Université de Lyon — Université Lumière Lyon 2
École Doctorale InfoMaths

Numéro : |_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de
Docteur en Informatique

présentée par

Matthieu Rogez

Utilisation du contexte pour la détection et le suivi d'objets en vidéosurveillance

JURY

Rapporteurs	Thierry Chateau	Professeur	Université Clermont Ferrand II
	François Brémond	Professeur	INRIA Sophia Antipolis
Examinateurs	Jean-Philippe Domenger	Professeur	Université Bordeaux 1
	Thierry Bouwmans	Maître de Conférences (HDR)	Université de La Rochelle
	Antoine Vacavant	Maître de Conférences	Université Clermont Ferrand 1
Directeurs	Laure Tougne	Professeure	Université Lyon 2
	Lionel Robinault	Docteur	Foxstream

Résumé

Les caméras de surveillance sont de plus en plus fréquemment présentes dans notre environnement (villes, supermarchés, aéroports, entrepôts, *etc.*). Ces caméras sont utilisées, entre autres, afin de pouvoir détecter des comportements suspects (intrusion par exemple) ou de reconnaître une catégorie d'objets ou de personnes (détection de genre, détection de plaques d'immatriculation par exemple). D'autres applications concernent également l'établissement de statistiques de fréquentation ou de passage (comptage d'entrée/sortie de personnes ou de véhicules) ou bien le suivi d'un ou plusieurs objets se déplaçant dans le champ de vision de la caméra (trajectoires d'objets, analyse du comportement des clients dans un magasin). Compte tenu du nombre croissant de caméras et de la difficulté à réaliser ces traitements manuellement, un ensemble de méthodes d'analyse vidéo ont été développées ces dernières années afin de pouvoir automatiser ces tâches.

Dans cette thèse, nous nous concentrons essentiellement sur les tâches de détection et de suivi des objets mobiles à partir d'une caméra fixe. Contrairement aux méthodes basées uniquement sur les images acquises par les caméras, notre approche consiste à intégrer un certain nombre d'informations contextuelles à l'observation afin de pouvoir mieux interpréter ces images. Ainsi, nous proposons de construire un modèle géométrique et géolocalisé de la scène et de la caméra. Ce modèle est construit directement à partir des études de prédéploiement des caméras et peut notamment utiliser les données OpenStreetMap afin d'établir les modèles 3d des bâtiments proches de la caméra. Nous avons complété ce modèle en intégrant la possibilité de prédire la position du Soleil tout au long de la journée et ainsi pouvoir calculer les ombres projetées des objets de la scène. Cette prédiction des ombres a été mise à profit afin d'améliorer la segmentation des piétons par modèle de fond en supprimant les ombres du masque de mouvement.

Concernant le suivi des objets mobiles, nous utilisons le formalisme des automates finis afin de modéliser efficacement les états et évolutions possibles d'un objet. Ceci nous permet d'adapter le traitement de chaque objet selon son état. Nous gérons les occultations inter-objets à l'aide d'un mécanisme de suivi collectif (suivi en groupe) des

objets le temps de l'occultation et de ré-identification de ceux-ci à la fin de l'occultation. Notre algorithme s'adapte à n'importe quel type d'objet se déplaçant au sol (piétons, véhicules, *etc.*) et s'intègre naturellement au modèle de scène développé.

Nous avons également développé un ensemble de "rétro-actions" tirant parti de la connaissance des objets suivis afin d'améliorer les détections obtenues à partir d'un modèle de fond. En particulier, nous avons abordé le cas des objets stationnaires, souvent intégrés à tort dans le fond, et avons revisité la méthode de suppression des ombres du masque de mouvement en tirant parti de la connaissance des objets suivis.

L'ensemble des solutions proposées a été implémenté dans le logiciel de l'entreprise Foxstream et est compatible avec la contrainte d'exécution en temps réel nécessaire en vidéosurveillance.

Abstract

Video-surveillance cameras are increasingly used in our environment. They are indeed present almost everywhere in the cities, supermarkets, airports, warehouses, *etc.* These cameras are used, among other things, in order to detect suspect behavior (an intrusion for instance) or to recognize a specific category of object or person (gender detection, license plates detection). Other applications also exist to count and/or track people in order to analyze their behavior. Due to the increasing number of cameras and the difficulty to achieve these tasks manually, several video analysis methods have been developed in order to address them automatically.

In this thesis, we mainly focus on the detection and tracking of moving objects from a fixed camera. Unlike methods based solely on images captured by cameras, our approach integrates contextual pieces of information in order better interpret these images. Thus we propose to build a geometric and geolocalized model of the scene and the camera. This model is built directly from the pre-deployment studies of the cameras and uses the OpenStreetMap geographical database to build 3d models of buildings near the camera. We added to this model the ability to predict the position of the sun throughout the day and the resulting shadows in the scene. By predicting the shadows, and deleting them from the foreground mask, our method is able to improve the segmentation of pedestrians.

Regarding the tracking of multiple mobile objects, we use the formalism of finite state machines to effectively model the states and possible transitions that an object is allowed to take. This allows us to tailor the processing of each object according to its state. We manage the inter-object occlusion using a collective tracking strategy. When taking part in an occlusion, objects are regrouped and tracked collectively. At the end of the occlusion, each object is re-identified and individual tracking resume. Our algorithm adapts to any type of ground-moving object (pedestrians, vehicles, *etc.*) and seamlessly integrates in the developed scene model.

We have also developed several retro-actions taking advantage of the knowledge of tracked objects to improve the detections obtained with the background model. In par-

ticular, we tackle the issue of stationary objects often integrated erroneously in the background and we revisited the initial proposal regarding shadow removal.

All proposed solutions have been implemented in the Foxstream products and are able to run in real-time.

Remerciements

Je tiens tout d'abord à remercier chaleureusement Thierry Chateau (Professeur à l'Université Clermont Ferrand II) et François Brémond (Directeur de Recherche à l'INRIA Sopia-Antipolis) d'avoir accepté de rapporter et de participer à mon jury de thèse de doctorat. Je remercie tout aussi chaleureusement Jean-Philippe Domenger (Professeur à l'Université Bordeaux 1), Thierry Bouwmans (Maître de Conférences HDR à l'Université de La Rochelle) et Antoine Vacavant (Maître de Conférences à l'Université Clermont Ferrand 1) de prendre part à mon jury en tant qu'examinateurs.

J'adresse également de vifs remerciements à mes directeurs de thèse Laure Tougne (Professeure à l'Université Lyon 2) et Lionel Robinault (Responsable R&D de l'entreprise Foxstream) qui m'ont encadré tout au long de mon périple de recherche et sans qui je n'aurais jamais pu arriver jusque là. Ils ont su me donner des conseils avisés pour me permettre de réaliser cette thèse dans les meilleures conditions. Leur soutien dans les moments difficiles a été aussi précieux.

Mes remerciements vont aussi à Jean-Baptiste Ducatez (Dirigeant de l'entreprise Foxstream) qui m'a fait confiance et m'a permis d'intégrer l'équipe Foxstream. J'en profite également pour remercier l'ensemble des membres de l'équipe Foxstream pour leur accueil et leur soutien. Je remercie en particulier Ionel Pop pour les nombreuses idées et discussions échangées.

Je remercie également mes collègues du LIRIS, en particulier ceux travaillant à Lyon 2, avec qui j'ai partagé le quotidien durant ma thèse.

J'adresse aussi ma gratitude à mes relecteurs (Laure, Lionel, Françoise, Philippe, Aurélie, Charlène et Matthieu) qui ont pris le temps de lire et de corriger ce manuscrit.

Je pense aussi à remercier mes parents qui m'ont soutenu et inspiré.

Enfin, je remercie Charlène, pour son soutien indéfectible et sa patience au quotidien.

Table des matières

Table des matières **ix**

Table des figures **xiii**

Liste des tableaux **xvii**

1 Introduction **1**
- 1.1 Contexte . . . 1
- 1.2 Cadre de la thèse . . . 5
- 1.3 Contributions . . . 6
- 1.4 Organisation de la thèse . . . 7

2 État de l'art et propositions **9**
- 2.1 Détection et segmentation d'objets mobiles . . . 9
  - 2.1.1 Détection par suppression de fond . . . 10
  - 2.1.2 Utilisation du flot optique . . . 19
  - 2.1.3 Détecteurs spécifiques . . . 20
  - 2.1.4 Conclusion sur les méthodes de détection et segmentation d'objets mobiles . . . 26
- 2.2 Détection et suppression des ombres . . . 27
  - 2.2.1 Approches basées sur la couleur . . . 27
  - 2.2.2 Approches basées sur la texture . . . 31
  - 2.2.3 Approches basées sur la géométrie . . . 32
  - 2.2.4 Conclusion sur les méthodes de détection et suppression des ombres 34
- 2.3 Suivi . . . 35
  - 2.3.1 Notions . . . 35
  - 2.3.2 Approches en lot – approches séquentielles . . . 37
  - 2.3.3 Mono-objet – multi-objets . . . 40
  - 2.3.4 Gestion des occultations . . . 42

2.3.5 Conclusion sur les méthodes de suivi . . . 43
2.4 Conclusion et propositions . . . 44

3 Modèle de scène 3d et caméra : contexte issu des études de prédéploiement 45
3.1 Modèle de caméra . . . 47
3.1.1 Modèle de caméra sténopé . . . 47
3.1.2 Prise en compte de la distorsion . . . 50
3.2 Modèle de scène . . . 52
3.2.1 Modélisation . . . 52
3.2.2 OpenStreetMap . . . 54
3.3 Intégration et utilisation en vidéosurveillance . . . 59
3.3.1 Les études de prédéploiement des caméras . . . 59
3.3.2 Rendus synthétiques . . . 65
3.4 Conclusion . . . 67

4 Ombres : prédiction et suppression 69
4.1 Introduction . . . 69
4.2 Ombres projetées . . . 70
4.2.1 Ombre propre, ombre portée et pénombre . . . 70
4.2.2 Méthodes classiques pour le calcul des ombres . . . 71
4.2.3 Projection des ombres portées sur le sol . . . 73
4.3 Direction du Soleil . . . 74
4.3.1 Coordonnées locales : azimut et altitude . . . 74
4.3.2 Algorithmes de calcul . . . 75
4.4 Intégration de la prédiction des ombres au modèle de scène et de caméra 79
4.5 Validation . . . 80
4.6 Une première méthode de suppression d'ombres pour les piétons . . . 83
4.6.1 Initialisation . . . 84
4.6.2 Recherche grossière . . . 85
4.6.3 Recalage local et suppression . . . 86
4.7 Évaluation . . . 87
4.7.1 Séquences d'évaluation . . . 87
4.7.2 Métriques d'évaluation . . . 88
4.7.3 Résultats . . . 90
4.7.4 Sensibilité aux paramètres de la caméra . . . 92
4.8 Conclusion . . . 93

5 Suivi multi-objets 95
5.1 Introduction . . . 96
5.1.1 Définitions . . . 96
5.1.2 Détections et difficultés associées . . . 96
5.1.3 Aperçu de la méthode . . . 97
5.2 Modèle et état d'un objet . . . 98
5.2.1 Modèle d'objet . . . 99
5.2.2 État d'un objet . . . 100
5.3 Association objets / blobs . . . 102
5.3.1 Mesure de similarité . . . 103
5.3.2 Première passe : associations sûres . . . 106
5.3.3 Deuxième passe : associations complexes . . . 106
5.3.4 Troisième passe : associations restantes . . . 107
5.3.5 Création des nouveaux objets . . . 107
5.4 Mise à jour des objets . . . 108
5.4.1 Mise à jour du modèle d'objet . . . 108
5.4.2 Mise à jour de l'état d'un objet et automate fini . . . 108
5.4.3 L'objet est-il complètement dans la scène ? . . . 110
5.5 Évaluation . . . 115
5.5.1 Métriques d'évaluation . . . 115
5.5.2 Séquences de test . . . 117
5.5.3 Évaluation . . . 119
5.6 Conclusion . . . 123

6 Améliorations des détections par rétro-action du module de suivi 125
6.1 Rappels sur l'algorithme ViBe original . . . 125
6.2 Extensions de l'algorithme de soustraction de fond ViBe . . . 129
6.2.1 Prise en compte locale du contexte : introduction des cartes de paramètres . . . 129
6.2.2 Blocage du rognage sur les objets . . . 129
6.2.3 Réinitialisation d'un pixel . . . 130
6.2.4 L'algorithme ViBe étendu . . . 130
6.2.5 Génération des cartes de paramètres . . . 130
6.3 Suppression des ombres des objets suivis . . . 134
6.3.1 Révision de l'inférence des cuboïdes . . . 134
6.3.2 Suppression des ombres portées des objets suivis . . . 136
6.4 Évaluation . . . 138

6.4.1 Choix des paramètres 138
6.4.2 Amélioration de la segmentation 139
6.4.3 Suppression des ombres 142
6.4.4 Amélioration du suivi 143
6.4.5 Temps de calculs 144
6.5 Conclusion 146

7 Conclusion **147**
7.1 Travail réalisé 147
7.2 Perspectives 148

Bibliographie **151**

Table des figures

1.1 Aperçu d'un système de VidéoSurveillance Intelligente (VSI) 2
1.2 Principaux éléments constituant l'analyse vidéo afin d'effectuer la détection et le suivi des objets 4

2.1 Étapes d'un algorithme de suppression de fond 12
2.2 Passage d'un nuage perturbant le modèle de fond 13
2.3 Mouvement des branches provoqué par le vent. 13
2.4 Camouflage d'un piéton sortant de sa voiture. 13
2.5 Ombres segmentées avec les objets. 14
2.6 Voiture ayant quitté sa place de parking. 14
2.7 Illustration de la méthode de calcul pour l'algorithme SLDP 19
2.8 Exemple de flot optique calculé entre deux images 19
2.9 Exemple d'histogrammes de gradients orientés 22
2.10 Illustration des canaux de caractéristiques intégrales (ICF) 23
2.11 Illustration des stixels 23
2.12 Illustration du modèle déformable de véhicule 24
2.13 Illustration de la méthode de détection de Carr et al. (2012) 26
2.14 Illustration du détecteur de cuboïdes 26
2.15 Illustration de la méthode de suppression des ombres de Finlayson et al. (2002) 31
2.16 Illustration des variations d'apparence d'un même objets. 37

3.1 Exemples de logiciels permettant de réaliser les études de prédéploiement. 46
3.2 Illustrations du modèle de sténopé. 48
3.3 Exemple de distorsion observée et image corrigée 50
3.4 Illustration des motifs de distorsion les plus courants 51
3.5 Illustration du modèle de bâtiment 53
3.6 Exemple de carte générée à partir de la base de données OpenStreetMap 55

3.7 Modèle OpenStreetMap d'un bâtiment complexe avec listing XML abrégé correspondant. 56
3.8 Répartition géographique des bâtiments répertoriés dans OpenstreetMap 57
3.9 Répartition géographique des hauteurs répertoriées dans OpenstreetMap 58
3.10 Capture d'écran du simulateur de caméra. 60
3.11 Modélisation de la ville de Passau (Allemagne) à partir des données OpenStreetMap. 63
3.12 Tours jumelles Petronas (Kuala Lumpur, Malaisie) 64
3.13 Bâtiments situés aux abords de la société Foxstream. (Vaulx en Velin, France) 64
3.14 Illustration de différents rendus synthétiques générés à partir des études de prédéploiement. 65

4.1 Exemple d'ombres segmentées avec les objets les projetant menant à une fusion d'objets normalement distincts. 69
4.2 Ombre propre, ombre portée et pénombre 70
4.3 Illustration de la technique du shadow volume. 71
4.4 Illustration de la technique du shadow mapping 72
4.5 Illustration de la technique de projection plane des ombres. Les points projetés sont notés avec une étoile en exposant. 73
4.6 Système de coordonnées horizontales 75
4.7 Exemple de prédiction d'ombres (Foxstream) sur deux demi-journées . . 80
4.8 Exemple de prédiction d'ombres (barrage de Génissiat) sur une durée de 8 mois 82
4.9 Illustration de la modélisation utilisée pour la suppression de l'ombre des piétons 83
4.10 Aperçu des étapes de l'algorithme de suppression des ombres des piétons. 84
4.11 Découpage en 9 tuiles d'une boîte englobante 2d 85
4.12 Illustration de la procédure de meanshift 86
4.13 Illustration de la procédure de recalage local et de suppression des ombres 87
4.14 Exemples de résultats de suppression d'ombres pour chacune des séquences. 90
4.15 Performances de segmentation des ombres 91
4.16 Amélioration de la qualité de la segmentation après suppression des ombres 91

5.1 Erreurs courantes de détection affectant le suivi (blobs 1,3,4 et 5). 97
5.2 Aperçu de l'algorithme de suivi 98
5.3 Illustration des principales caractéristiques du modèle d'objet 99

5.4 Exemples de matrices de similarité. 103
5.5 Algorithmes d'association 104
5.6 Schématisation de l'automate fini utilisé 108
5.8 Exemple de carte sémantique et de masque de sol générés à partir du modèle de scène 110
5.7 Exemple d'évolution d'un objet de sa première à sa dernière détection. . 114
5.9 Capture d'écran du logiciel d'annotation de trajectoire utilisé. 119
5.10 Exemple de création de groupe mieux réalisée par notre méthode. . . . 121
5.11 Exemples de généralisation à d'autres types d'objets se déplaçant au sol. 122
5.12 Exemple de plan "OpenStreetMap" enrichi par les trajectoires et les cuboïdes des objets suivis. 122
5.13 Cas limite où des objets de tailles très inégales fusionnent 123

6.1 Architecture du pipeline avec rétro-action réalisant la détection et le suivi d'objets 132
6.2 Exemples de cartes de paramètres. 134
6.3 Procédure d'inférence des cuboïdes des objets suivis tenant compte de la présence ou non d'ombres portées. 135
6.4 Exemple de carte d'ombres 136
6.5 Architecture du pipeline avec rétro-action réalisant la suppression des ombres portées des objets suivis 137
6.6 Courbes de performance de segmentation obtenues pour différentes valeurs de paramètres. 139
6.7 Description de quelques éléments de la scène PETS01 140
6.8 Exemples de masques de mouvement obtenus avec et sans rétro-actions. . 141
6.9 Exemples de masques de mouvement obtenus avec et sans rétro-actions de suppression des ombres. 143
6.10 Cas défavorable où deux piétons sont suivis collectivement avec rétro-action, et individuellement sans rétro-action. 145

Liste des tableaux

2.1 Classification des principales méthodes de soustraction de fond 16

4.1 Caractéristiques des différents algorithmes de calcul de la position du Soleil 76
4.2 Évaluation pixel à pixel de la qualité de la prédiction des ombres 81
4.3 Caractéristiques des séquences de test 88
4.4 Influence des paramètres de la caméra sur les métriques de performance. 93

5.1 Caractéristiques principales des séquences d'évaluation du suivi. 118
5.2 Tableau comparatif des différentes métriques de performance. 120

6.1 Tableau récapitulatif des paramètres de ViBe 128
6.2 Résultats de segmentation obtenus sans et avec rétro-actions 140
6.3 Résultats de segmentation dans diverses configurations de suppression des ombres 142
6.4 Résultats de suivi obtenus sans et avec les rétro-actions 144
6.5 Temps de calculs moyens et écarts types des différentes étapes pour chacune des vidéos 145

Chapitre 1

Introduction

1.1 Contexte

Depuis quelques années les caméras de surveillance sont de plus en plus fréquemment présentes dans notre environnement. En effet, elles sont largement répandues dans les villes, les supermarchés, le métro, les autoroutes, les aéroports, *etc.* Ce nombre considérable de caméras engendre une masse de données très importantes qu'il s'agit alors d'analyser afin de détecter des événements "anormaux" (intrusion ou vol par exemple) et, le cas échéant, déclencher une alarme.

