J., and Roy-Chowdhury, A. (2012). Information weighted consensus. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2732–2737. (Cité en page 20.)

[Kamal et al., 2013] Kamal, A. T., Farrell, J. A., and Roy-Chowdhury, A. K. (2013). Information consensus for distributed multi-target tracking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2403–2410. IEEE. (Cité en page 20.)

[Kandhalu et al., 2009] Kandhalu, A., Rowe, A., and Rajkumar, R. (2009). Dspcam : A camera sensor system for surveillance networks. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–7. (Cité en page 24.)

[Kass et al., 1988] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes : Active contour models. *International journal of computer vision*, 1(4) :321–331. (Cité en page 18.)

[Kasturi et al., 2009] Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., and Zhang, J. (2009). Framework for performance evaluation of face, text, and vehicle detection and tracking in video : Data, metrics, and protocol. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2) :319–336. (Cité en page 44.)

[Kettnaker and Zabih, 1999] Kettnaker, V. and Zabih, R. (1999). Bayesian multi-camera surveillance. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –259 Vol. 2. (Cité en pages 21 et 116.)

[Khan and Shah, 2006] Khan, S. and Shah, M. (2006). A multiview approach to tracking people in crowded scenes using a planar homography constraint. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision - ECCV 2006*, volume 3954 of *Lecture Notes in Computer Science*, pages 133–146. Springer Berlin Heidelberg. (Cité en page 19.)

[Khan and Shah, 2009] Khan, S. and Shah, M. (2009). Tracking multiple occluding people by localizing on multiple scene planes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3) :505–519. (Cité en page 19.)

[Kim et al., 2010] Kim, I., Choi, H., Yi, K., Choi, J., and Kong, S. (2010). Intelligent visual surveillance : A survey. *International Journal of Control, Automation and Systems*, 8(5) :926–939. (Cité en page 5.)

[Kim et al., 2005] Kim, K., Chalidabhongse, T. H., Harwood, D., and Davis, L. (2005). Real-time foreground–background segmentation using codebook model. *Real-time imaging*, 11(3) :172–185. (Cité en pages 36 et 38.)

[Kleihorst et al., 2006] Kleihorst, R., Schueler, B., Danilin, A., and Heijligers, M. (2006). Smart camera mote with high performance vision system. In *ACM SenSys 2006 Workshop on Distributed Smart Cameras (DSC 2006)*. (Cité en pages 22, 23 et 24.)

[Konstantinova and Alexiev, 2001] Konstantinova, P. and Alexiev, K. (2001). A comparison of two hypothesis generation algorithms in jpdaf multiple target tracking. In *Proceedings of the International Conference on Computer Systems and Technologies*, pages 18–5. (Cité en page 65.)

[Kristan et al., 2013] Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Porikli, F., Cehovin, L., Nebehay, G., Fernandez, G., Vojir, T., Gatt, A., Khajenezhad, A., Salahledin, A., Soltani-Farani, A., Zarezade, A., Petrosino, A., Milton, A., Bozorgtabar, B., Li, B., Chan, C. S., Heng, C., Ward, D., Kearney, D., Monekosso, D., Karaimer, H., Rabiee, H., Zhu, J., Gao, J., Xiao, J., Zhang, J., Xing, J., Huang, K., Lebeda, K., Cao, L., Maresca, M., Lim, M. K., El Helw, M., Felsberg, M., Remagnino, P., Bowden, R.,

Goecke, R., Stolkin, R., Lim, S., Maher, S., Poullot, S., Wong, S., Satoh, S., Chen, W., Hu, W., Zhang, X., Li, Y., and Niu, Z. (2013). The visual object tracking vot2013 challenge results. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 98–111. (Cité en page 5.)

[Kristof et al., 2015] Kristof, V. B., , and Toon, G. (2015). Low complexity multi-object tracking system dealing with occlusions. In *Computer Vision Theory and Applications (VISAPP), 2015 10th International Conference on*, pages 1–8. (Cité en page 138.)