Traditionnellement, cette analyse était réalisée intégralement par des opérateurs humains qui avaient donc pour mission de détecter les situations suspectes et, au besoin, de déclencher une alarme. Le problème majeur de cette approche est qu'elle requiert une vigilance importante de la part des opérateurs et ce, sur une très longue durée (surtout pour des caméras fonctionnant en permanence). Ce problème est d'autant plus important qu'il a été constaté (Green et al., 1999) qu'au delà de 20 minutes à visualiser et analyser une scène, le niveau d'attention d'un opérateur humain même *"bien intentionné"* descend *"en dessous d'un niveau acceptable"* et ne permettra donc pas de garantir l'efficacité de la surveillance. On peut également imaginer qu'avec la fatigue accumulée au cours de la journée, cette durée peut encore chuter grandement.

Étant donné la disproportion entre la durée d'attention d'un opérateur et celle des séquences à analyser, renforcer les équipes de surveillance ne constitue pas une approche pertinente pour réaliser l'analyse des vidéos acquises par les caméras. De plus, cette approche ne serait pas économiquement viable.

C'est de ce constat sur les limites humaines que sont nés les systèmes de VidéoSurveillance Intelligente (VSI). Nous donnons un aperçu de tels systèmes à la figure 1.1.

Un tel système est constitué d'une partie d'acquisition (caméras analogiques, IP,

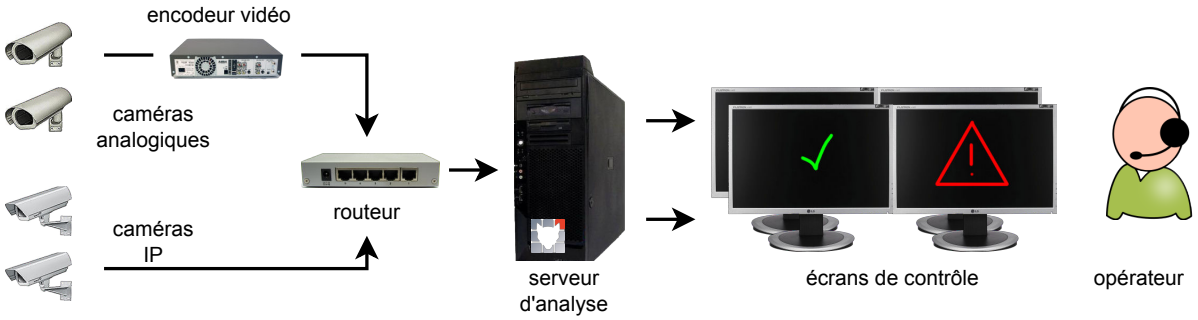


FIGURE 1.1 Aperçu d'un système de VidéoSurveillance Intelligente (VSI).
Illustration réalisée à partir d'oeuvres multiples : CCTV cameras by Tamasflex (CC by SA 3.0), LAN switch by filque (CC by SA 2.5), Monitor LCD by florisla (CC by SA 2.0), Series 2 tivo back by Jared C. Benedict (CC by SA 3.0) et autres oeuvres du domaine public.

encodeur), d'un serveur d'analyse et d'une partie de contrôle (opérateurs + écrans). Lorsqu'une alerte est signalée par le serveur d'analyse, l'opérateur réalise la levée de doute en effectuant un contrôle visuel et, transmet ou non, l'alarme.

Ces systèmes de VSI permettent d'automatiser l'analyse des vidéos afin d'en extraire les séquences suspectes et d'alerter en temps réel l'opérateur qui confirme ou infirme le déclenchement de la procédure d'alarme. Cette solution a beaucoup d'avantages par rapport à l'analyse humaine : elle fonctionne en permanence et ce, sans baisse de vigilance ; elle est plus économique et permet de traiter une bien plus grande masse de données que ne saurait le faire un opérateur humain. Cette délégation de la tâche de détection à la machine est d'un grand bénéfice pour les opérateurs qui n'ont alors plus qu'à se concentrer sur le traitement et la gestion des alarmes. Car il ne faut pas l'oublier, le rôle premier d'un télésurveilleur est le traitement et la gestion des alarmes et une caméra n'est qu'un "capteur" parmi tant d'autres (barrière infrarouge, détecteur de mouvement, par exemple). Ce "capteur" est, en revanche, intéressant car il permet de contrôler visuellement la pertinence de l'alerte déclenchée par le système de VSI, là où une barrière infrarouge, par exemple, ne permet pas cette levée de doute. De plus, l'automatisation du traitement peut être plus facilement acceptable par la population du fait du mode de consultation des vidéos par les opérateurs exclusivement lorsqu'il y a une alarme et non en continu.

En revanche, la réalisation d'un tel système "intelligent" est délicate et se heurte à la barrière d'interprétation sémantique des vidéos. Par exemple, lorsqu'un humain regarde une vidéo montrant une intrusion, il voit une personne qui s'introduit dans une zone non autorisée et décide donc de déclencher une alarme. Pour un ordinateur, cette même vidéo n'est guère plus qu'une matrice de pixels dont les valeurs évoluent au cours du temps.

Il y a un décalage important entre le niveau de raisonnement que peut avoir un humain et celui que peut avoir un ordinateur. Tout l'art de l'analyse vidéo est de développer des algorithmes permettant à l'ordinateur d'interpréter les images qu'il reçoit en vue d'effectuer une action dictée par le contexte d'application (alerter le télésurveilleur en cas d'intrusion, de trajectoire suspecte, de maraudage, ouvrir la barrière d'entrée d'un parking lorsqu'une plaque d'immatriculation est reconnue, *etc.*).

Comme nous pouvons le remarquer dans les exemples précités, les situations que le système de VSI doit être capable de détecter sont relativement complexes et soulèvent des problèmes de détection de mouvement, de trajectoire et de reconnaissance notamment. Ces problèmes fondamentaux, non encore complètement résolus, de l'analyse vidéo font actuellement l'objet de recherches très actives. Nous y reviendrons dans le détail un peu plus tard. Nous nous contenterons, pour l'instant, d'adopter un point de vue orienté "résultats" en qualifiant les erreurs commises par le système en terme d'"omissions" et de "fausses alarmes".

Les omissions sont des cas où le système n'a pas été capable de détecter une situation pour lequel il a été conçu. En vidéosurveillance, ce type d'erreur est très mal accepté car il peut être lourd de conséquences, par exemple dans le cas d'une intrusion ou d'un vol. *A contrario* les "fausses alarmes" sont des situations dans lesquelles le système a déclenché à tort une alarme. Ce type de défaut est en général mieux accepté qu'une omission, mais il ne faut pas en négliger l'impact : comme toute alarme remontée par le système doit être traitée par le télésurveilleur, ces fausses alarmes impliquent une charge de travail supplémentaire qui, si elle est répétée trop souvent, peut avoir un impact économique mesurable. Un autre danger de ces fausses alarmes à répétition est de décrédibiliser le système vis-à-vis de ses utilisateurs qui n'en tiendront alors plus compte, au risque de laisser passer une alarme bien réelle[1].

Le système de VSI idéal doit garantir simultanément l'absence de fausse alarme et l'absence d'omission. Cependant aucun système n'a été capable à ce jour d'offrir de telles performances. La difficulté vient du fait que ces indicateurs évoluent souvent dans des sens contraires : la diminution du nombre de fausses alarmes se fait au détriment du nombre d'omissions et inversement. Autrement dit, il y a un compromis à trouver entre omissions et fausses alarmes. Dans le contexte de vidéosurveillance et particulièrement de détection d'intrusion, les omissions ne sont pas acceptées et donc le compromis choisi favorise la réduction des omissions au prix de potentielles fausses alarmes.

Par ailleurs, les fausses alarmes ne sont pas toutes égales et un opérateur acceptera

1. Ceci n'est pas sans rappeler la fable d'Esope "Le garçon qui criait au loup" écrite au VII^{e} siècle avant J.C.

mieux une fausse alarme s'il peut facilement lever le doute par inspection visuelle. Par exemple, quand une alarme se déclenche à cause du passage d'un animal ou d'un papier qui vole, l'opérateur peut constater et infirmer facilement l'alarme. Certes il y a une fausse alarme qui nécessite d'être traitée mais, celle-ci reste compréhensible du point de vue de l'opérateur. À l'inverse, d'autres types de fausses alarmes sont bien moins faciles à diagnostiquer, comme par exemple celles qui sont dues aux conditions d'éclairage (ombres, reflets, éblouissements), aux ajustements internes de la caméra (changement de gain), aux conditions climatiques (neige, pluie, vent fort) ou à une combinaison de ces facteurs. Ces fausses alarmes posent problèmes car elles peuvent laisser un doute sur la provenance réelle de l'alarme. S'agit-il d'un défaut de détection ou d'une réelle intrusion masquée par ces fausses alarmes ? Si l'opérateur ne peut lever le doute, il doit déclencher une intervention pour s'assurer de la situation, ce qu'il faut éviter de faire abusivement autant que possible.

L'objectif de cette thèse est de proposer un ensemble d'améliorations sur l'analyse permettant de réduire le nombre de fausses alarmes. Plus précisément, nous nous placerons dans le cadre d'une analyse effectuant à la fois la détection et le suivi des objets d'intérêts comme illustré à la figure 1.2.

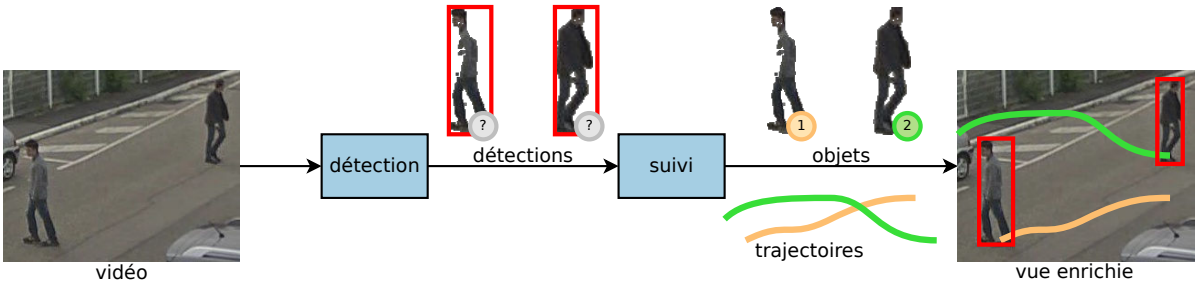


FIGURE 1.2 Principaux éléments constituant l'analyse vidéo afin d'effectuer la détection et le suivi des objets

L'analyse vidéo comporte généralement une étape de détection dont le rôle est de localiser les objets d'intérêt présents sur l'image analysée. Cette étape de détection peut éventuellement inclure une sous-étape permettant de filtrer les objets indésirables qui pourraient déclencher des fausses alarmes (ombres, petits objets ou reflets). Le résultat de cette étape de détection est souvent donné sous la forme d'un ensemble de boîtes englobantes.

Notons qu'il serait envisageable, à l'issue de cette étape de détection, de déclencher une alarme si la présence d'un objet indésirable était détectée. Cependant il est préférable, afin d'éviter les fausses alarmes, de confirmer dans le temps cette présence

indésirable via une étape de suivi. Ceci présente plusieurs avantages, d'une part, cela fiabilise l'analyse car le système accumule plus de preuves avant de déclencher une alarme. D'autre part, les trajectoires des objets construites lors de l'étape de suivi permettent aux opérateurs d'interpréter instantanément la provenance des objets présents dans la scène et donc facilite leur prise de décision. De plus, l'analyse des trajectoires peut également être mise à profit pour extraire les trajectoires types dans une scène et déduire les chemins les plus fréquentés, les points d'entrées ou de sorties, détecter les trajectoires anormales, *etc.*.

Toutefois, la construction des trajectoires pose quelques difficultés car il faut pouvoir attribuer à chaque objet un identifiant unique tout au long de la séquence. L'étape de suivi doit donc être en mesure de reconnaître d'une image sur la suivante chacun des objets qui étaient suivis précédemment pour leur attribuer le même identifiant. C'est ce que nous symbolisons à la figure 1.2 par les petits disques qui passent d'un point d'interrogation sur fond gris à un numéro d'identifiant grâce à l'étape de suivi.

1.2 Cadre de la thèse

Nous explorons dans cette thèse l'intégration de connaissances disponibles *a priori*, complémentaires aux images acquises par les caméras afin d'améliorer les performances des étapes de détection et de suivi des objets.

Ce choix d'utiliser le contexte pour améliorer les performances du système est justifié par les constats suivants :

- La qualité des images en vidéosurveillance n'est pas toujours optimale et peut affecter les résultats de l'analyse qui s'en suit. Plusieurs facteurs, peuvent en effet perturber la qualité des flux d'images acquis par les caméras. On citera par exemple : les faibles résolution et fréquence d'acquisition des images, les artéfacts dus à la compression des flux vidéos ou la propreté des objectifs qui se dégrade au cours du temps (poussières, toiles d'araignées). C'est pourquoi, nous souhaitons intégrer d'autres sources d'informations complémentaires à l'image afin d'améliorer les performances de détection et de suivi.
- Le déploiement de caméras de vidéosurveillance n'est pas fait au hasard ! Des études préliminaires au déploiement des caméras sont réalisées systématiquement et permettent notamment de spécifier les positions, orientations, et caractéristiques des caméras déployées. En outre, ces études tiennent compte des obstacles présents dans le champ de vision de la caméra tels que les bâtiments. Jusqu'à présent, les données issues de ces études de prédéploiement n'étaient pas réinvesties dans

l'analyse ce qui constitue donc une opportunité manquée de l'améliorer.

Par ailleurs, les applications en vidéosurveillance, doivent satisfaire un certain nombre de contraintes, parmi lesquelles :

- Le traitement réalisé doit être rapide afin de permettre un bonne réactivité du système (et ainsi être capable de déclencher rapidement des alarmes en cas d'intrusion par exemple).
- Le paramétrage du système doit être simple afin de permettre un déploiement rapide sur les sites surveillés (afin de réduire les coûts de paramétrage et de maintenance du système).
- Dans une logique de proposer des installations compétitives, il faudra réduire les coûts liés aux capteurs tout en maximisant la surface de la zone surveillée. Cela implique un nombre limité de caméras, celles-ci ayant des champs de vision qui ne se recoupent pas ou peu et exclut les approches de stéréovision.

1.3 Contributions

Dans cette thèse nous proposons donc :

- De réutiliser les données issues des études de prédéploiement des caméras afin de construire un modèle géométrique de scène et de caméra. Le modèle de scène que nous proposons est géolocalisé et s'appuie sur la base de données OpenStreetMap afin d'intégrer la connaissance des bâtiments environnant la scène. Ainsi, nous pouvons situer les observations de la caméra dans son contexte spatial.
- D'intégrer une modélisation géométrique des sources de lumière et des ombres portées en résultant. En particulier, nous utilisons un algorithme permettant de calculer la position du Soleil en fonction de la date, de l'heure et de la position courante afin de pouvoir prédire comment se projetteront les ombres portées tout au long de la journée.
- De généraliser l'approche de suivi proposée par Di Lascio et al. (2013) afin de pouvoir suivre n'importe quel type d'objet (pas seulement des piétons). Nous avons également amélioré la gestion des occultations inter-objets en modélisant les objets suivis à l'aide de cuboïdes et en intégrant la modélisation 3d de scène mentionnée plus haut.
- De tirer parti de la connaissance des objets suivis, afin d'améliorer l'étape de détection. En particulier, nous proposons une rétro-action du module de suivi sur le module de détection, sous la forme de cartes de paramètres, permettant d'adapter les paramètres (notamment de sensibilité) de détection selon la position des ob-

jets suivis. Nous proposons également d'autres types de rétro-actions permettant d'améliorer la persistance des objets immobiles et la gestion des voitures se garant dans un parking.

1.4 Organisation de la thèse

Nous présenterons dans le chapitre 2 une revue des méthodes existantes en matière de détection, de suivi d'objets et de suppression d'ombres. Le chapitre 3 sera l'occasion de présenter le modèle de scène et de caméra développé à partir des études de prédéploiement des caméras. Dans le chapitre 4, nous décrirons les travaux effectués afin de pouvoir détecter et supprimer les ombres du masque de mouvement. Nous aborderons au chapitre 5 l'algorithme développé afin de réaliser le suivi des objets. Dans le chapitre 6, nous présenterons différentes améliorations des détections rendues possibles grâce à la connaissance des objets suivis. Enfin, nous résumerons les différentes contributions de cette thèse au chapitre 7 et proposerons quelques perspectives qu'il serait intéressant d'explorer dans de prochains travaux.