[Kuo et al., 2010a] Kuo, C.-H., Huang, C., and Nevatia, R. (2010a). Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision - ECCV 2010*, volume 6311 of *Lecture Notes in Computer Science*, pages 383–396. Springer Berlin Heidelberg. (Cité en page 21.)

[Kuo et al., 2010b] Kuo, C.-H., Huang, C., and Nevatia, R. (2010b). Multi-target tracking by on-line learned discriminative appearance models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 685–692. IEEE. (Cité en pages 13 et 14.)

[Kwatra et al., 1987] Kwatra, S. C., Lin, C.-M., and Whyte, W. (1987). An adaptive algorithm for motion compensated color image coding. *Communications, IEEE Transactions on*, 35(7) :747–754. (Cité en page 11.)

[Lamard et al., 2013] Lamard, L., Chapuis, R., and Boyer, J. (2013). Multi target tracking with cphd filter based on asynchronous sensors. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 892–898. (Cité en page 17.)

[Lamard et al., 2012a] Lamard, L., Chapuis, R., and Boyer, J.-P. (2012a). A comparison of two different tracking algorithms is provided for real application. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 414–419. IEEE. (Cité en pages 64 et 65.)

[Lamard et al., 2012b] Lamard, L., Chapuis, R., and Boyer, J.-P. (2012b). Dealing with occlusions with multi targets tracking algorithms for the real road context. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 371–376. (Cité en pages 82, 87, 88 et 93.)

[Lampert et al., 2008] Lampert, C. H., Blaschko, M., and Hofmann, T. (2008). Beyond sliding windows : Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. (Cité en page 13.)

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk : Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE. (Cité en page 12.)

[Li and Zhang, 2013] Li, J. and Zhang, Y. (2013). Learning surf cascade for fast and accurate object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3468–3475. IEEE. (Cité en page 12.)

[Li et al., 2003] Li, L., Huang, W., Gu, I. Y., and Tian, Q. (2003). Foreground object detection from videos containing complex background. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10. ACM. (Cité en page 36.)

[Li et al., 2013] Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., and Hengel, A. V. D. (2013). A survey of appearance models in visual object tracking. *ACM Trans. Intell. Syst. Technol.*, 4(4) :58 :1–58 :48. (Cité en page 15.)

[Lin and Davis, 2008] Lin, Z. and Davis, L. S. (2008). A pose-invariant descriptor for human detection and segmentation. In *Computer Vision–ECCV 2008*, pages 423–436. Springer. (Cité en page 12.)

[Litzenberger et al., 2007] Litzenberger, M., Belbachir, A. N., Schon, P., and Posch, C. (2007). Embedded smart camera for high speed vision. In *Distributed Smart Cameras, 2007. ICDSC'07. First ACM/IEEE International Conference on*, pages 81–86. IEEE. (Cité en pages 22 et 24.)

[Litzenberger et al., 2006] Litzenberger, M., Posch, C., Bauer, D., Belbachir, A., Schon, P., Kohn, B., and Garn, H. (2006). Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. In *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th*, pages 173–178. (Cité en page 102.)

[Liu et al., 1998] Liu, H., Hong, T.-H., Herman, M., Camus, T., and Chellappa, R. (1998). Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer vision and image understanding*, 72(3) :271–286. (Cité en page 10.)

[Lucas et al., 1981] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679. (Cité en page 10.)

[Luo et al., 2014] Luo, W., Zhao, X., and Kim, T. (2014). Multiple object tracking : A review. *CoRR*, abs/1409.7618. (Cité en pages 5 et 15.)

[Ma et al., 2008] Ma, N., Bailey, D., and Johnston, C. (2008). Optimised single pass connected components analysis. In *ICECE Technology, 2008. FPT 2008. International Conference on*, pages 185–192. (Cité en pages 40 et 48.)

[Maddalena and Petrosino, 2008] Maddalena, L. and Petrosino, A. (2008). A self-organizing approach to background subtraction for visual surveillance applications. *Image Processing, IEEE Transactions on*, 17(7) :1168–1177. (Cité en page 35.)