Chapitre 2

État de l'art et propositions

L'objet de ce chapitre est de présenter les méthodes existantes relatives à la détection et au suivi d'objets pour la vidéosurveillance. En effet, avec l'essor du nombre de caméras de surveillance, l'analyse par un opérateur humain afin de détecter et de suivre un individu n'est pas envisageable, ne serait-ce que par la quantité de vidéos à analyser, mais aussi par l'attention que cela demande. C'est pourquoi, un ensemble de méthodes automatiques d'analyse vidéo a été développé ces dernières années.

Ces méthodes s'articulent généralement autour de trois grandes étapes que sont la détection des objets (2.1), des éventuels post-traitements permettant d'améliorer la qualité de ces détections, notamment en supprimant les ombres détectées de manière erronée avec les objets (2.2) et enfin une étape de suivi qui permet d'estimer les trajectoires des objets détectés au cours du temps (2.3). En conséquence, notre revue des méthodes existantes suivra également ce découpage. Enfin, à la lumière de cette revue des méthodes existantes, nous montrerons les limites de celles-ci et formulerons un ensemble de propositions afin d'améliorer la détection et le suivi des objets dans un contexte de vidéosurveillance (2.4).

2.1 Détection et segmentation d'objets mobiles

Dans cette section nous allons présenter les différentes approches permettant d'effectuer la détection et la segmentation d'objets mobiles.

Dans la littérature, on retrouve trois grandes familles d'approches permettant de détecter et segmenter les objets mobiles :

• **Détection par modèle de fond :** le principe de ces approches est de construire un modèle caractérisant les pixels appartenant à la scène dénuée d'objets d'intérêt. Une fois ce modèle (appelé modèle de "fond") construit, les pixels de l'image courante sont

comparés au modèle de fond. Les pixels présentant des différences avec le modèle de fond traduisent alors la présence d'un objet d'intérêt.

• **Détection utilisant le flot (ou flux) optique :** le flot optique caractérise le déplacement apparent de l'intensité des pixels d'une image causé par le mouvement relatif des objets de la scène par rapport à l'observateur. Ce déplacement est estimé en faisant l'hypothèse de conservation de l'intensité entre deux images proches. Les objets mobiles de la scène sont alors détectés dans les zones présentant un mouvement significatif.

• **Détection par détecteurs spécialisés :** lorsque le type d'objet à détecter est connu *a priori*, il est possible d'entraîner un classifieur spécifique. Dans ce genre d'approche, l'image est généralement parcourue à l'aide de fenêtres glissantes de différentes tailles où, pour chacune d'elle, le classifieur détecte ou non la présence d'un objet. A l'issue de cette phase, les détections multiples d'un même objet sont regroupées afin de ne présenter qu'une détection par objet.

Voyons maintenant chacune de ces familles de méthodes dans le détail.

2.1.1 Détection par suppression de fond

Notions

En début de section nous avions appelé "modèle de fond" un modèle caractérisant la "scène dénuée d'objets d'intérêt". Cette définition générale permet de comprendre intuitivement l'essence de l'approche, le "fond" désignant alors tout ce qui n'est pas "objet d'intérêt". De plus cette définition implique alors que toute déviation observée par rapport au modèle de fond indique la présence d'un "objet d'intérêt". En revanche, comme elle ne précise ni ce qu'est un "objet d'intérêt", ni la nature du "fond", cette définition est insuffisante pour comprendre les problématiques liées à ces méthodes. Nous proposons donc de préciser les notions de "fond" et "d'objets d'intérêt" (également appelés *objets d'avant-plan*, ou de *premier plan*) avant de poursuivre la présentation des méthodes de suppression de fond.

En pratique, les méthodes de suppression de fond s'appliquent au cas de la détection d'objets mobiles à partir d'une caméra fixe. Dans ce contexte, il est alors possible en première approximation de distinguer le "fond" par son caractère "fixe" en opposition aux objets d'intérêt qui sont mobiles.

Cette première approximation "fond fixe/objets mobiles" n'est cependant pas toujours vraie en pratique. En effet, certains objets d'intérêt peuvent être temporairement fixes (valise abandonnée, piéton qui s'arrête avant de traverser une route) sans devoir pour

autant être intégrés dans le fond. Et inversement, certaines parties du fond peuvent être dynamiques sans pour autant que cela ne reflète le mouvement d'un objet d'intérêt (branche d'un arbre mue par le vent, reflets à la surface de l'eau). Ainsi un modèle de fond robuste devra pouvoir gérer à la fois les fonds statiques et dynamiques, ainsi que les objets d'intérêts mobiles et temporairement stationnaires.

Nous reviendrons sur ces situations qui peuvent mettre en défaut les modèles de fond après avoir présenté les différentes étapes permettant d'effectuer la suppression de fond.

Étapes

La soustraction de fond est réalisée à partir des étapes suivantes (voir également la figure 2.1a) :

- **Initialisation du modèle de fond :** cette étape permet de construire le modèle de fond. Elle peut être réalisée hors-ligne à partir de séquences où il n'y a aucun objet dans la scène ou, à défaut, à partir des premières images de la séquence à analyser.

- **Classification objet/fond :** Cette étape compare l'image courante au modèle de fond. Cette comparaison peut s'effectuer à différentes échelles (pixel ou bloc de pixels) et porter sur différentes caractéristiques (couleur, texture). Les résultats de cette classification binaire des pixels (ou blocs de pixels) de l'image courante sont souvent donnés sous la forme d'un masque binaire (figure 2.1d) appelé *masque d'avant-plan* (*foreground mask* en anglais), masque de mouvement ou masque de détection.

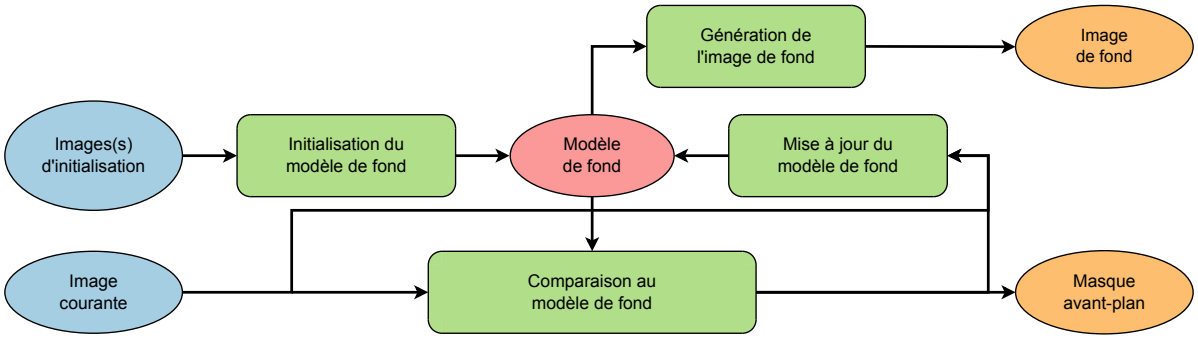
- **Mise à jour du modèle de fond :** Afin de permettre au modèle de fond de s'adapter aux changements progressifs se manifestant dans le fond, il est nécessaire de mettre à jour le modèle de fond en fonction des résultats de la classification objet/fond.

Notons également qu'il est courant de générer à partir du modèle de fond une image représentant une estimation du fond. Cette image est appelée image de fond (figure 2.1c).

Difficultés et sources d'erreurs

En pratique, plusieurs difficultés rendent l'estimation du masque d'avant-plan imparfaite. Brutzer et al. (2011) recensent par exemple les sources d'erreurs suivantes :

- **Changements d'éclairage progressifs :** ces changements modifient progressivement l'apparence de la scène et nécessitent une prise en compte explicite par le modèle de fond, typiquement en utilisant une stratégie de mise à jour progressive du modèle de fond. Ces changements se manifestent habituellement dans les scénarios en extérieur où



(a) Organisation d'un algorithme de soustraction de fond

(b) Image courante (c) Image de fond (d) Masque d'avant-plan

FIGURE 2.1 Étapes d'un algorithme de suppression de fond

les ombres projetées par les éléments fixes de la scène se déplacent lentement au cours de la journée.

• **Changement d'éclairages brutaux :** ces changements brutaux ne sont en général pas pris en compte par les modèles de fond, et causent un nombre important de faux positifs[1]. Dans un scénario d'utilisation en intérieur, cela peut être causé par l'actionnement d'un interrupteur éclairant ou éteignant subitement la pièce surveillée. Dans un scénario d'utilisation en extérieur, le passage de nuages peut obstruer temporairement les rayons du Soleil, ce qui résulte également en un changement brusque de l'éclairement. Un tel exemple est donné à la figure 2.2.

• **Fond dynamique :** certaines régions de la scène peuvent présenter du mouvement qui ne doit pas être considéré dans l'avant-plan mais au contraire qui doit être considéré comme faisant partie du fond. Ces mouvements peuvent être périodiques ou irréguliers (oscillation des branches d'arbres mues par le vent). Nous donnons un exemple de fond dynamique à la figure 2.3.

• **Camouflage :** certains objets ont une apparence très similaire avec celle du fond, rendant ainsi la séparation avant-plan/fond difficile comme montré à la figure 2.4.

1. Dans ce contexte, les faux positifs sont les pixels classifiés comme faisant partie de l'avant-plan, alors qu'ils appartiennent en réalité au fond.

FIGURE 2.2 Passage d'un nuage perturbant le modèle de fond. La première ligne montre les images couleur à différents instants. La deuxième ligne montre les masques de mouvement correspondants obtenus par la méthode de Kaewtrakulpong and Bowden (2002).

(a) Image couleur (b) Masque de mouvement

FIGURE 2.3 Mouvement des branches provoqué par le vent.

(a) Image couleur (b) Image avec détourage (c) Masque de mouvement

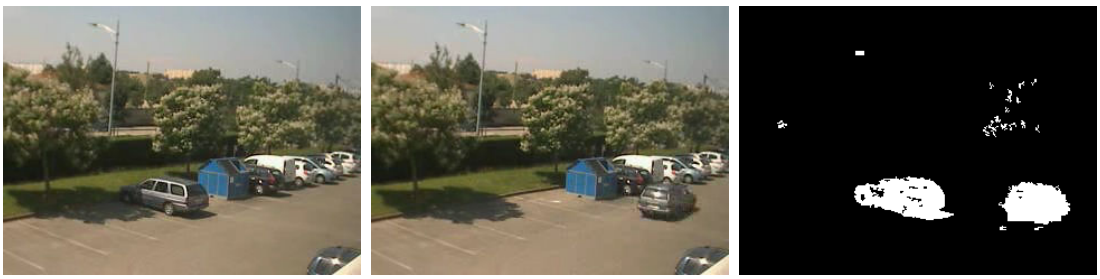
FIGURE 2.4 Camouflage d'un piéton sortant de sa voiture.

• **Ombres :** les ombres projetées par les objets d'avant-plan sont en général classifiées à tort comme faisant partie de l'avant-plan, du fait de leur dissemblance avec le fond, compliquant ainsi les étapes ultérieures d'identification et de suivi des objets. Nous donnons un exemple, à la figure 2.5, où la segmentation des ombres provoque la "fusion" des objets segmentés dans le masque de mouvement.

(a) Image couleur (b) Masque de mouvement

FIGURE 2.5 Ombres segmentées avec les objets.

• **Construction initiale du fond :** lorsque la scène ne peut pas être observée initialement vide d'objet (car cela est impossible, ou non pratique), la construction du modèle de fond est délicate, car les régions où il y a des objets empêchent l'observation du fond à cet endroit. Ce problème est couramment rencontré dans les parkings où les voitures sont initialement garées puis quittent leur place de parking. Il apparait alors un "fantôme" dans le masque de mouvement à l'endroit où était la voiture. Nous montrons un tel exemple à la figure 2.6



(a) Voiture garée (b) Voiture sortie (c) Masque de mouvement avec fantôme

FIGURE 2.6 Voiture ayant quitté sa place de parking.

• **Bruit dans le signal :** la source du flux vidéo peut ne pas être parfaite et ainsi contenir du bruit. Dans le contexte de la vidéosurveillance, ce bruit peut être dû au

capteur d'image (cas des webcams de faible qualité), ou aux artéfacts de compression du flux vidéo.

État de l'art

Un état de l'art exhaustif et récent concernant les méthodes de suppression de fonds a été réalisé par Bouwmans (2014). Dans ce manuscrit, nous nous restreindrons aux principales méthodes utilisables en temps réel.

Afin de structurer notre revue des principales méthodes de soustraction de fond, nous regroupons ces méthodes, classées par date, dans la table 2.1. Dans cette table, nous précisions la modélisation retenue, le critère de classification avant-plan / fond, la politique de mise à jour du modèle de fond, les caractéristiques utilisées, et enfin l'échelle du traitement.

Dans les modélisations les plus simples de type différence temporelle ou filtre médian (McFarlane and Schofield, 1995), seule une valeur sert de référence pour modéliser le fond. Cette modélisation unimodale trouve cependant ses limites dès lors que le fond présente une certaine dynamique et où une modélisation multimodale est préférable (cas par exemple des pixels pouvant représenter alternativement le ciel et des branches d'arbres). C'est pourquoi plusieurs approches ont été proposées, qu'elles soient paramétriques, typiquement en utilisant des mélanges de gaussiennes (Stauffer and Grimson, 1999; Kaewtrakulpong and Bowden, 2002; Zivkovic and Van der Heijden, 2006; Chen et al., 2007; Yoshinaga et al., 2013), ou non (CodeWord (Kim et al., 2005), Vumètre (Goyat et al., 2006), collection d'échantillons observés (Barnich and Van Droogenbroeck, 2011; Hofmann et al., 2012), carte auto organisée (Maddalena and Petrosino, 2008)) afin de réaliser une modélisation multimodale des valeurs de fond.

À cette modélisation du fond unimodale ou multimodale, il faut ajouter un critère permettant de classifier de manière binaire l'élément considéré afin de signifier s'il appartient ou non au fond. La plupart des méthodes utilisent alors une distance au modèle de fond, principalement sous la forme d'une distance à une gaussienne ou d'une distance signée par rapport à une valeur de référence (selon le modèle choisi) afin d'établir cette classification. On notera toutefois que certaines méthodes récentes (Barnich and Van Droogenbroeck, 2011; Hofmann et al., 2012; Yoshinaga et al., 2013) utilisent également la notion de vote afin de réaliser cette classification. D'autres incluent également des contraintes sur les résultats de classification des éléments voisins (Elgammal et al., 2000) afin de forcer une certaine cohérence spatiale des résultats.

D'autre part, la stratégie de mise à jour du modèle de fond joue un rôle essentiel afin de permettre au modèle de fond de s'adapter aux changement affectant la scène. La

TABLE 2.1 Classification des principales méthodes de soustraction de fond

Méthode (date) [**Nom**]	Modélisation	Classification	Mise à jour	Caractéristiques	Échelle
Différence temporelle **Frame Difference**	unimodale	distance à la valeur précédente	aveugle	couleur	pixel
McFarlane and Schofield (1995) **Adaptive Median**	médiane unimodale	distance à la médiane	sélective	couleur	pixel
Stauffer and Grimson (1999) **GMM1**	mélange de gaussiennes multimodale	distance à une gaussienne	sélective taux d'apprentissage	couleur	pixel
Elgammal et al. (2000) **KDE**	noyau de densité multimodale non paramétrique	seuillage contrainte sur les voisins	court terme (sélective) long terme (aveugle)	couleur	pixel
Oliver et al. (2000) **Eigenbackground**	*eigenbackground* sous espace linéaire	distance à la projection	calcul sur fenêtre temporelle	couleur	image
Kaewtrakulpong and Bowden (2002) **GMM2**	mélange de gaussiennes multimodale	distance à une gaussienne	sélective taux d'apprentissage	couleur	pixel
Kim et al. (2005) **Codebook**	*codeword* multimodale non paramétrique	distance couleur, luminance	sélective	couleur luminance	pixel
Goyat et al. (2006) **Vumètre**	échantillons multimodale non paramétrique	seuillage	sélective taux d'apprentissage	couleur	pixel
Zivkovic and Van der Heijden (2006) **GMM3**	mélange de gaussiennes multimodale	seuillage	sélective taux d'apprentissage	couleur	pixel
Chen et al. (2007) **Hierarchical GMM**	mélanges de gaussiennes multimodale	distance à une gaussienne	sélective taux d'apprentissage	histogramme de contraste	bloc pixel
Maddalena and Petrosino (2008) **Adaptive SOM**	*Self Organazing Neural Network* multimodale non paramétrique	distance perceptuelle au modèle	sélective taux d'apprentissage impact sur le voisinage	couleur	pixel
Barnich and Van Droogenbroeck (2011) **ViBe**	échantillons multimodale non paramétrique	distance aux échantillons vote	non récursive - aléatoire impact sur le voisinage	couleur	pixel
Hofmann et al. (2012) **PBAS**	échantillons multimodale non paramétrique	distance aux échantillons vote	non récursive - aléatoire impact sur le voisinage	couleur	pixel
Yoshinaga et al. (2013) **SLDP**	mélanges de gaussiennes multimodale	vote	sélective taux d'apprentissage	différences locales	pixel

plupart des méthodes utilisent ici une stratégie "sélective" de mise à jour, c'est-à-dire ne mettant à jour que les modèles des pixels ayant été classés comme faisant partie du fond. On notera deux exceptions parmi les méthodes que nous présentons, l'une (McFarlane and Schofield, 1995) qui réalise une mise à jour "aveugle" du modèle de fond, c'est-à-dire, sans considérer le résultat de la classification avant-plan/fond. L'autre, (Elgammal et al., 2000) qui combine les deux politiques de mise à jour "aveugle" et "sélective" avec des objectifs différents : la première est paramétrée afin de capturer les changements à court-terme, et la seconde afin de capturer les changements à plus long terme.

Le choix de la vitesse d'apprentissage du modèle est généralement réalisé en utilisant des taux d'apprentissage (choix du poids donné aux nouvelles valeurs pour mettre à jour les valeurs courantes), mais d'autres approches sont également possibles notamment en utilisant une mise à jour aléatoire dont on règle la fréquence moyenne (Barnich and Van Droogenbroeck, 2011; Hofmann et al., 2012).

Concernant le choix des caractéristiques utilisées pour réaliser la soustraction de fond, on retrouve majoritairement des caractéristiques de couleur RGB ou éventuellement exprimées dans un espace de couleur plus à même de séparer les composantes de chromacité et de luminance (HSV). On notera que la méthode proposée par Yoshinaga et al. (2013) utilise des différences locales afin d'être plus robuste aux changements de conditions lumineuses (ombres notamment).

Enfin, nous souhaitons également attirer l'attention sur l'échelle de traitement des modèles de fond. La plupart des méthodes raisonnent à l'échelle des pixels composant l'image, ce qui permet de donner des formes segmentées relativement précises. Cependant, cette précision est accompagnée d'un coût non négligeable dès lors que la résolution de l'image devient importante. De plus, s'il n'y a pas de mécanisme forçant une certaine cohérence spatiale des résultats, il est possible d'observer des résultats "bruités" dont il faudra "nettoyer" les irrégularités (typiquement via des opérations morphologiques). Afin de rendre un peu plus robuste la segmentation, il est possible de raisonner également à l'échelle de blocs de pixel (Chen et al., 2007) et ainsi d'exploiter les statistiques obtenues sur les pixels constituant le bloc considéré et par là-même réduire le nombre d'éléments à traiter.

Maintenant que nous avons brossé un tableau des principales options disponibles afin de réaliser la soustraction de fond, nous souhaitons décrire un peu plus précisément les méthodes suivantes (Kaewtrakulpong and Bowden, 2002; Barnich and Van Droogenbroeck, 2011; Yoshinaga et al., 2013) que nous utiliserons par la suite car elles ont démontré de bonnes performances dans les évaluations récentes de modèles de fond (Brutzer et al., 2011; Vacavant et al., 2013).

La méthode proposée par Stauffer and Grimson (1999) modélise les valeurs prises par les pixels du fond par un mélange de gaussiennes. Ce choix de distribution permet une représentation multimodale prenant en compte plusieurs couleurs pour chaque pixel (en général entre 3 et 5). Elle gère donc les cas de mouvements périodiques dans le fond (mouvement des branches d'un arbre à cause du vent par exemple). Cette méthode a ensuite été améliorée par Kaewtrakulpong and Bowden (2002) qui ont notamment intégré une procédure de détection des ombres et amélioré la vitesse d'apprentissage du modèle.

Les méthodes précédentes utilisent les valeurs de composantes RGB des pixels afin de construire le modèle de fond, ce qui les rend sensibles aux changements de luminosité. C'est pourquoi Yoshinaga et al. (2013) proposent d'utiliser à la place une caractéristique locale plus robuste aux changements de luminosité. Ils considèrent ainsi, pour un pixel p_c donné, N points p_j équirépartis sur un cercle centré en p_c de rayon r, et mesurent la différence locale : $\mathrm{LD}_j = f(p_c) - f(p_j)$ où $f(p)$ est l'intensité au pixel p. Pour chaque pixel, N mélanges de gaussiennes sont alors construits à partir des valeurs de différence locale LD_j. Le test d'appartenance est réalisé en comptant le nombre de mélanges de gaussiennes étant en accord avec le modèle : si plus de T_B mélanges correspondent, alors le pixel fait partie du fond ; sinon le pixel fait partie d'un objet. L'avantage de ce modèle est d'être à la fois robuste aux changements des conditions d'éclairage (grâce à l'utilisation de caractéristiques locales) et aux mouvements périodiques dans le fond (grâce à l'utilisation des mélanges de gaussiennes) comme illustré à la figure 2.7.

La méthode proposée par Barnich and Van Droogenbroeck (2011) conserve pour chaque pixel un ensemble fini de valeurs observées au niveau du pixel considéré ou dans son voisinage proche. Pour tester si un pixel appartient au fond, la valeur du pixel est comparée aux valeurs stockées dans le fond. S'il y a suffisamment de valeurs dans le fond compatibles avec la valeur du pixel à tester, celui-ci est marqué comme appartenant au fond, sinon le pixel est considéré comme faisant partie d'un objet du premier plan. Lorsqu'un pixel est identifié comme faisant partie du fond, sa valeur va servir à mettre à jour le modèle en remplaçant des valeurs existantes au niveau du pixel considéré, mais aussi sur un pixel voisin. Cependant, cette mise à jour est aléatoire, tant au niveau du choix de valeur à remplacer, qu'au niveau du voisin à mettre à jour, ou de la fréquence à laquelle cette mise à jour est autorisée. Cette méthode est remarquable pour plusieurs raisons : d'une part, elle ne fait aucune hypothèse sur les densités de probabilité des valeurs du fond, elle est donc universelle. D'autre part, cette méthode est rapide et donne de très bons résultats, même en présence de bruit, ou de faibles mouvements de caméras. Ce modèle de fond fait le choix délibéré de ne pas gérer explicitement les ombres

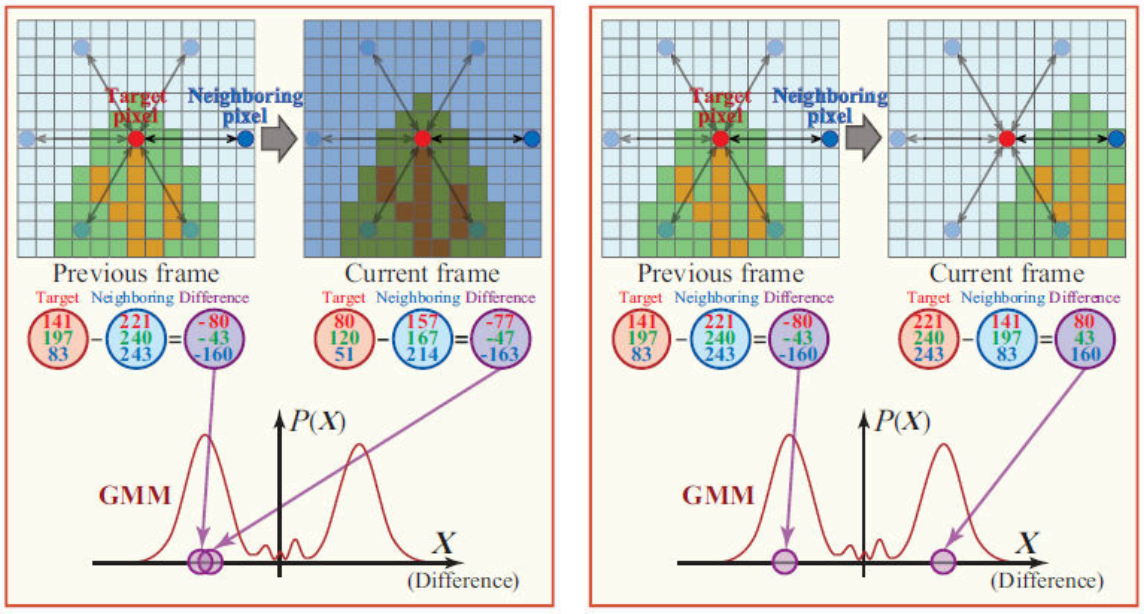
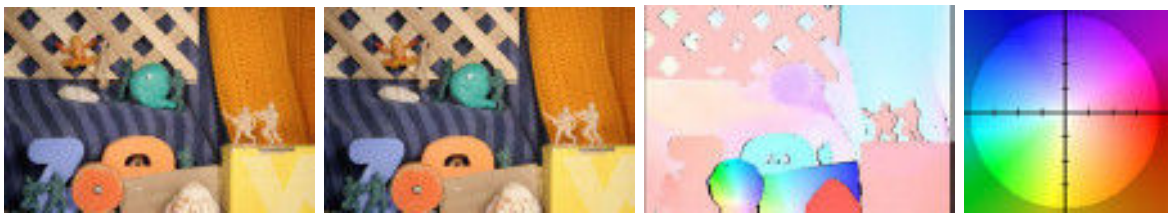


FIGURE 2.7 Illustration de la méthode de calcul pour l'algorithme SLDP. Issue de Yoshinaga et al. (2013).

et délègue cette tâche aux traitements ultérieurs qui disposent de plus d'informations.

2.1.2 Utilisation du flot optique

Un autre type d'approche pour la détection du mouvement consiste à calculer le déplacement apparent de chaque pixel entre deux images à des instants proches t et $t + \Delta t$.



(a) image 0 (b) image 1 (c) Flux optique (d) Code couleur du flot optique

FIGURE 2.8 Exemple de flot optique calculé entre deux images. Issu de Baker et al. (2011).

L'hypothèse fondamentale de ce genre d'approches est de considérer qu'entre deux images, l'intensité d'un pixel qui "bouge" ne change pas : $I(x, y, t) = I(x+u, y+v, t+\Delta t)$.

La résolution de cette équation consiste alors à calculer en chaque pixel (x, y) de l'image, le vecteur de déplacement apparent (u, v), aussi appelé *flot optique* (ou *flux optique*). Le calcul du flot optique n'est cependant pas possible tel quel car il y a deux inconnues (u, v) pour une seule équation.

Pour compléter cette équation, on peut supposer que des pixels voisins ont des déplacements similaires, et restreindre les déplacements dans un voisinage du pixel considéré. Avec l'ajout de ces contraintes, il se peut qu'il n'y ait pas de solution qui satisfasse toutes les contraintes, c'est pourquoi, une stratégie visant à minimiser les violations de contraintes peut être employée telle que l'estimation par moindres carrés (Lucas and Kanade, 1981). D'autres formulations de contraintes de régularisations et stratégies d'estimation sont également possibles (Horn and Schunck, 1981; Black and Jepson, 1996).

Les principaux problèmes liés à l'estimation du flot optique concernent les occultations qui font apparaître ou disparaître des points entre deux images consécutives, la gestion des frontières entre les objets où se manifestent des discontinuités du flot optique, les changements d'intensités liés aux changements des conditions d'éclairage et non au déplacement propre de l'objet ou de la caméra et enfin, les problèmes d'ouverture typiquement rencontrés dans les zones de couleur uniforme où le mouvement est alors ambigu (cas par exemple d'une ligne infinie verticale déplacée verticalement : l'image ne change pas alors que la ligne se déplace).

2.1.3 Détecteurs spécifiques

Dans le cas où le type d'objet à détecter est connu *a priori*, il est possible d'utiliser un détecteur spécifique. À noter que ces approches par détecteur spécifique ne font généralement pas d'hypothèses sur le mouvement des objets à détecter et s'appliquent donc également à de la segmentation sur des images, pas seulement des vidéos.

Dans le cadre de la vidéosurveillance, les exemples les types de détection les plus courants concernent les piétons (Dalal and Triggs, 2005; Wu and Nevatia, 2005; Zhu et al., 2006; Dollár et al., 2010; Benenson et al., 2013), les visages (Viola and Jones, 2004; Mathias et al., 2014), les véhicules (Leotta and Mundy, 2011) ou les formes géométriques (Carr et al., 2012; Xiao et al., 2012). C'est pourquoi nous nous focaliserons sur ces méthodes dans cette section.

Détection de piéton

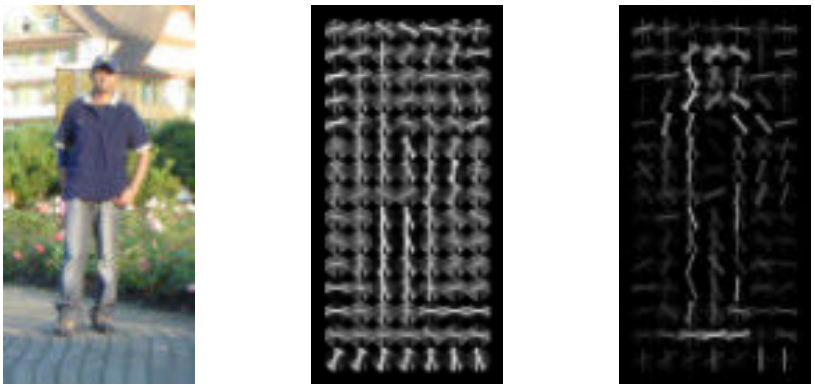
La détection de piéton est au cœur de nombreuses applications de vision par ordinateur, et justifie amplement la création de détecteurs spécifiques. Les approches récentes

en terme de détection de piétons suivent généralement le schéma à base de fenêtre glissante suivant : ce schéma est composé d'une étape d'extraction de caractéristiques, d'une étape de classification binaire, d'un parcours de l'image multi-échelles et d'une étape de suppression des détections non optimales.

Les premières approches à base de fenêtre glissante ont été proposées par Papageorgiou and Poggio (2000) et Viola and Jones (2004). Papageorgiou and Poggio (2000) réalisent la classification à l'aide d'une machine à vecteur support (SVM) et des ondelettes de Haar. Viola and Jones (2004) introduisent la notion d'image intégrale afin d'accélérer le calcul des caractéristiques similaires à celles de Haar. De plus, ils opèrent une sélection des caractéristiques à l'aide d'un algorithme d'apprentissage de type AdaBoost afin de ne conserver que les caractéristiques discriminantes. Enfin, ils structurent la classification en cascade, afin de n'effectuer les classifications les plus lourdes que sur les régions n'ayant pas été rejetées par les classifieurs précédents. Ces contributions ont permis de réaliser un détecteur d'objet générique (bien qu'appliqué ici pour les visages) et rapide.

L'introduction des Histogrammes de Gradient Orienté (HOG) par Dalal and Triggs (2005) a amélioré significativement les détecteurs de personnes. Cette caractéristique est calculée sur un ensemble de cellules carrées de petite taille (8x8 pixels), où chaque pixel de la cellule vote pour une orientation en fonction de l'orientation du gradient en ce point. Ce vote est pondéré par l'intensité du gradient en ce point. Une fois les histogrammes calculés pour chaque cellule, un regroupement en blocs et une normalisation de ceux-ci est opérée. Les blocs ne sont pas disjoints (*i.e.* une même cellule peut contribuer dans plusieurs blocs) ce qui permet d'introduire de la redondance dans le descripteur. La normalisation à l'intérieur d'un bloc permet de réduire l'influence des variations d'illumination. Le vecteur de caractéristiques est alors formé par concaténation des valeurs de chaque bloc. Afin de ne mesurer que l'apport de leur descripteur, Dalal and Triggs (2005) utilisent une simple SVM linéaire pour réaliser la classification.

Par la suite, Zhu et al. (2006) ont accéléré ce détecteur en utilisant un procédé analogue à celui proposé par Viola and Jones (2004) pour les caractéristiques de Haar. D'une part, ils calculent les HOG sur des blocs de taille variable à l'aide d'histogrammes intégraux (principe d'accélération similaire aux images intégrales) et d'autre part, ils opèrent une sélection des caractéristiques à l'aide d'un apprentissage AdaBoost et structurent la classification en cascade afin de rejeter rapidement les régions inintéressantes. Ces améliorations permettent d'obtenir un facteur d'accélération de l'ordre de 70x sans dégrader significativement les performances de détection. Une autre piste d'accélération des HOG consiste à utiliser le GPU comme l'ont proposé Prisacariu and Reid (2009).



(a) image (b) HOG (c) Orientation dominante après classification par la SVM.

FIGURE 2.9 Exemple d'histogrammes de gradients orientés. Issu de Dalal and Triggs (2005).

D'autres descripteurs de forme ont également été proposés, tels que les *edgelets* (court segment ou portion de courbe) par Wu and Nevatia (2005), ou les *shapelets* (combinaison de gradients orientés dans une sous-fenêtre de la fenêtre de détection) introduits par Sabzmeydani and Mori (2007). Dans ces deux cas, la sélection des caractéristiques optimales est apprise par boosting (*i.e.* combinaison de plusieurs classifieurs simples afin de former un classifieur fort). À noter, que Wu and Nevatia (2005) entraînent plusieurs classifieurs pour différentes parties du corps humain (tête+épaule, torse, jambes) et un détecteur pour l'intégralité de la personne, afin d'améliorer la robustesse de détection face aux occultations.

Les dernières tendances en matière de détection de piétons concernent l'amélioration de la qualité de détection en couplant des caractéristiques plus robustes (Dollár et al., 2009), et l'accélération de ces détections en intégrant du contexte (Benenson et al., 2012), ou en approximant le calcul des caractéristiques multi-échelles (Dollár et al., 2010).

Dollár et al. (2009), par exemple, proposent l'architecture suivante dans le but d'obtenir un ensemble de caractéristiques robustes (voir figure 2.10). Premièrement, plusieurs "canaux" sont extraits de l'image initiale à l'aide de transformations (linéaires ou non). Tous ces canaux sont des images construites de telle manière qu'elles sont recalées entre elles. Ensuite, plusieurs caractéristiques, appelées *Integral Channel Features*, sont extraites de ces canaux à l'aide de sommes réalisées sur des régions rectangulaires. Ces sommes sont calculées efficacement à l'aide d'images intégrales. Les *Integral Channel Features* préconisées pour la détection de piétons sont les histogrammes de gradient (6 orientations), l'intensité du gradient, et les composantes LUV au point considéré ; soit un

ensemble de 10 canaux de caractéristiques. De la même manière que les méthodes précédentes, une phase de boosting (apprentissage d'une combinaison de classifieurs simples afin de former un classifieur final robuste) est également réalisée.

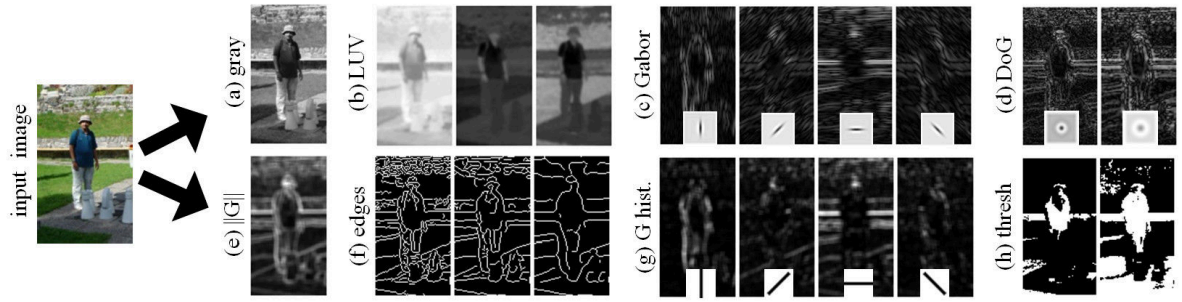


FIGURE 2.10 Illustration des canaux de caractéristiques intégrales (ICF). Issue de Dollár et al. (2009).