[Maddalena and Petrosino, 2010] Maddalena, L. and Petrosino, A. (2010). A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection. *Neural Computing and Applications*, 19(2) :179–186. (Cité en page 35.)

[Maggio and Cavallaro, 2011] Maggio, D. E. and Cavallaro, D. A. (2011). *Video Tracking : Theory and Practice*. Wiley Publishing, 1st edition. (Cité en page 5.)

[Maggio et al., 2008] Maggio, E., Taj, M., and Cavallaro, A. (2008). Efficient multitarget visual tracking using random finite sets. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(8) :1016–1027. (Cité en pages 13 et 17.)

[Magno et al., 2008] Magno, M., Tombari, F., Brunelli, D., Di Stefano, L., and Benini, L. (2008). Multi-modal video surveillance aided by pyroelectric infrared sensors. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*. (Cité en page 22.)

[Mahler, 2003] Mahler, R. (2003). Multitarget bayes filtering via first-order multitarget moments. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4) :1152–1178. (Cité en page 17.)

[Mahler, 2007] Mahler, R. (2007). Phd filters of higher order in target number. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(4) :1523–1543. (Cité en page 17.)

[Manzanera, 2007] Manzanera, A. (2007). σ-δ background subtraction and the zipf law. In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 42–51. Springer. (Cité en pages 36, 39, 40 et 44.)

[Manzanera and Richefeu, 2007] Manzanera, A. and Richefeu, J. C. (2007). A new motion detection algorithm based on &#931 ;-&#916 ; background estimation. *Pattern Recogn. Lett.*, 28(3) :320–328. (Cité en page 7.)

[McFarlane and Schofield, 1995] McFarlane, N. and Schofield, C. (1995). Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3) :187–193. (Cité en pages 7 et 35.)

[Mosqueron et al., 2007] Mosqueron, R., Dubois, J., and Paindavoine, M. (2007). High-speed smart camera with high resolution. *EURASIP Journal on Embedded Systems*, 2007(1) :23–23. (Cité en page 22.)

[Nethercote et al., 2006] Nethercote, N., Walsh, R., and Fitzhardinge, J. (2006). Building workload characterization tools with valgrind. *2006 IEEE International Symposium on Workload Characterization (IISWC)*, page 2. (Cité en pages 49, 79, 80 et 111.)

[Nummiaro et al., 2002] Nummiaro, K., Koller-Meier, E., and Van Gool, L. (2002). Object tracking with an adaptive color-based particle filter. In *Pattern Recognition*, pages 353–360. Springer. (Cité en page 14.)

[Olfati-Saber, 2005] Olfati-Saber, R. (2005). Distributed kalman filter with embedded consensus filters. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 8179–8184. (Cité en pages 20 et 138.)

[Olfati-Saber, 2007] Olfati-Saber, R. (2007). Distributed kalman filtering for sensor networks. In *Decision and Control, 2007 46th IEEE Conference on*, pages 5492–5498. (Cité en page 20.)

[Oliver et al., 2000] Oliver, N., Rosario, B., and Pentland, A. (2000). A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8) :831–843. (Cité en pages 8, 35 et 38.)

[Panta et al., 2009] Panta, K., Clark, D. E., and Vo, B.-N. (2009). Data association and track management for the gaussian mixture probability hypothesis density filter. *Aerospace and Electronic Systems, IEEE Transactions on*, 45(3) :1003–1016. (Cité en pages 86 et 88.)

[Panta et al., 2004] Panta, K., Vo, B.-N., Singh, S., and Doucet, A. (2004). Probability hypothesis density filter versus multiple hypothesis tracking. In *Defense and Security*, pages 284–295. International Society for Optics and Photonics. (Cité en page 64.)

[Papageorgiou and Poggio, 2000] Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, 38(1) :15–33. (Cité en page 12.)

[Pham et al., 2011] Pham, N. T., Chang, R., Leman, K., Chua, T. W., and Wang, Y. (2011). Random finite set for data association in multiple camera tracking. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1357–1360. IEEE. (Cité en page 20.)