Une des pistes proposées par Benenson et al. (2012) pour accélérer la détection de piétons, est d'intégrer des contraintes sur les positions et tailles probables de détection (voir figure 2.11). Il utilise, pour ce faire, des images stéréos afin d'estimer des *stixels* (sorte de bâtonnets vertical fixé au sol représentant le premier obstacle rencontré), et par la suite obtenir la frontière haute et la frontière basse sur l'image des détections possibles. L'avantage des *stixels* par rapport à une plus classique carte de profondeur est que ceux-ci sont plus rapides à calculer et suffisamment expressifs pour limiter la zone de recherche des détections.

(a) Stixels (b) Limites de recherche estimées grâce aux stixels (c) Boites candidates

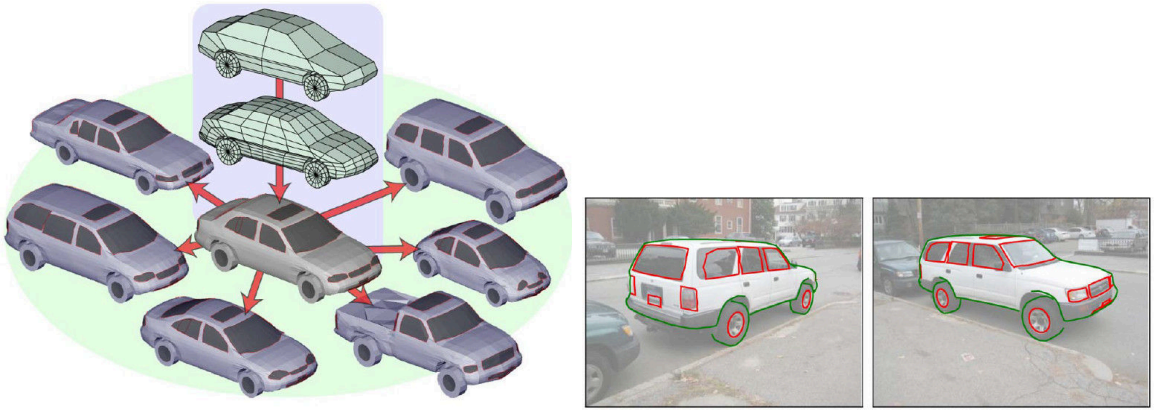
FIGURE 2.11 Illustration des stixels. Issue de Benenson et al. (2011).

Détection de véhicules

La détection de véhicules est également un domaine très actif de recherche. Les cas d'application courant concernent la surveillance du trafic routier, la gestion de parkings

ou l'assistance à la conduite. Certaines des méthodes présentées précédemment pour la détection de piétons peuvent être généralisées au cas des véhicules, à condition d'adapter l'apprentissage réalisé (*i.e.* base d'apprentissage, paramètres d'apprentissages, *etc.*). Ces méthodes ne seront donc pas représentées dans ce qui suit.

Leotta and Mundy (2011) proposent un système complet de détection et suivi de véhicules (voir figure 2.12). Ils utilisent pour cela un modèle générique, déformable de véhicule permettant de représenter la plupart des véhicules courants (berline, monospace, SUV, camionnette, *etc.*). Plus précisément, ils réalisent une estimation itérative basée sur une réduction de la distance entre les contours de l'image et ceux prédits à partir du modèle de véhicule déformable afin d'obtenir à la fois la forme, la position et l'orientation du véhicule suivi. La particularité de cette approche est d'utiliser un modèle géométrique permettant de modéliser avec finesse une large gamme de véhicules, et d'en extraire à la fois la forme, la position et l'orientation à partir d'une simple caméra calibrée.



(a) Modèle déformable de véhicule

(b) Résultats de superposition des contours réel et prédits d'une voiture.

FIGURE 2.12 Illustration du modèle déformable de véhicule. Issue de Leotta and Mundy (2011).

Détecteurs géométriques

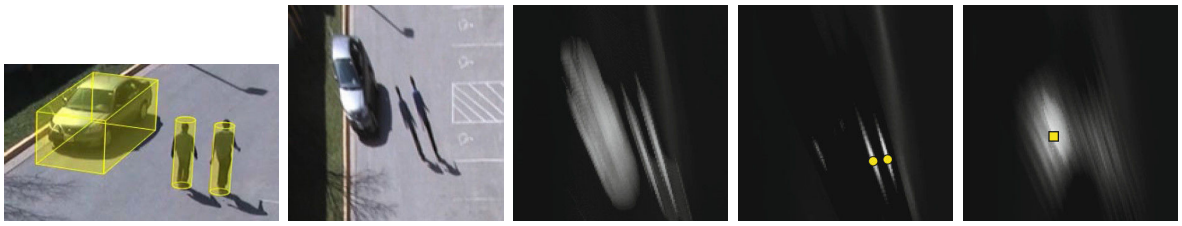
Les détecteurs spécialisés présentés précédemment caractérisent, pour la plupart, principalement l'apparence des objets à détecter, sans forcément prendre en compte les contraintes géométriques fortes s'appliquant aux objets que l'on souhaite détecter. La contrainte la plus évidente dans le contexte de vidéosurveillance est que les objets d'intérêts (piétons, véhicules) se déplacent sur le sol. Il est également possible de modéliser grossièrement la forme 3d des objets à détecter à l'aide de primitives géométriques

simples tels que des rectangles (Fleuret et al., 2008), des cylindres (Carr et al., 2012), ou des parallélépipèdes (Zuniga et al., 2006; Carr et al., 2012). Ces approches géométriques requièrent généralement l'utilisation d'une ou plusieurs caméras calibrées afin de réaliser les raisonnements nécessaires en 3d. Un des outils permettant de mener des raisonnements sur la position au sol d'un objet est la carte de d'occupation que nous présentons brièvement dans ce qui suit.

Les cartes d'occupations (*Occupancy Maps*) décrivent la probabilité qu'une partie du sol soit occupée par un objet. En pratique, ces cartes discrétisent les positions au sol de la zone à visualiser sous forme de grille rectangulaire, ce qui permet de les représenter sous forme d'image. Ces cartes sont en général calculées en projetant le masque de mouvement observé par une caméra sur la plan au sol de la zone. Lorsque plusieurs caméras sont disponibles et visualisent la même zone avec des points de vue différents, il est possible d'accumuler ces projections afin de mieux localiser les positions des objets présents. Dans le cas où une seule caméra est disponible, il est également possible d'accumuler les projections du masque de mouvement sur un ensemble de plans parallèles au sol à différentes hauteurs (entre le sol et la hauteur de l'objet à détecter). Dans le cas multi-caméras, les cartes d'occupations montrent des réponses relativement précises sur la localisation des objets, du fait de l'accumulation de preuves issues de points de vues différents, alors que dans le cas mono-caméra, les cartes d'occupations obtenues sont plus floues et rendent l'estimation de la localisation des objets plus difficile.

Carr et al. (2012) constatent que les réponses floues sur une carte d'occupation obtenue avec une seule caméra sont en fait caractéristiques de la forme de l'objet et de la position de celui-ci par rapport à la caméra. Ils proposent ainsi un modèle de formation de carte d'occupation où celle-ci est obtenue par convolution d'une carte de localisation et d'un noyau spécifique à la forme à détecter et variant avec sa position. Ils modélisent alors les piétons par des cylindres de taille fixe (0.5m de diamètre, 1.8m de hauteur) et décrivent alors le noyau correspondant. Estimer les positions des piétons revient alors à déconvoluer la carte d'occupation avec le noyau du cylindre décrit plus haut. Les auteurs montrent également, qu'il est possible de détecter des voitures avec leur système en utilisant non pas un cylindre, mais un parallélépipède de taille fixe (2m de largeur, 4m de longueur et 1.5m de hauteur) pouvant prendre 4 orientations (0°,22.5°, 45° et 67.5°). Sa méthode est illustrée à la figure 2.13.

Xiao et al. (2012), quant à eux, proposent un système capable de détecter des parallélépipèdes dans des images sans nécessiter de caméra calibrée (voir figure 2.14). Le principe de cette méthode est d'utiliser un détecteur de parties de parallélépipèdes modélisant l'apparence des sommets et des arêtes, tout en contraignant la configuration



(a) Image (b) Carte monde (c) Carte d'occupation (POM) (d) Carte des cylindres (e) Carte des cuboïdes

FIGURE 2.13 Illustration de la méthode de détection de Carr et al. (2012).

spatiale 3d de ces détections pour être en accord avec un modèle 3d de parallélépipède.

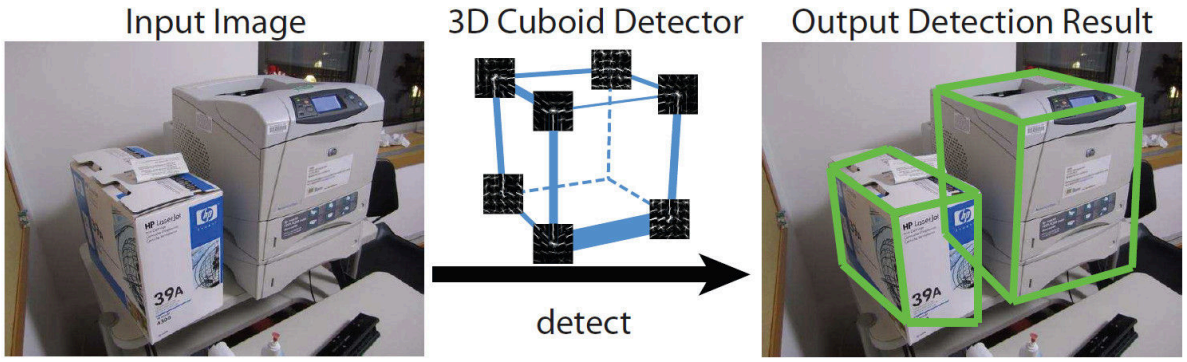


FIGURE 2.14 Illustration du détecteur de cuboïdes. Issue de Xiao et al. (2012).

2.1.4 Conclusion sur les méthodes de détection et segmentation d'objets mobiles

La détection d'objets mobiles dans le contexte de la vidéosurveillance est sujette aux contraintes de simplicité de déploiement, d'exécution en temps réel et de détection sans *a priori* important sur le type d'objets à détecter.

Ces contraintes écartent les approches à base de détecteurs spécifiques qui nécessitent un apprentissage conséquent étant donné la variabilité des poses et des apparences des objets à suivre. De plus, ces méthodes nécessitent souvent une résolution d'analyse plus importante que les méthodes de soustraction de fond afin de pouvoir détecter les objets. Cela est problématique notamment pour les détections au loin où les objets ne couvrent que quelques pixels sur l'image. Toutefois, les méthodes à base de détecteurs spécialisés permettent de distinguer chaque objet individuellement même en cas d'occultation partielle ce que ne permettent en général pas les méthodes à base de flot optique ou de

modèle de fond. De plus, ces méthodes sont plus robustes face aux ombres portées que les méthodes à base de modèle de fond ou de flot optique.

L'usage du flot optique n'est pas non plus recommandé dans notre cas à cause de la contrainte d'exécution en temps réel, des difficultés de ces méthodes à gérer les frontières séparant les différents objets et du fait que les images peuvent être bruitées.

Par ailleurs, comme nous travaillons avec des caméras fixes, l'usage des modèles de fond est possible et permet d'obtenir une segmentation rapide des objets en mouvement quelque soit leur nature. Cependant, les modèles de fond, contrairement aux détecteurs spécifiques, ne permettent pas de distinguer les objets les uns des autres lorsque ceux-ci sont fusionnés. De plus les modèles de fond sont sensibles aux ombres portées des objets mobiles. La suppression de ces ombres portées nécessite donc un traitement spécifique. Nous présentons dans la partie suivante les principales méthodes de la littérature permettant de réaliser cette suppression des ombres du masque de mouvement.

2.2 Détection et suppression des ombres

La problématique de détection et de suppression des ombres est, dans notre cas, motivée principalement par l'amélioration du suivi des objets. En effet, si l'ombre n'est pas supprimée des détections, elle peut générer de fausses détections et/ou modifier les caractéristiques (telles que la taille, la forme, le centre de gravité, ou l'apparence) de vraies détections. Or, ces caractéristiques sont celles qui sont en général utilisées pour identifier l'objet et permettre son suivi.

Nous proposons de classer les méthodes de détection et de suppression d'ombre en fonction des caractéristiques principales utilisées, à savoir les caractéristiques de couleur, les caractéristiques géométriques et les caractéristiques de texture. Notons, cependant, qu'il n'est pas rare qu'une approche combine plusieurs de ces caractéristiques, en ajoutant éventuellement une contrainte de cohérence temporelle afin d'obtenir de meilleurs résultats.

2.2.1 Approches basées sur la couleur

Intensité, constance chromatique et atténuation linéaire

Les approches utilisant principalement la couleur pour détecter les zones d'ombres, tentent plus précisément de détecter l'atténuation de luminosité résultant du blocage des rayons de lumière par l'objet occultant. Intuitivement, cela se traduit par une baisse d'intensité des pixels ombrés ; cette baisse est cependant limitée par l'éclairage ambiant.

Les valeurs admissibles d'intensité d'un pixel ombré sont donc bornées, la limite haute étant fixée en dessous de la valeur du pixel éclairé, et la limite basse étant fonction de l'éclairage ambiant de la scène.

Cependant ce critère à lui seul n'est pas utilisable en pratique car peu robuste. C'est pourquoi la plupart des méthodes basées sur la couleur (Cucchiara et al., 2003; Salvador et al., 2004; Chen et al., 2010; Lalonde et al., 2010) ajoutent une contrainte supplémentaire de constance chromatique : les pixels ombrés, bien que moins lumineux, conservent leur chromaticité (teinte et saturation). À noter que l'utilisation d'un espace de couleur adapté (HSV Cucchiara et al. (2001), C1C2C3 Salvador et al. (2004), YUV Chen et al. (2010), HSI Sun and Li (2010), LAB Lalonde et al. (2010)) peut faciliter la formulation de ces contraintes. En effet, ces espaces de couleur décorrèlent l'intensité des composantes chromatiques de la couleur.

Cucchiara et al. (2001), par exemple, formalisent cette atténuation d'intensité et cette constance chromatique dans l'espace HSV. Dans cet espace, cela signifie que la composante de teinte (H), et celle de saturation (S) ne doivent pas trop changer, alors que pour la composante de valeur (V) une atténuation est attendue.

Salvador et al. (2004) détectent les ombres en deux passes, la première utilise un critère "faible" portant sur la diminution d'intensité due à l'ombre. La deuxième passe confirme ou infirme les résultats de la première en travaillant dans l'espace C1C2C3, qui est invariant aux conditions d'éclairage, afin de détecter ou non la présence des frontières délimitant les ombres. À noter que pour réduire la sensibilité au bruit des capteurs, les auteurs travaillent sur des fenêtres centrées autour du pixel courant.

Ces méthodes exploitant la constance chromatique supposent souvent un modèle d'atténuation linéaire : les valeurs des composantes d'un pixel ombré tombent sur la ligne entre le point correspondant au pixel illuminé et le point correspondant à l'éclairage ambiant. Les modèles les plus simples négligent l'éclairage ambiant, et donc prennent comme point d'ombre minimal l'origine de l'espace de couleur (noir). D'autres modèles, plus adaptés aux applications en extérieur, modélisent la contribution du ciel et prennent un point sombre dans les tons bleus comme point d'ombre minimal. Ces derniers modèles, plus complexes, sont souvent issus d'une modélisation physique de la lumière et des surfaces éclairées.

Modélisations physiques

Nadimi and Bhanu (2004) ancrent leur méthode de détection de surfaces ombrées dans une modélisation physique des sources lumineuses et des surfaces éclairées. En particulier, ils utilisent un modèle de réflexion dichromatique prenant également en compte

l'éclairage ambiant : l'intensité lumineuse finale est la somme des contributions de la source de lumière (Soleil), de la surface éclairée et de la lumière ambiante (ciel). L'algorithme proposé fonctionne par étapes successives, chacune retirant les pixels qui ne peuvent pas représenter de l'ombre en mouvement. La première étape utilise un modèle de fond, pour ne conserver que les pixels en mouvement. La deuxième étape filtre les pixels dont l'intensité n'a pas diminué par rapport aux pixels correspondants dans l'image de fond. La troisième étape, considère le ratio pour chaque composante (R,G,B) entre l'image courante et l'image de fond. Les pixels pour lesquels ce ratio, pour la composante bleue n'est pas supérieur à ceux des composantes rouge et verte, sont supprimés. Ce test sur les ratios n'est cependant appliqué que pour les surfaces neutres (grises) et traduit le fait que pour les zones ombrées l'éclairage ambiant dû au ciel est dominant. Enfin, en appliquant un raisonnement sur les zones à albédo uniforme, l'auteur estime la couleur intrinsèque de la surface éclairée et la compare à un ensemble de couleurs de références apprises pour les surfaces de fond courantes (herbe, béton, brique, ...). Si cette couleur est similaire à une couleur de fond apprise, le pixel considéré fait partie de l'ombre.

Huang and Chen (2009) utilisent un modèle de réflexion différent (Bi-Illuminant Dichromatic Reflection Model). L'utilisation de ce modèle aboutit à une intensité lumineuse variant linéairement, pour une même surface, entre une valeur totalement ombrée (où seule la lumière ambiante contribue) et une valeur complètement éclairée où les deux composantes (lumière directe et lumière ambiante) sont présentes. Les valeurs correspondant à de l'ombre sont situées le long du segment décrit ci-dessus. Leur stratégie de détection couple un modèle global d'ombre tirant parti de cette distribution des pixels ombrés le long d'un segment et, un modèle local considérant l'écart entre les gradients sur l'image de fond et ceux sur l'image courante.

Gu and Robles-Kelly (2012) formalisent la détection des ombres, sur une image prise en extérieur, en une segmentation par contours actifs guidée par les ratios d'intensité entre les pixels ombrés et ceux éclairés (*i.e.* de part et d'autre du contour). Plus précisément, les zones ombrées sont celles qui ne reçoivent que l'éclairage provenant du ciel. Cet éclairage est décomposé en deux contributions, l'une due à la diffusion de Rayleigh, l'autre à la diffusion de Mie, selon la constitution du milieu traversé : les particules constituant l'atmosphère, de taille très petite, favorisent la diffusion de Rayleigh ; les gouttelettes constituant les nuages, de taille bien plus grande, favorisent la diffusion de Mie. Cette décomposition fait apparaître un ratio, entre pixels ombrés et éclairés d'une même surface, ne dépendant que de la longueur d'onde, des conditions atmosphériques et de la proportion de diffusion de Mie par rapport à la diffusion de Rayleigh. Ces deux

derniers paramètres sont des paramètres globaux de l'image et peuvent être estimés à partir de l'ensemble des pixels appartenant à un contour. Une fois cette estimation faite, il devient possible de connaître les ratios d'intensité attendus pour une frontière entre une zone ombrée et une zone éclairée et ainsi guider l'évolution du contour actif.

Images intrinsèques

Une autre catégorie de méthodes basées sur la modélisation physique des sources lumineuses et des surfaces éclairées, concerne les méthodes estimant des images dites intrinsèques (Barrow and Tenenbaum (1978)). L'idée est qu'il est possible de décomposer une image observée en deux composantes orthogonales : une composante lumineuse, et une composante de réflectance. À noter que, par construction, la composante de réflectance est libre de toute ombre, ce qui s'avère utile pour détecter et/ou supprimer celle-ci.

Finlayson et al. (2002) constatent que, pour une caméra donnée, observant une mire de calibration de couleur (figure 2.15a) dans différentes conditions d'éclairage, la représentation des valeurs RGB des pixels de la mire dans le plan $(\log(G/R), \log(B/R))$ donne un ensemble de droites parallèles entre elles (figure 2.15b). La direction de ces droites parallèles est constante pour une caméra donnée et les variations de conditions d'éclairage se traduisent par des déplacements le long de ces droites. Ainsi, si l'on dispose de la direction caractéristique de la caméra, projeter les points du plan $(\log(G/R), \log(B/R))$ sur une droite perpendiculaire à cette direction permet de s'affranchir des conditions d'éclairage. Les points projetés forment une représentation 1D (niveau de gris) indépendante des conditions d'éclairage (figure 2.15d). La justification physique de cette approche découle d'hypothèses faites sur la source lumineuse (loi de Planck), la réponse du capteur de la caméra (capteur à bande étroite) et un modèle Lambertien de formation des images. Pour détecter les zones d'ombre de l'image, l'auteur compare les contours de l'image originale à ceux de l'image en niveau de gris indépendante des conditions d'éclairage. Dans ses travaux ultérieurs, Finlayson et al. (2009) décrivent une procédure permettant de trouver automatiquement la direction de projection dans le plan $(\log(G/R), \log(B/R))$ qui rend l'image indépendante de l'illumination. Cette direction est celle pour laquelle l'entropie des points projetés est minimale.

Une autre approche permettant de réaliser la décomposition d'une image en sa composante lumineuse et sa composante de réflectance a été proposée par Weiss (2001). L'auteur, propose d'estimer l'illumination d'une séquence d'images à réflectance constante, par exemple dans le cas d'une scène filmée par une caméra fixe tout au long de la journée. L'hypothèse que fait l'auteur est de considérer que, pour les images naturelles, l'applica-

(a) Mire (b) Projection optimale (c) Image couleur (d) Image Invariante 1D

FIGURE 2.15 Illustration de la méthode de suppression des ombres de Finlayson et al. (2002). Issue de Finlayson et al. (2002)

tion d'un filtre dérivatif à la composante d'illumination renvoie une réponse modélisable par une distribution Laplacienne uniforme dans le temps et l'espace. Cette hypothèse permet de construire un estimateur optimal (*i.e.* maximisant la vraisemblance de la composante de réflectance connaissant la réponse d'un filtre) pour la composante de réflectance. Cet estimateur est la médiane temporelle de la réponse d'un filtre dérivatif. Ainsi, l'auteur propose d'obtenir la composante de réflectance, supposée constante sur la séquence, et peut ainsi retrouver la composante d'illumination propre à chaque image de la séquence.

2.2.2 Approches basées sur la texture

Les méthodes précédentes utilisant la couleur considèrent généralement chaque pixel de manière isolée. Cela présente l'avantage de permettre des traitements rapides et simples. Cependant, négliger le voisinage du pixel considéré peut nuire à la robustesse de la détection, notamment vis-à-vis des conditions d'éclairage. L'utilisation de caractéristiques de texture vise à pallier ce problème. Le principe de base est de considérer que les ombres ne modifient pas la texture des surfaces sur lesquelles elles sont projetées. Plusieurs stratégies de corrélation de la texture de l'image courante et de l'image de fond ont été proposées parmi lesquelles on trouve : le calcul de corrélation croisée normalisée (Tian et al., 2005), la corrélation de gradients ou de contours (Xu et al., 2005; Panicker and Wilscy, 2010; Sanin et al., 2010), l'utilisation de transformations orthogonales (Zhang et al., 2006), la convolution sur filtres de Gabor (Leone and Distante, 2007), ou l'utilisation de motifs locaux (Qin et al., 2010) (Scale Invariant Local Ternary Pattern). À noter cependant que le calcul de ces caractéristiques de textures est souvent coûteux et n'est donc effectué qu'après un premier filtrage rapide basé sur l'intensité ou la couleur.

Tian et al. (2005) travaillent sur des images en niveaux de gris et caractérisent la présence d'ombre en calculant la corrélation croisée normalisée sur un voisinage du pixel considéré sur l'image courante et l'image du fond. Une forte corrélation indique une zone d'ombre.

Zhang et al. (2006) proposent d'utiliser cinq types de décompositions orthogonales (DCT, DFT, transformée de Haar, décomposition en valeurs singulières, transformée de Hadamard) afin de caractériser les ombres. Pour chaque pixel, ils considèrent un bloc de pixels voisins afin de calculer les transformées précitées et normalisent les coefficients obtenus afin d'obtenir une représentation indépendante de l'éclairage. Si les coefficients obtenus sur les pixels de l'image courante et celle du fond sont proches, le pixel considéré est de l'ombre.

Leone and Distante (2007) quant à eux, caractérisent les pixels ombrés en décomposant le voisinage de chaque pixel candidat sur un banc de filtres de Gabor, appris hors ligne. Les caractéristiques obtenues pour le pixel candidat et pour le pixel correspondant de l'image de fond sont comparées : si elles sont similaires, le pixel considéré fait partie d'une zone d'ombre. Ce critère de texture est évidemment couplé à un critère portant sur la diminution d'intensité afin de réduire le nombre de pixels d'ombre candidats.

Sanin et al. (2010) caractérisent les zones d'ombre en comparant la direction des gradients dans l'image courante à ceux correspondant dans l'image de fond. Plus précisément, partant du masque de mouvement, les auteurs opèrent une première sélection des pixels d'ombre en utilisant les mêmes critères colorimétriques que Cucchiara et al. (2001). Les seuils choisis lors de cette étape sont délibérément peu sélectifs afin d'obtenir un fort taux de détection d'ombre et de grandes composantes connexes où les caractéristiques de texture seront significatives. Ensuite, une étape visant à éclater les composantes connexes qui contiendraient à la fois de l'ombre et des objets est réalisée en découpant les composantes connexes le long des contours présents dans l'image courante, mais pas dans l'image de fond. À l'issue de cette étape, idéalement, les composantes connexes obtenues contiennent soit de l'ombre, soit une partie d'objet, mais pas les deux ensemble. Enfin, on calcule pour chaque composante connexe la fraction de gradients de magnitude suffisante dont l'orientation est proche du gradient correspondant dans l'image de fond. Les composantes pour lesquelles cette fraction est élevée sont considérées comme étant de l'ombre, et sont donc retirées du masque de mouvement.

2.2.3 Approches basées sur la géométrie

Les caractéristiques de couleur et de textures présentées dans les parties précédentes ne considèrent pas les contraintes géométriques liant une ombre projetée et l'objet qui la

projette. Par exemple, pour un objet se déplaçant sur le sol (piéton qui marche, voiture), l'ombre projetée est attenante à l'objet et peut être modélisée par une projection de la silhouette de l'objet sur le sol selon la direction de la source lumineuse. L'avantage de ces méthodes est d'être ancré dans une réalité physique relativement indépendante des conditions colorimétriques d'acquisition de l'image : seules les positions et formes des objets et sources de lumières importent, pas les couleurs des pixels de l'image. Cette propriété est intéressante dans les cas où les conditions d'éclairages sont changeantes ou lorsque la qualité des capteurs des caméras est faible. Ces méthodes sont cependant souvent spécifiques à des types d'objets précis (piétons (Hsieh et al., 2003), véhicules (Yoneyama et al., 2003; Leotta and Mundy, 2011)) et supposent le plus souvent une source lumineuse unique ou requièrent une surface de projection plane.

Hsieh et al. (2003) proposent une méthode de suppression des ombres projetées par des piétons capable de gérer le cas où plusieurs piétons sont regroupés dans une même composante connexe. Pour ce faire, ils analysent les projections verticales des composantes connexes afin de détecter les positions des têtes, et les bandes verticales délimitant le corps de chacun de ces piétons. Une fois les piétons séparés, ils estiment pour chaque piéton une ligne de séparation grossière entre un piéton et son ombre. Idéalement cette ligne sépare la composante connexe en une région ne contenant que de l'ombre, et une région contenant tout le piéton et éventuellement de l'ombre. Cette première séparation piéton / ombre est ensuite affinée en construisant un modèle gaussien de l'ombre à partir des pixels de la région identifiée comme ombre. Ce modèle tient compte à la fois de l'intensité du pixel mais aussi de sa position dans l'ellipse englobante de sa région. Une fois ce modèle construit, un test d'appartenance à ce modèle est appliqué aux pixels de l'autre région (identifiée comme piéton) afin de supprimer les pixels d'ombre qui seraient présents dans cette région.

Plus récemment Chen et al. (2010) ont revisité la suppression d'ombre portée par des piétons. Outre les caractérisations de couleur (HSI) et de texture (HOG), les auteurs intègrent une caractérisation de position via l'utilisation d'un système log-polaire de coordonnées. L'origine de ce repère est fixé au niveau de la tête du piéton et s'étend sur un demi-disque de telle manière que le piéton debout occupe l'axe de symétrie du demi-disque. Ce repère permet d'augmenter la résolution autour de l'origine par rapport à la résolution de la périphérie et ainsi, permet de mieux utiliser l'information de position à des fins de classification. Une première détection grossière de l'ombre est alors réalisée par une SVM, entrainée préalablement, utilisant les trois descripteurs de couleur, de texture et de position précités. À partir de cette première estimation, une frontière linéaire de séparation entre piéton et ombre est réalisée. Enfin les pixels de la région d'ombre sont

remplacés par une estimation du fond de la scène, ce qui conduit à une image d'où les ombres ont été retirées.

La méthode proposée par Nicolas and Pinel (2006) permet de segmenter les objets et leur ombre, tout en estimant la position de la source lumineuse éclairant la scène. Cette méthode s'appuie notamment sur des hypothèses de projection des ombres sur une surface plane, et sur l'unicité de la source lumineuse éclairant la scène. Outre ces hypothèses destinées à un raisonnement géométrique, l'auteur ajoute également des critères sur la couleur et sur la cohérence temporelle afin d'améliorer la robustesse de sa méthode. L'originalité de la méthode est qu'elle ne fait pas d'hypothèse sur la forme des objets – tous les raisonnements sont menés en 2d dans le plan de l'image – et permet d'estimer la position de la source lumineuse (éventuellement non ponctuelle).

Leotta and Mundy (2011) proposent un système complet de classification et de suivi de véhicules utilisant un modèle 3d prenant en compte les ombres projetées par ceux-ci. Le système réalise de manière itérative une estimation contrainte de la forme 3d du véhicule suivi. Cette estimation intègre, si la localisation, la date et l'heure sont connues, une prédiction de la projection de l'ombre du véhicule suivi. Cette prédiction de l'ombre du véhicule s'obtient en projetant le modèle 3d du véhicule sur le sol selon la direction du Soleil. Au contraire de la méthode précédente, cette méthode raisonne en 3d et modélise la forme des objets ayant une ombre.

2.2.4 Conclusion sur les méthodes de détection et suppression des ombres

Nous avons organisé notre revue des méthodes de détection et de suppression des ombres du masque de mouvement en trois catégories principales : les méthodes basées sur la couleur, les méthodes basées sur la texture et les méthodes géométriques.

Les méthodes colorimétriques sont comparativement les plus simples à implémenter et sont les plus rapides car elles travaillent à l'échelle du pixel et négligent les contraintes spatiales pouvant exister entre ceux-ci. Les principaux défauts de ces approches concernent l'impossibilité de résoudre les problèmes de camouflage où objets et ombres ont des couleurs similaires. Ce problème est exacerbé par la compression forte que subissent les flux vidéo (MJPEG) afin d'être transmis de la caméra à l'unité de traitement. Par ailleurs, lorsque ces méthodes travaillent à l'échelle du pixel, elles sont plus sensibles au bruit d'acquisition. Enfin les approches colorimétriques nécessitent de fixer un certain nombre de seuils (atténuation d'intensité ou distance maximale entre teintes similaires, par exemple) ce qui peut être délicat en pratique.

Les méthodes texturales intègrent le voisinage de chacun des pixels afin d'extraire des caractéristiques robustes. Cela résout notamment les problèmes de sensibilité au bruit d'acquisition. En revanche, il peut être délicat de choisir la taille du voisinage à considérer pour calculer les caractéristiques de texture : il faut prendre en effet un voisinage suffisamment grand pour extraire des motifs pertinents et éviter les zones uniformes, mais pas trop, afin d'éviter les régions qui contiendraient à la fois des ombres et des objets d'intérêt. Par ailleurs ces méthodes nécessitent plus de ressources de calcul que les méthodes travaillant à l'échelle du pixel. Ce point est souvent résolu en utilisant une première passe sélectionnant les zones d'ombres candidates à l'aide d'un critère colorimétrique simple.

Les méthodes géométriques, quant à elles, exploitent les contraintes spatiales liant les objets opaques, les sources de lumières et les ombres. Ces méthodes ont l'avantage d'être complémentaires (voire orthogonales) à l'image acquise par les caméras et sont donc moins sensibles aux dégradations que peuvent subir les images acquises par les caméras. En revanche, elles requièrent par définition la connaissance de la géométrie des objets opaques de la scène ainsi que des caractéristiques de la (ou des) source(s) de lumière.

Dans le cadre de cette thèse, explorant particulièrement l'intégration du contexte pour la détection et le suivi des objets, les approches géométriques nous semblent particulièrement intéressantes. En effet, ces approches doivent modéliser géométriquement les objets opaques ainsi que les sources de lumière composant la scène et peuvent donc directement bénéficier de l'intégration du contexte spatial d'observation (positions et orientations des cameras et des sources de lumières) obtenu lors des études de prédéploiement des caméras. De plus, la faible sensibilité intrinsèque de ces méthodes aux dégradations de la qualité des images acquises ainsi que le coût d'exécution modéré sont particulièrement intéressants dans notre cas.

2.3 Suivi

2.3.1 Notions

Dans le cadre de cette thèse, nous entendons par suivi d'objets le fait d'estimer à chaque instant t les états $X^i(t)$ de chacun des objets i de la scène à partir de mesures effectuées sur les images composant la séquence vidéo. On appelle alors trajectoire de l'objet i l'ensemble des états $\{X^i(t)\}$ pour t variant du début à la fin de la séquence vidéo. Cette définition de la trajectoire est un peu plus générale que la définition "courante" qui réduit en général l'état d'un objet à sa seule position.

Suite à cette définition, nous pourrions penser qu'il suffit simplement de concaténer l'ensemble des détections obtenues sur chacune des images de la séquence pour réaliser le suivi des objets. Cette approche n'est cependant pas satisfaisante car elle ne résout pas le problème d'association entre un objet i et les détections qui lui correspondent, ce qui est le centre du problème du suivi. Pour arriver à résoudre ce problème d'association, les méthodes de suivi intègrent la connaissance des états passés des objets suivis et les mesures disponibles afin d'estimer le (ou les) état(s) futur(s) des objets.

Éléments d'un algorithme de suivi

Afin de construire une telle méthode de suivi, il convient de déterminer les éléments suivants :

- **Un modèle d'état** qui constitue la représentation des objets suivis. Celui-ci décrit l'objet par un point, un nuage de points, une boîte 2d, une ellipse, un contour, un squelette ou une silhouette selon l'application visée. On adjoindra également très souvent une composante cinématique permettant d'introduire la vitesse et éventuellement l'accélération de l'objet.
- **Un modèle d'observation** qui réalise les mesures images concernant les objets suivis.
- **Un modèle dynamique** qui caractérise l'évolution des objets.
- **Une stratégie d'estimation** qui constitue le cœur du suivi et permet ainsi, à partir des observations, des états passés et du modèle dynamique, d'inférer le (ou les) état(s) futur(s).

Difficultés

Les principales difficultés auxquelles un algorithme de suivi est confronté concernent :

- **Les changements d'apparence** qui peuvent se manifester notamment lorsque l'objet n'est plus vu sous le même angle (piéton qui se tourne) ou se déforme (piéton qui marche) comme illustré aux figures (2.16a – 2.16c).
- **Les occultations** qui masquent tout ou une partie de l'objet sur une durée plus ou moins longue comme illustré aux figures (2.16d – 2.16e).
- **Les changements de taille** que la projection d'un objet sur l'image subit à cause de la perspective (les objets au loin apparaissent plus petits que les objets proches de la caméra).

• **Les ambiguïtés résultant** du suivi d'objets similaires (par exemple les joueurs d'une même équipe de foot ont tous le même maillot).

• **La qualité des détections/observations** qui sont imparfaites (typiquement omissions ou fausses détections).

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

FIGURE 2.16 Illustration des variations d'apparence d'un même objets.

Bien que de formulation simple, ce problème d'estimation de trajectoire est loin d'être résolu comme en témoignent les nombreuses publications récentes à ce sujet [2].

Nous proposons de présenter les principales méthodes de suivi au travers de trois points différenciateurs : premièrement, leur rapport au temps ainsi que les stratégies d'inférence associées (2.3.2) ; deuxièmement, leur capacité à suivre un ou plusieurs objets (2.3.3) et finalement leur gestion des occultations (2.3.4).