[Pham et al., 2007] Pham, N. T., Huang, W., and Ong, S. (2007). Tracking multiple objects using probability hypothesis density filter and color measurements. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1511–1514. (Cité en page 15.)

[Piao and Berns, 2014] Piao, S. and Berns, K. (2014). Multi-object tracking based on tracking-learning-detection framework. In Berns, K., Mehdi, S. A., and Muhammad, A., editors, *Field and Assistive Robotics - Advances in Systems and Algorithms*, pages 74–87. Shaker Verlag, Aachen. ISBN-13 : 978-3-8440-2753-2. (Cité en page 18.)

[Po and Ma, 1996] Po, L.-M. and Ma, W.-C. (1996). A novel four-step search algorithm for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3) :313–317. (Cité en page 11.)

[Poiesi and Cavallaro, 2015] Poiesi, F. and Cavallaro, A. (2015). Tracking multiple high-density homogeneous targets. *Circuits and Systems for Video Technology, IEEE Transactions on*, 25(4) :623–637. (Cité en pages 13, 14, 15, 44, 104, 107, 108, 110 et 112.)

[Porikli and Tuzel, 2005] Porikli, F. and Tuzel, O. (2005). Multi-kernel object tracking. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1234–1237. IEEE. (Cité en page 17.)

[Possegger et al., 2013] Possegger, H., Sternig, S., Mauthner, T., Roth, P., and Bischof, H. (2013). Robust real-time tracking of multiple objects by volumetric mass densities. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2395–2402. (Cité en page 19.)

[Prisacariu and Reid, 2009] Prisacariu, V. and Reid, I. (2009). fasthog-a real-time gpu implementation of hog. *Department of Engineering Science*, 2310. (Cité en page 13.)

[Quaritsch et al., 2007] Quaritsch, M., Kreuzthaler, M., Rinner, B., Bischof, H., and Strobl, B. (2007). Autonomous multicamera tracking on embedded smart cameras. *EURASIP J. Embedded Syst.*, 2007(1) :35–35. (Cité en page 102.)

[Radke, 2010] Radke, R. J. (2010). A survey of distributed computer vision algorithms. In *Handbook of Ambient Intelligence and Smart Environments*, pages 35–55. Springer. (Cité en page 18.)

[Rahimi et al., 2005] Rahimi, M., Baer, R., Iroezi, O. I., Garcia, J. C., Warrior, J., Estrin, D., and Srivastava, M. (2005). Cyclops : in situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 192–204. ACM. (Cité en pages 23 et 24.)

[Rinner et al., 2008] Rinner, B., Winkler, T., Schriebl, W., Quaritsch, M., and Wolf, W. (2008). The evolution from single to pervasive smart cameras. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–10. IEEE. (Cité en page 23.)

[Ristic et al., 2011] Ristic, B., Vo, B.-N., Clark, D., and Vo, B.-T. (2011). A metric for performance evaluation of multi-target tracking algorithms. *Signal Processing, IEEE Transactions on*, 59(7) :3452–3457. (Cité en page 77.)

[Rowe et al., 2007] Rowe, A. G., Goode, A., Goel, D., and Nourbakhsh, I. (2007). Cmucam3 : An open programmable embedded vision sensor. (Cité en pages 23 et 24.)

[Sabzmeydani and Mori, 2007] Sabzmeydani, P. and Mori, G. (2007). Detecting pedestrians by learning shapelet features. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE. (Cité en page 12.)

[Sato and Aggarwal, 2004] Sato, K. and Aggarwal, J. K. (2004). Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vis. Image Underst.*, 96(2) :100–128. (Cité en page 18.)

[Schaeffer, 2013] Schaeffer, C. (2013). A comparison of keypoint descriptors in the context of pedestrian detection : Freak vs. surf vs. brisk. (Cité en page 12.)

[Schwartz et al., 2009] Schwartz, W. R., Kembhavi, A., Harwood, D., and Davis, L. S. (2009). Human detection using partial least squares analysis. In *Computer vision, 2009 IEEE 12th international conference on*, pages 24–31. IEEE. (Cité en page 12.)