2.3.2 Approches en lot – approches séquentielles

Nous distinguons ici, les méthodes séquentielles (ou *online*), qui ne se basent que sur les données passées afin d'estimer les états futurs par opposition aux méthodes en lot (ou *batch, offline*) qui ont à leur disposition toutes les observations même "futures" afin d'estimer les trajectoires des objets.

2. plus de 6000 publications référencées sur google scholar pour l'année 2014 avec les mots clés "object tracking"

Paradigmes globaux d'inférence des trajectoires

Pour les méthodes en lot, il est possible de formaliser cette estimation de trajectoire en un problème d'optimisation globale où l'énergie à optimiser permet de relier les différentes détections afin de former les trajectoires des objets sur la séquence considérée.

Une telle approche n'est en général pas applicable directement en l'état sur l'ensemble de la séquence dès que celle-ci est un peu longue et qu'il y a trop d'objets. Aussi, plusieurs stratégies sont possibles pour réduire la complexité de cette optimisation.

Le premier moyen de réduire la complexité de cette optimisation est de ne pas travailler sur l'ensemble de la séquence, mais sur une fenêtre temporelle (Berclaz et al., 2006). Cela introduit toutefois un délai de réponse au moins égal à la durée de la fenêtre temporelle considérée. Dans le même esprit, il est possible de constituer dans un premier temps des "bouts" de trajectoires fiables, aussi appelés *tracklets* (Kaucic et al., 2005; Li et al., 2009; Henriques et al., 2011; Yang and Nevatia, 2012), de manière itérative. Puis, dans un second temps de relier ces *tracklets* grâce à une démarche d'optimisation globale. Ceci permet de réduire considérablement les ressources nécessaires à l'extraction des trajectoires. Afin de faciliter la résolution des associations de *tracklets* difficiles, Henschel et al. (2014) ont récemment proposé d'utiliser des *tree tracklets*. Ceux-ci permettent d'encoder les bifurcations émanant des cas difficiles et d'en repousser la résolution une fois que des informations supplémentaires seront disponibles.

Un autre moyen de simplifier l'optimisation globale est de discrétiser l'ensemble des positions sous forme de grille, et de créer un graphe dont les nœuds représentent les points discrets de la grille à chaque instant et dont les arêtes modélisent les transitions possibles entre les différents nœuds (voisinage spatial et temporel). Avec cette formulation, il est alors possible de résoudre le problème de calcul de trajectoire en utilisant un algorithme de plus court chemin dans ce graphe (Andriyenko and Schindler, 2010; Berclaz et al., 2011). Toutefois, le fait d'utiliser une grille discrète induit une erreur sur l'estimation des positions et une trajectoire "crénelée".

Par ailleurs, cette possibilité de connaître les observations futures permet une plus grande robustesse dans l'estimation des trajectoires, notamment dans la résolution des occultations, par rapport à ce qui est possible avec les méthodes séquentielles. Cette stratégie est particulièrement adaptée au cas de la relecture *a posteriori* des vidéos car dans ce cas toute la vidéo est disponible et les contraintes portant sur le temps d'exécution sont moindres. En revanche, ces méthodes ne sont pas applicables en l'état au suivi en temps réel car elles nécessitent l'ensemble des données même futures ou induisent un délai de réponse et sont généralement coûteuses en temps de calcul.

Paradigmes récursifs d'inférence des trajectoires

Au contraire, les approches séquentielles, n'exploitent que les observations passées afin de réaliser l'estimation de l'état courant des objets. Les trajectoires sont alors construites de manière itérative au fur et à mesure que les nouvelles images sont acquises. Nous présentons dans ce qui suit les stratégies classiques d'estimations récursives des trajectoires telles que le filtre de Kalman et les filtres à particules.

Le filtre de Kalman (Kalman, 1960) est une des approches parmi les plus populaires en matière de suivi. Ce filtre est en effet un estimateur récursif qui permet d'estimer l'état d'un système à partir de mesures bruitées. C'est même l'estimateur linéaire optimal (au sens des moindres carrés) lorsque le système et ses observations sont régis par des équations linéaires et que les éventuels bruits associés sont additifs, blancs et gaussiens. En outre, ces hypothèses permettent d'obtenir une formulation analytique de ce filtre facile à implémenter. Les principales utilisations du filtre de Kalman modélisent le déplacement des objets en faisant l'hypothèse de déplacement soit à vitesse constante, soit à accélération constante. Pour les objets non ponctuels modélisés par des boîtes englobantes 2d, il est possible également d'intégrer leurs dimensions et leurs variations dans la modélisation de l'état afin de permettre une estimation lissée de celles-ci. Dans certains cas, cependant, l'hypothèse de linéarité du système (transition et observation) n'est pas réalisable. Il a donc été proposé, sous le nom de filtre de Kalman étendu (ou *Extended Kalman Filter*), d'appliquer le filtre de Kalman sur les équations linéarisées du système. À noter que cette généralisation aux systèmes non linéaires fait perdre le caractère optimal du filtre, et peut perturber la convergence des estimations. Une autre extension du filtre de Kalman est le filtre de Kalman "*unscented*" (Julier and Uhlmann, 1997) qui corrige les problèmes de convergence que peut avoir le filtre de Kalman étendu en utilisant une approche d'échantillonnage déterministe pour estimer les variables d'états et leurs covariances.

Le principal défaut des méthodes dérivant du filtre de Kalman est le fait que ces approches sont unimodales : seule une valeur d'état est conservée et donc la récupération en cas d'erreur est plus délicate que si une modélisation multimodale avait été employée. Les filtres à particules permettent une telle modélisation multimodale. Ils ont été popularisés par Gordon et al. (1993) dans la communauté radar et par Isard and Blake (1998) dans la communauté vision avec l'algorithme nommé *condensation*. Depuis, cette approche est régulièrement revisitée (Vermaak et al., 2003; Okuma et al., 2004; Bardet et al., 2009; Breitenstein et al., 2011; Zuriarrain et al., 2013).

Outre la possibilité d'utiliser une modélisation multimodale, plus robuste aux dérives, cette approche permet également de lever les contraintes de linéarité du système et de

bruit blanc gaussien faites par le filtre de Kalman. La seule réelle contrainte sur le système est qu'il doit être facilement *simulable*. En effet, les approches à base de filtres à particules formalisent le suivi en une estimation récursive de la densité de probabilités de l'état du système en supposant connues les observations et les états passés et courants. En toute rigueur, il est même fait l'hypothèse de système markovien, où l'état futur ne dépend que de l'état courant et des observations courantes. La stratégie adoptée par les filtres à particules afin d'estimer cette densité de probabilité est de simuler le système un certain nombre de fois, chaque résultat de simulation étant représenté par une particule. Un mécanisme de sélection de ces particules est également appliqué afin de donner plus de poids aux états en accord avec les observations. Ce mécanisme de sélection est spécifié par une fonction de vraisemblance et permet de réaffecter, pour l'itération suivante, les particules de poids les plus faibles vers les états les plus probables. Ainsi, les états les plus probables se voient affecter plus de particules (et donc plus de ressources de calcul), et les états les moins probables sont oubliés.

Le filtre à particules jouit d'une grande flexibilité vis-à-vis des systèmes qu'il peut modéliser et se montre robuste grâce à la modélisation multimodale probabiliste adoptée. Cependant, cette approche requiert une puissance de calcul bien plus importante que les filtres de Kalman du fait de la nécessité de simuler un grand nombre de fois le système et d'évaluer puis réaffecter chacune des particules en fonction des observations. Des applications de tels filtres au cas temps réel existent toutefois (Montemayor et al., 2004; Gelencsér-Horváth et al., 2013) et exploitent généralement les capacités du processeur graphique afin de paralléliser les simulations.

2.3.3 Mono-objet – multi-objets

Une autre distinction significative parmi les méthodes de suivi concerne leur capacité à suivre un seul objet, ou plusieurs objets simultanément. Cette distinction peut paraître relativement minime de prime abord, mais impacte profondément la manière d'aborder le suivi.

Les méthodes dites de *Visual Tracking* réalisent un tel suivi mono-objet. Ces méthodes sont le plus souvent génériques dans le sens où elles s'appliquent à n'importe quel type d'objet, mais supposent connue la position initiale de l'objet à suivre (marquée par l'utilisateur sur la première image). Nous ne nous attarderons pas sur ces méthodes mono-objets qui sont hors du cadre de cette thèse, mais nous invitons le lecteur curieux à aller consulter l'étude récente proposée par Smeulders et al. (2014).

Les méthodes de suivi multi-objets, quant à elles, ne peuvent faire d'hypothèses sur le nombre d'objets à suivre (et qui peut être variable au cours de la séquence vidéo) :

elles doivent donc gérer les entrées / sorties d'objets. Par ailleurs, se pose la question de la manière d'aborder le suivi multi-objet : soit en considérant chaque objet à suivre de manière séparée, (un "traqueur" par objet), soit en considérant de manière jointe l'ensemble des objets à suivre (un seul "traqueur" pour l'ensemble des objets). Dans le cas de suivi joint, le nombre de combinaisons d'états possibles du système augmente exponentiellement avec le nombre d'objets et permet de modéliser les interactions entre les objets suivis (Khan et al., 2004; Zhao and Nevatia, 2004; Bardet et al., 2009). Cependant cette plus grande finesse de modélisation induit une explosion combinatoire des cas à considérer lors de l'inférence des nouveaux états des objets. Pour ces algorithmes, une stratégie efficace d'exploration de l'espace des configurations est donc nécessaire.

Au contraire, lorsque chaque objet est suivi individuellement, l'espace des configurations de celui-ci est de taille fixe, mais ne permet pas de modéliser simplement les interactions entre les différents objets suivis.

De plus, dans le cas où chaque objet est suivi par un "traqueur", le problème se pose de savoir comment réaliser l'association entre un objet et son traqueur et de garantir que cette association est maintenue tout au long de la vie de l'objet. Cette problématique, aussi appelée *Data Association*, peut être abordée de plusieurs manières.

L'algorithme Hongrois (Kuhn, 1955) est une approche s'attachant à résoudre de manière optimale les problèmes d'associations en les formalisant comme une minimisation du coût total d'association. Notons que cet algorithme peut être approximé par une simple heuristique gloutonne réalisant à chaque itération la meilleure association, jusqu'à ce qu'il n'y ait plus d'association de qualité suffisante (Breitenstein et al., 2011; Di Lascio et al., 2013). Cette simplification lorsqu'elle est acceptable (non optimale) peut avantageusement réduire les temps de calcul nécessaires pour réaliser les associations.

Contrairement à ces algorithmes déterministes, l'approche du *Multiple Hypothesis Tracker* (Reid, 1979) est une approche probabiliste plus complexe qui propose de conserver en mémoire toutes les associations possibles (hypothèses) issues des événements passés. Ceci afin de pouvoir, en cas d'ambiguïté de décision, retarder la prise de décision à un instant ultérieur à la lumière des nouvelles observations. En pratique, il n'est cependant pas possible de garder en mémoire autant d'informations (augmentation exponentielle du nombre d'hypothèses avec le temps), aussi, une politique d'élagage est nécessaire afin de limiter le nombre d'hypothèses conservées.

Une autre approche probabiliste d'association courante est celle proposée par Fortmann et al. (1983) sous le nom *Joint Probability Data Association Filter*. Dans celle-ci, à chaque observation est associée un tableau de valeurs correspondant aux probabilités que chacun des objets (ou le fond) ait donné naissance à cette mesure. Cependant, cette

approche suppose connu le nombre d'objets.

2.3.4 Gestion des occultations

À part certains cas où les conditions d'acquisition permettent d'éviter les occultations (cas d'une scène dégagée vue par une caméra dont l'objectif pointe vers le sol à la verticale), il n'est pas rare de constater que des objets subissent des occultations totales ou partielles de plus ou moins longue durée, que soit dû à des occultations entre objets mobiles (piétons marchant en groupe, croisements, *etc.*) ou avec des éléments fixes de la scène (bâtiments, mobilier urbain). Ces occultations perturbent la qualité des détections (omissions, fausses détections) et nécessitent un traitement spécifique de la part de l'algorithme de suivi afin de compenser ces défauts.

Deux grandes familles d'approches se dégagent afin de gérer ce problème des occultations inter-objets dans des conditions d'acquisition mono-caméra.

Nous avons d'une part, les approches basées sur les notions de *merge* et *split* (fusion et scission) afin de grouper les objets suivis intervenant dans une occultation (*merge*) et de réidentifier ceux-ci à la fin de l'occultation (*split*). Ce mécanisme permet notamment d'éviter le suivi individuel des objets durant les phases d'occultation (où la qualité des détections n'est pas suffisante pour un tel suivi), au profit d'un éventuel suivi collectif temporaire, plus adapté, le temps de l'occultation. Bose et al. (2007) constatent que les détections issues du masque de mouvement ne sont pas parfaites et fragmentent ou fusionnent plusieurs objets. Ils proposent donc une approche à base de graphes afin de gérer ces défauts de détections. Motivés par les mêmes constats, Di Lascio et al. (2013) proposent un mécanisme de suivi collectif des objets lors d'occultations, via l'utilisation de groupes. Leur méthode est également capable de réidentifier correctement les objets individuels à la fin des occultations.

D'autre part, il existe des approches qui raisonnent explicitement sur la visibilité des objets (ou parties d'objets) afin de pouvoir continuer à suivre les objets partiellement occultés. Wu and Nevatia (2006) utilisent par exemple des détecteurs de parties de personnes (corps entier, tête/épaule, torse, jambes) et fusionnent les résultats afin de réduire la sensibilité de leur méthode aux occultations partielles. Greenhill et al. (2008) modélisent explicitement la profondeur en chaque pixel pour ainsi raisonner sur la visibilité des objets. Andriyenko et al. (2011) intègrent dans leur approche globale de suivi une formulation analytique du recouvrement entre objets participant à une occultation.

Notons également la possibilité de combiner différents capteurs afin de réduire les ambiguïtés liées aux occultations. Par exemple, l'utilisation de plusieurs caméras à champ recouvrant. Cette configuration permet de pouvoir capturer la scène selon plusieurs

points de vue différents et donc réduire l'incertitude liée aux occultations présentes dans le cas mono-caméra. On pourra alors raisonner sur une carte commune de positions et ainsi réduire les erreurs d'occultations (Fleuret et al., 2008).

Un cas particulier de cette configuration est celle de la stéréovision où deux caméras synchronisées et ayant des positions relatives connues capturent la scène et permettent de reconstruire une carte de profondeur permettant d'aider à la résolution des occultations (Munoz-Salinas et al., 2007). Il est également possible de coupler la caméra à un capteur de distances type laser, afin d'obtenir une mesure de la profondeur directement sans passer par la stéréovision (Goyat et al., 2010).

2.3.5 Conclusion sur les méthodes de suivi

Nous avons présenté et structuré différentes méthodes permettant de réaliser le suivi d'objets au travers de trois points différentiant : la stratégie d'inférence des trajectoires, la capacité à suivre un ou plusieurs objets et, la gestion des occultations.

Dans notre contexte, seule les approches d'inférence de trajectoires itératives sont envisageables étant donné le besoin de résultats immédiat pour l'analyse en temps-réel. Toutefois, nous notons les vertus des techniques "hybrides" à base de *tracklets* qui permettent de traiter de manière itérative les situations simples et n'utilisent l'optimisation globale que pour la fusion de ces *tracklets*.

Par ailleurs, il est attendu de notre algorithme d'être capable de suivre plusieurs objets sans *a priori* sur leur nombre et donc de gérer les entrées et sorties de manière automatique. C'est pourquoi nous ne pouvons utiliser les méthodes de *Visual Tracking* classiques. Parmi les méthodes de suivi multi-objets, nous privilégierons la simplicité des approches à base de suivi individuel et d'association heuristique des données. Dans ce contexte la méthode proposée par Di Lascio et al. (2013) est remarquable car elle suit effectivement chaque objet de manière indépendante tout en autorisant certaines interactions via l'utilisation des groupes. Ce mécanisme de groupe permet également de gérer les occultations.

Concernant la gestion des occultations, les deux familles d'approches mono-caméra présentées, à savoir, gestion explicite ou gestion par *merge/split*, sont recevables. Notre préférence va cependant aux méthodes par *merge/split* qui sont pensées explicitement pour compenser les défauts des segmentations obtenues par modèles de fond.

Parmi toutes les méthodes présentées, la méthode proposée par Di Lascio et al. (2013) est celle qui s'approche le plus des conditions de cette thèse, c'est pourquoi nous utiliserons et étendrons cette méthode au contexte de la vidéosurveillance (suivi d'objets de tout type).

2.4 Conclusion et propositions

Dans ce chapitre nous avons proposé une revue des principales méthodes concernant les trois domaines suivants : détection d'objets mobiles, suppression d'ombre et suivi d'objets. Loin d'être exhaustif, cet état de l'art nous a cependant permis de dégager les grandes tendances et compromis sous-jacents à l'application de ces méthodes.

Du fait des contraintes d'exécution en "temps réel" liées à la vidéosurveillance et le besoin de ne pas faire d'hypothèses trop fortes sur les types d'objets à détecter, nous avons choisi de baser notre travail sur l'utilisation de modèles de fond afin de détecter les objets mobiles de la scène, en particulier les modèles ViBe (Barnich and Van Droogenbroeck, 2011) et SLDP (Yoshinaga et al., 2013) qui ont démontré de bonnes performances dans les évaluations récentes (Brutzer et al., 2011; Vacavant et al., 2013).

Ces détections issues de modèles de fond étant sensibles aux conditions d'éclairage et détectant souvent à tort des ombres, nous utilisons une méthode permettant de supprimer ces dernières du masque de mouvement. La méthode de suppression proposée est géométrique et s'appuie sur une modélisation de la scène observée et de la caméra, construite à partir des études de prédéploiement des caméras.

En ce qui concerne le suivi des objets à partir de détections issues de l'étape de soustraction de fond, nous avons choisi de travailler en particulier sur la méthode proposée par Di Lascio et al. (2013) qui prend en compte explicitement les défauts liées aux détections issues des masques de mouvement et affiche de bonnes performances tout en étant compatible avec la contrainte d'exécution en temps réel. Nous avons en conséquence étendu ces travaux afin de pouvoir suivre n'importe quel type d'objet (pas seulement des piétons) modélisable par un cuboïde se déplaçant au sol.