[Seema and Reisslein, 2011] Seema, A. and Reisslein, M. (2011). Towards efficient wireless video sensor networks : A survey of existing node architectures and proposal for a flexi-wvsnp design. *Communications Surveys & Tutorials, IEEE*, 13(3) :462–486. (Cité en page 23.)

[Senior et al., 2006] Senior, A., Hampapur, A., Tian, Y.-L., Brown, L., Pankanti, S., and Bolle, R. (2006). Appearance models for occlusion handling. *Image and Vision Computing*, 24(11) :1233–1243. (Cité en page 18.)

[Shafique and Shah, 2005] Shafique, K. and Shah, M. (2005). A noniterative greedy algorithm for multiframe point correspondence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(1) :51–65. (Cité en pages 15 et 53.)

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600. (Cité en page 14.)

[Shin et al., 2005] Shin, J., Kim, S., Kang, S., Lee, S.-W., Paik, J., Abidi, B., and Abidi, M. (2005). Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging*, 11(3) :204–218. (Cité en pages 11 et 14.)

[Sidenbladh, 2003] Sidenbladh, H. (2003). Multi-target particle filtering for the probability hypothesis density. In *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, volume 2, pages 800–806. (Cité en pages 61 et 64.)

[Sigari et al., 2008] Sigari, M., Mozayani, N., and Pourreza, H. (2008). Fuzzy running average and fuzzy background subtraction : concepts and application. *International Journal of Computer Science and Network Security*, 8(2) :138–143. (Cité en page 35.)

[SmartCamera, 2008] SmartCamera (2008). http ://www.smartcamera.it/links.htm. In *Vu mai 2015*. (Cité en page 23.)

[Sobral, 2013] Sobral, A. (2013). BGSLibrary : An opencv c++ background subtraction library. In *IX Workshop de Visão Computacional (WVC'2013)*, Rio de Janeiro, Brazil. (Cité en page 8.)

[Sobral and Vacavant, 2014] Sobral, A. and Vacavant, A. (2014). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122(0) :4 – 21. (Cité en pages 8, 34, 35, 37 et 44.)

[Song et al., 2011] Song, B., Ding, C., Kamal, A., Farrell, J., and Roy-Chowdhury, A. (2011). Distributed camera networks. (Cité en page 20.)

[Song et al., 2010] Song, B., Kamal, A., Soto, C., Ding, C., Farrell, J., and Roy-Chowdhury, A. (2010). Tracking and activity recognition through consensus in distributed camera networks. *Image Processing, IEEE Transactions on*, 19(10) :2564–2579. (Cité en page 20.)

[Song and Roy-Chowdhury, 2008] Song, B. and Roy-Chowdhury, A. (2008). Robust tracking in a camera network : A multi-objective optimization framework. *Selected Topics in Signal Processing, IEEE Journal of*, 2(4) :582–596. (Cité en page 21.)

[Song et al., 2013] Song, M., Tao, D., and Maybank, S. J. (2013). Sparse camera network for visual surveillance–a comprehensive survey. *arXiv preprint arXiv :1302.0446*. (Cité en page 5.)

[Soto et al., 2009] Soto, C., Song, B., and Roy-Chowdhury, A. K. (2009). Distributed multi-target tracking in a self-configuring camera network. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1486–1493. IEEE. (Cité en page 20.)

[Stauffer and Grimson, 1999] Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –252 Vol. 2. (Cité en pages 7, 35 et 38.)

[Sternig et al., 2011] Sternig, S., Mauthner, T., Irschara, A., Roth, P., and Bischof, H. (2011). Multi-camera multi-object tracking by robust hough-based homography projections. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1689–1696. (Cité en page 19.)

[Sun et al., 2011] Sun, X., Yao, H., and Zhang, S. (2011). A novel supervised level set method for non-rigid object tracking. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3393–3400. IEEE. (Cité en page 13.)

[Szarvas et al., 2005] Szarvas, M., Yoshizawa, A., Yamamoto, M., and Ogata, J. (2005). Pedestrian detection with convolutional neural networks. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 224–229. (Cité en page 13.)

[Szegedy et al., 2013] Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561. (Cité en page 13.)

[Taj and Cavallaro, 2011] Taj, M. and Cavallaro, A. (2011). Distributed and decentralized multicamera tracking. *Signal Processing Magazine, IEEE*, 28(3) :46–58. (Cité en page 20.)

[Taj et al., 2007] Taj, M., Maggio, E., and Cavallaro, A. (2007). Multi-feature graph-based object tracking. In *Multimodal Technologies for Perception of Humans*, pages 190–199. Springer. (Cité en page 15.)

[Takala and Pietikainen, 2007] Takala, V. and Pietikainen, M. (2007). Multi-object tracking using color, texture and motion. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE. (Cité en page 14.)

[Tang et al., 2012] Tang, D., Liu, Y., and Kim, T.-K. (2012). Fast pedestrian detection by cascaded random forest with dominant orientation templates. In *BMVC*, pages 1–11. (Cité en pages 12 et 13.)

[Truong Cong et al., 2009] Truong Cong, D. N., Achard, C., Khoudour, L., and Douadi, L. (2009). Video sequences association for people re-identification across multiple non-overlapping cameras. In *Proceedings of the 15th International Conference on Image Analysis and Processing*, ICIAP '09, pages 179–189, Berlin, Heidelberg. Springer-Verlag. (Cité en page 116.)

[Ueberhuber, 1997] Ueberhuber, C. W. (1997). *Numerical Computation 1 : Methods, Software, and Analysis.*, volume 16. Springer Science & Business Media. (Cité en page 71.)

[Valera and Velastin, 2005] Valera, M. and Velastin, S. (2005). Intelligent distributed surveillance systems : a review. *Vision, Image and Signal Processing, IEE Proceedings* -, 152(2) :192–204. (Cité en page 5.)

[Velipasalar et al., 2008] Velipasalar, S., Schlessman, J., Chen, C.-Y., Wolf, W. H., and Singh, J. P. (2008). A scalable clustered camera system for multiple object tracking. *EURASIP Journal on Image and Video Processing*, 2008 :22. (Cité en page 103.)

[Vezzani et al., 2013] Vezzani, R., Baltieri, D., and Cucchiara, R. (2013). People reidentification in surveillance and forensics : A survey. *ACM Comput. Surv.*, 46(2) :29 :1–29 :37. (Cité en pages 5, 15 et 21.)

[Vijverberg et al., 2009] Vijverberg, J. A., Koeleman, C. J., and With, P. H. N. d. (2009). Tracking rectangular targets in surveillance videos with the gm-phd filter. In *Proceedings of the 30-th Symposium on Information Theory in the Benelux*, pages 177–184. (Cité en pages 82, 85 et 92.)

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2) :137–154. (Cité en page 12.)

[Vo and Ma, 2006] Vo, B.-N. and Ma, W.-K. (2006). The gaussian mixture probability hypothesis density filter. *Signal Processing, IEEE Transactions on*, 54(11) :4091–4104. (Cité en pages 62, 63, 68 et 73.)

[Vo et al., 2003] Vo, B.-N., Singh, S., and Doucet, A. (2003). Sequential monte carlo implementation of the phd filter for multi-target tracking. In *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, volume 2, pages 792–799. (Cité en page 61.)

[Vo et al., 2005] Vo, B.-N., Singh, S., and Doucet, A. (2005). Sequential monte carlo methods for multitarget filtering with random finite sets. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4) :1224–1245. (Cité en page 61.)

[Vo, 2008] Vo, B. T. (2008). *Random finite sets in multi-object filtering.* PhD thesis, Citeseer. (Cité en page 60.)

[Vo et al., 2007] Vo, B.-T., Vo, B.-N., and Cantoni, A. (2007). Analytic implementations of the cardinalized probability hypothesis density filter. *Signal Processing, IEEE Transactions on*, 55(7) :3553–3567. (Cité en pages 61, 62 et 64.)

[Walther, 1971] Walther, J. S. (1971). A unified algorithm for elementary functions. In *Proceedings of the May 18-20, 1971, spring joint computer conference*, pages 379–385. ACM. (Cité en page 73.)

[Wang et al., 2009a] Wang, H., Liu, C., Zhang, X., and Li, Q. (2009a). Multi-camera tracking based on information fusion in video surveillance. In *Image and Signal Processing, 2009. CISP'09. 2nd International Congress on*, pages 1–5. IEEE. (Cité en page 20.)

[Wang, 2013] Wang, X. (2013). Intelligent multi-camera video surveillance : A review. *Pattern Recogn. Lett.*, 34(1) :3–19. (Cité en pages 5 et 18.)

[Wang et al., 2009b] Wang, X., Han, T. X., and Yan, S. (2009b). An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE. (Cité en page 12.)

[Wang et al., 2010] Wang, Y., Velipasalar, S., and Casares, M. (2010). Cooperative object tracking and composite event detection with wireless embedded smart cameras. *Image Processing, IEEE Transactions on*, 19(10) :2614–2633. (Cité en pages 103 et 112.)

[Wang et al., 2014] Wang, Y., Velipasalar, S., and Gursoy, M. (2014). Distributed wide-area multi-object tracking with non-overlapping camera views. *Multimedia Tools and Applications*, 73(1) :7–39. (Cité en page 22.)

[Wang et al., 2011] Wang, Y., Velipasalar, S., and Gursoy, M. C. (2011). Wide-area multi-object tracking with non-overlapping camera views. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE. (Cité en pages 21 et 22.)

[Wang et al., 2007] Wang, Y.-D., Wu, J.-K., Huang, W., Kassim, A., et al. (2007). Gaussian mixture probability hypothesis density for visual people racking. In *Information Fusion, 2007 10th International Conference on*, pages 1–6. IEEE. (Cité en pages 73 et 82.)

[Wang et al., 2006] Wang, Y.-D., Wu, J.-K., Kassim, A., and Huang, W.-M. (2006). Tracking a variable number of human groups in video using probability hypothesis density. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1127–1130. (Cité en pages 13, 14, 15 et 82.)

[Winkler et al., 2014] Winkler, T., Erdelyi, A., and Rinner, B. (2014). Trusteye.m4 : Protecting the sensor - not the camera. In *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*, pages 159–164. (Cité en page 24.)

[Winkler and Rinner, 2010] Winkler, T. and Rinner, B. (2010). Trustcam : Security and privacy-protection for an embedded smart camera based on trusted computing. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 593–600. IEEE. (Cité en pages 22 et 24.)

[Wojek and Schiele, 2008] Wojek, C. and Schiele, B. (2008). A performance evaluation of single and multi-feature people detection. In *Pattern Recognition*, pages 82–91. Springer. (Cité en page 12.)

[Wren et al., 1997] Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfinder : real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7) :780–785. (Cité en pages 7 et 35.)

[Wu et al., 2008] Wu, B., Nevatia, R., and Li, Y. (2008). Segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. (Cité en page 18.)

[Wu and Hong, 2013] Wu, F. and Hong, W. D. (2013). A survey on the nodes of wireless multimedia sensor networks. (Cité en page 23.)

[Yamamoto et al., 1995] Yamamoto, S., Mae, Y., Shirai, Y., and Miura, J. (1995). Real-time multiple object tracking based on optical flows. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 3, pages 2328–2333 vol.3. (Cité en page 11.)

[Yan et al., 2013] Yan, X., Xu, D., and Yao, B. (2013). Large-scale surveillance system based on hybrid cooperative multi-camera tracking. *Open Journal of Applied Sciences*, 3(01) :79. (Cité en pages 22 et 23.)

[Yang and Nevatia, 2012] Yang, B. and Nevatia, R. (2012). Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1918–1925. (Cité en page 110.)

[Yang et al., 2005a] Yang, C., Duraiswami, R., and Davis, L. (2005a). Fast multiple object tracking via a hierarchical particle filter. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 212–219. IEEE. (Cité en pages 13 et 14.)

[Yang et al., 2005b] Yang, T., Pan, Q., Li, J., and Li, S. (2005b). Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 970–975. IEEE. (Cité en pages 18 et 19.)

[Yao and Odobez, 2007] Yao, J. and Odobez, J.-M. (2007). Multi-layer background subtraction based on color and texture. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE. (Cité en page 35.)

[Yazdian-Dehkordi et al., 2012] Yazdian-Dehkordi, M., Rojhani, O., and Azimifar, Z. (2012). Visual target tracking in occlusion condition : A gm-phd-based approach. In *Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on*, pages 538–541. (Cité en pages 82, 88 et 93.)

[Yilmaz et al., 2006] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking : A survey. *ACM Comput. Surv.*, 38(4). (Cité en pages 5, 6, 14 et 15.)

[Yilmaz et al., 2004] Yilmaz, A., Li, X., and Shah, M. (2004). Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11) :1531–1536. (Cité en page 13.)

[Yoon et al., 2015] Yoon, J. H., Yang, M.-H., Lim, J., and Yoon, K.-J. (2015). Bayesian multi-object tracking using motion context from multiple objects. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 33–40. (Cité en page 13.)

[Zhang et al., 2009a] Zhang, C., Qiao, Y., Fallon, E., and Xu, C. (2009a). An improved camshift algorithm for target tracking in video surveillance. (Cité en page 17.)

[Zhang et al., 2014] Zhang, F., Gao, Y., and Bakos, J. (2014). Lucas-kanade optical flow estimation on the ti c66x digital signal processor. In *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*, pages 1–6. (Cité en page 28.)

[Zhang and Xu, 2006] Zhang, H. and Xu, D. (2006). Fusing color and texture features for background model. In *Fuzzy Systems and Knowledge Discovery : Third International Conference, FSKD 2006, Xi'an, China, September 24-28, 2006. Proceedings*, pages 887–893. Springer. (Cité en page 35.)

[Zhang et al., 2013] Zhang, X., Yang, Y.-H., Han, Z., Wang, H., and Gao, C. (2013). Object class detection : A survey. *ACM Comput. Surv.*, 46(1) :10 :1–10 :53. (Cité en page 13.)

[Zhang et al., 2009b] Zhang, Z., Scanlon, A., Yin, W., Yu, L., and Venetianer, P. L. (2009b). Video surveillance using a multi-camera tracking and fusion system. *Multi-Camera Networks : Principles and Applications*, pages 435–456. (Cité en page 20.)

[Zhao et al., 2012] Zhao, Z., Bouwmans, T., Zhang, X., and Fang, Y. (2012). A fuzzy background modeling approach for motion detection in dynamic backgrounds. In Wang, F., Lei, J., Lau, R., and Zhang, J., editors, *Multimedia and Signal Processing*, volume 346 of *Communications in Computer and Information Science*, pages 177–185. Springer Berlin Heidelberg. (Cité en pages 8 et 35.)

[Zhu et al., 2010] Zhu, L., Chen, Y., Yuille, A., and Freeman, W. (2010). Latent hierarchical structural learning for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1062–1069. IEEE. (Cité en page 12.)

[Zivkovic, 2004] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 28–31, Washington, DC, USA. IEEE Computer Society. (Cité en pages 8 et 35.)

[Zivkovic and Krose, 2004] Zivkovic, Z. and Krose, B. (2004). An em-like algorithm for color-histogram-based object tracking. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–798–I–803 Vol.1. (Cité en page 14.)

[Zivkovic and van der Heijden, 2006] Zivkovic, Z. and van der Heijden, F. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7) :773–780. (Cité en page 35.)

[Zuriarrain et al., 2013] Zuriarrain, I., Mekonnen, A. A., Lerasle, F., and Arana, N. (2013). Tracking-by-detection of multiple persons by a resample-move particle filter. *Machine vision and applications*, 24(8) :1751–1765. (Cité en pages 13, 15 et 17.)