Enfin, nous proposons d'utiliser les résultats du suivi des objets afin d'améliorer les détections du modèle de fond pour les instants suivants.

Chapitre 3

Modèle de scène 3d et caméra : contexte issu des études de prédéploiement

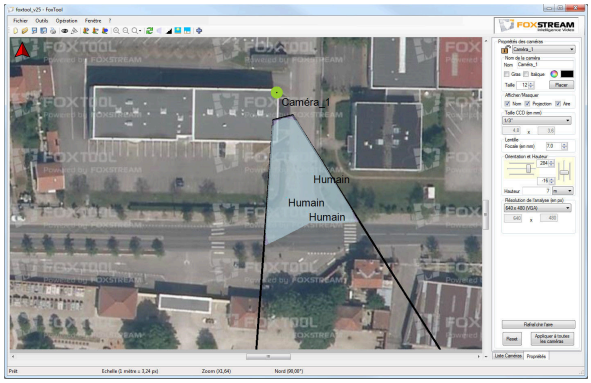
Les caméras de vidéosurveillance sont disposées stratégiquement dans leur environnement conformément à la fonction de sécurité à assurer. Ce placement, réalisé préalablement au déploiement des caméras, obéit aux contraintes, parfois contradictoires, suivantes :

- Garantir des conditions optimales d'observation sur les zones à observer (taille des objets observés suffisante, objets intégralement dans le champ de vision de la caméra, champ de vision dégagé, *etc.*).
- Minimiser le coût d'installation (nombre de caméras, tirage de câble, rapport qualité/prix des caméras)

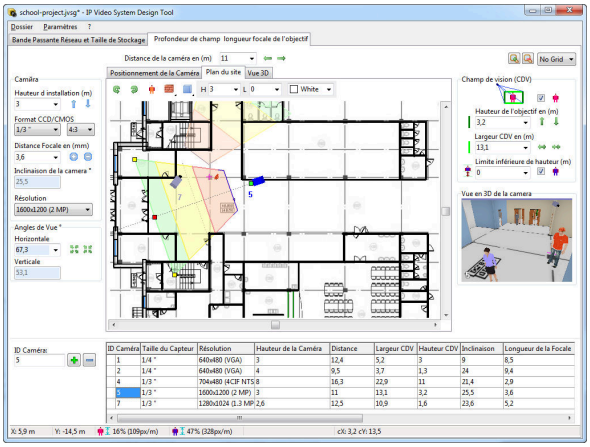
C'est pourquoi les personnes réalisant les études de prédéploiement des caméras sont assistées par un logiciel permettant de simuler les caméras et ainsi calculer les zones d'observation optimales (voir captures d'écran figure 3.1). Dans ces logiciels, l'utilisateur fournit un plan du site à surveiller et renseigne les paramètres des caméras ainsi que les obstacles potentiels et observe en retour les zones couvertes par les caméras. Les études ainsi réalisées servent généralement à produire les devis et consignes d'installation à destination des installateurs.

Ce que nous proposons est de réinvestir ces études de prédéploiement afin d'améliorer la détection et le suivi des objets. En particulier, nous exportons les modèles de scène et de caméra renseignés par l'utilisateur afin de pouvoir simuler ce que voient les caméras (rendus synthétiques), et ainsi inférer les positions et les tailles 3d réelles des objets suivis durant l'analyse.

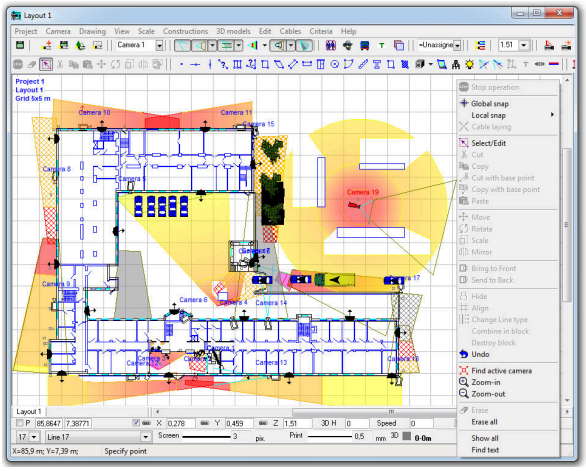
L'intérêt de cette démarche est que les études de prédéploiement sont disponibles avant tout traitement (données disponibles *a priori*), et ne nécessitent pas de travail supplémentaire de la part des utilisateurs étant donné que, si l'on respecte les bonnes



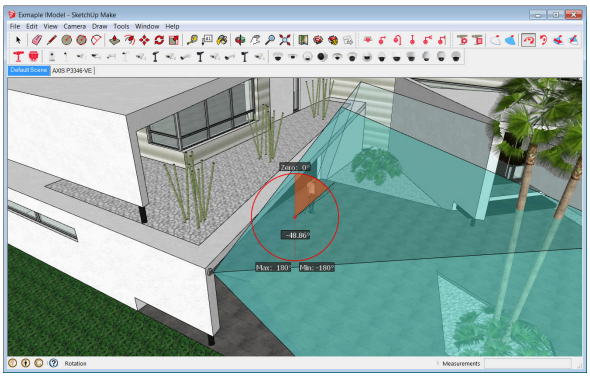
(a) FoxTool



(b) JVSG



(c) VideoCAD



(d) Extension AXIS pour Sketchup

FIGURE 3.1 Exemples de logiciels permettant de réaliser les études de prédéploiement. Les marques citées appartiennent à leurs propriétaires respectifs.

pratiques, un devis est nécessaire avant déploiement sur site.

Nous commencerons ce chapitre par la présentation du modèle de caméra utilisé (section 3.1), puis nous décrirons le modèle de scène développé en section (3.2). Enfin, nous montrerons comment sont intégrés et utilisés ces modèles dans le cadre de la vidéosurveillance en section (3.3).

3.1 Modèle de caméra

L'établissement d'un modèle de caméra permet de décrire le processus de formation des images vues par les caméras à partir d'une description 3d de son environnement. Ces modèles de caméra prennent en compte les propriétés des caméras telles que leurs position et orientation dans l'environnement, ainsi que leurs caractéristiques optiques.

Nous proposons ici d'utiliser le modèle de caméra sténopé augmenté d'une prise en compte de la distorsion dues aux lentilles (Brown, 1971; Hartley and Zisserman, 2004). Ce choix de modèle permet une utilisation avec un très grand nombre de caméras et peut gérer les cas où la distorsion n'est plus négligeable.

3.1.1 Modèle de caméra sténopé

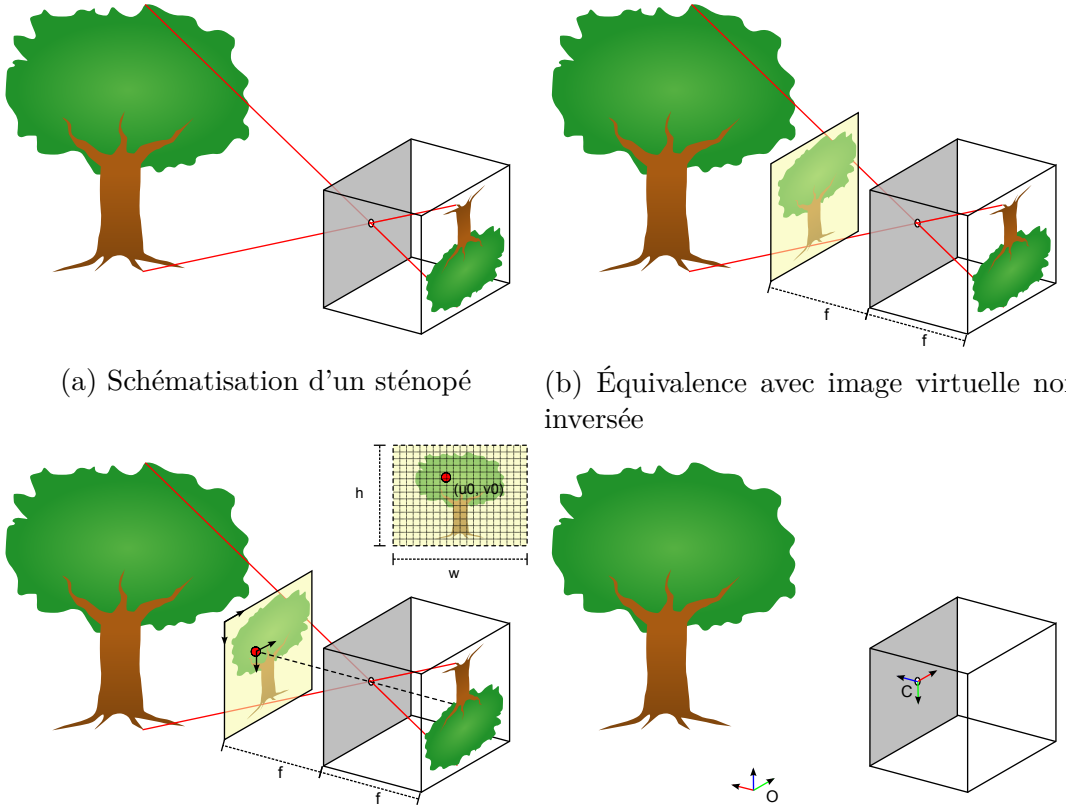
Un sténopé est un dispositif optique simple composé essentiellement d'une boîte fermée dont l'une des faces est percée par un petit trou laissant passer la lumière. En disposant un capteur photosensible sur la face opposée au trou, on peut obtenir l'image inversée de l'environnement extérieur (figure 3.2a).

La caractéristique essentielle d'un sténopé est qu'il ne contient pas de lentille (seulement un trou) et donc les rayons de lumière passant par le trou ne sont pas déviés [1]. Ainsi un point 3d réel, sa projection sur le plan image et le centre optique sont alignés. C'est cette caractéristique fondamentale qui est utilisée pour formuler le modèle de caméra sténopé.

L'image obtenue sur le fond du sténopé est inversée. Il est possible de raisonner de manière équivalente sur l'image virtuelle non inversée située symétriquement par rapport au trou comme montré sur la figure (3.2b). La distance f séparant le centre optique (trou) du plan du capteur (aussi appelé plan focal) est appelée distance focale.

Pour une caméra numérique, l'image finale est une image matricielle formée de pixels (voir figure 3.2c), il convient donc de décrire également la transformation qui permet de

1. Nous faisons l'hypothèse que le diamètre du trou est grand devant les longueurs d'ondes du faisceau lumineux pour nous placer dans le cadre de l'optique géométrique.



(a) Schématisation d'un sténopé

(b) Équivalence avec image virtuelle non inversée

(c) Formation de l'image matricielle

(d) Repère scène (O) et repère caméra (C)

FIGURE 3.2 Illustrations du modèle de sténopé.

convertir les coordonnées du plan de l'image virtuelle en coordonnées image exprimées en pixels. Cette conversion tient compte de la position (u_0, v_0) de l'axe optique par rapport à l'image, et de la résolution $(w \times h)$ de celle-ci.

Les équations de projection d'un point de coordonnées 3d de la scène (x_w, y_w, z_w) exprimées dans le repère de la scène sur le pixel (u, v) de l'image sont alors données par :

$$\begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + T \tag{3.1}$$

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} x_e/z_e \\ y_e/z_e \end{pmatrix} \tag{3.2}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x x_n + s y_n + u_0 \\ f_y y_n + v_0 \end{pmatrix} \tag{3.3}$$

L'équation (3.1) permet de passer d'un point (x_w, y_w, z_w) dans le repère de la scène à un point (x_e, y_e, z_e) dans le repère caméra (voir figure 3.2d) par une transformation rigide composée d'une rotation décrite par la matrice R de taille 3×3, et d'une translation décrite par le vecteur T. R et T sont appelés paramètres extrinsèques, et rendent compte de la position et de l'orientation de la caméra dans la scène.

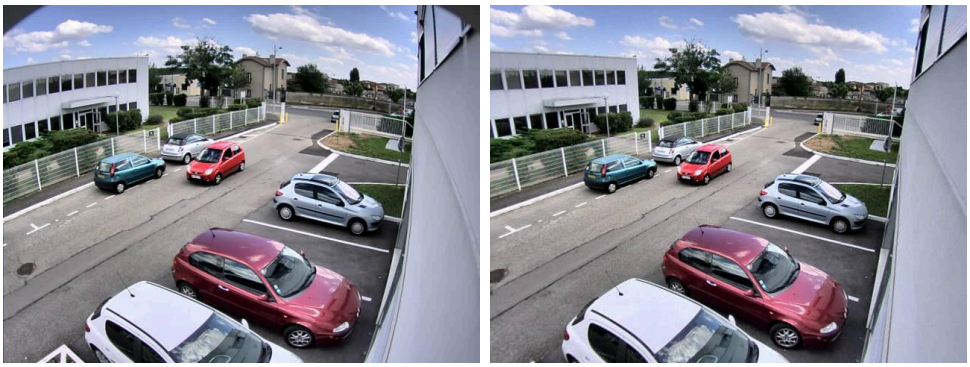
L'équation (3.2) réalise la projection, et permet de passer des coordonnées caméra $(x_e, y_e, z_e)^\top$ aux coordonnées normalisées $(x_n, y_n)^\top$.

L'équation (3.3) convertit les coordonnées normalisées (x_n, y_n) en coordonnées image exprimées en pixels (u, v) ; cette transformation est dite intrinsèque car elle ne dépend que des caractéristiques propres de la caméra, indépendamment de sa position ou son orientation.

Dans cette équation, $f_x = f m_x$ et $f_y = f m_y$ traduisent les distances focales exprimées en pixels, avec m_x et m_y les facteurs d'échelle et f la distance focale. Ces facteurs d'échelles m_x et m_y permettent de rendre compte de la densité de pixel dans les dimensions horizontale et verticale du capteur. Notons que pour des capteurs ayant des pixels carrés $f_x = f_y$.

s (aussi appelé paramètre *skew*) permet de représenter l'inclinaison caractéristique de certains capteurs à balayage. En pratique, cependant, nous le négligerons systématiquement en le considérant égal à zéro.

(u_0, v_0) décrivent la position de l'intersection de l'axe optique sur l'image (point principal). Ces coordonnées sont exprimées en pixels, et correspondent idéalement au centre de l'image.



(a) Image distordue (b) Image corrigée

FIGURE 3.3 Exemple de distorsion observée (3.3a) et image où cette distorsion a été supprimée (3.3b). Les contours des bâtiments sont courbés dans l'image de gauche, alors qu'ils sont rectilignes sur l'image de droite.

Notons, qu'il est également possible de reformuler ces équations sous forme matricielle par l'égalité à un facteur d'échelle λ près de la manière suivante (Hartley and Zisserman, 2004) :

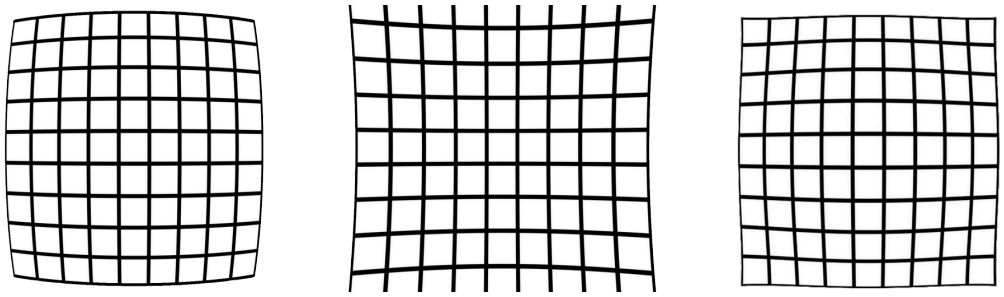
$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{3.4}$$

Avec $r_{i,j}$ les éléments de la matrice de rotation R et t_x, t_y et t_z les composantes du vecteur de translation T. Dans cette équation, la matrice 3×3 de gauche est la matrice intrinsèque ; et la matrice 4×4 est la matrice extrinsèque.

3.1.2 Prise en compte de la distorsion

Le modèle idéal de caméra sténopé néglige l'influence des lentilles qui se matérialise notamment par la présence de distorsions radiales et tangentielles dans l'image : le caractère rectiligne des droites dans le monde réel est perdu lors de la projection et ces droites apparaissent courbées sur l'image. Cet effet est de plus en plus marqué à mesure que l'on s'éloigne du centre de l'image (ou plus précisément centre de distorsion), et apparait surtout avec des caméras à courtes focales, comme nous pouvons le voir sur la figure 3.3.

La distorsion due aux lentilles d'une caméra prend généralement l'une des trois formes décrites dans ce qui suit. Afin de visualiser l'effet de ces différentes formes de distorsion, nous illustrons comment chacune d'elle déforme une grille régulière (voir figure 3.4).



(a) Distorsion en barillet (b) Distorsion en coussinet (c) Distorsion en moustache

FIGURE 3.4 Illustration des motifs de distorsion les plus courants

• **Distorsion en barillet (figure 3.4a) :** le grossissement de l'image diminue avec l'éloignement à l'axe optique. Ceci se traduit par une incurvation des lignes de la grille en direction du centre de l'image.

• **Distorsion en coussinet (figure 3.4b) :** le grossissement de l'image augmente avec l'éloignement à l'axe optique. Ceci se traduit par une incurvation des lignes de la grille en direction des bords de l'image.

• **Distorsion en moustache (figure 3.4c) :** c'est une combinaison des deux types de distorsion précédents. La distorsion commence comme une distorsion en barillet au centre de l'image, et évolue progressivement vers une distorsion en coussinet à mesure que l'on s'approche des bords de l'image.

Afin de décrire analytiquement ce phénomène, nous utilisons le modèle de distorsion suivant qui est une variante du modèle proposé par Brown (1971) [2] exprimant les coordonnées normalisées d'un point après distorsion $(x_d, y_d)^\top$ à partir de ses coordonnées normalisées avant distorsion $(x_n, y_n)^\top$:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + k_1 r_n^2 + k_2 r_n^4) \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} 2p_1 x_n y_n + p_2(r_n^2 + 2x_n^2) \\ 2p_2 x_n y_n + p_1(r_n^2 + 2y_n^2) \end{pmatrix} \tag{3.5}$$

Avec $r_n^2 = x_n^2 + y_n^2$ traduisant la distance du point à l'axe optique, k_1, k_2 les paramètres de distorsion radiale et p_1, p_2 les paramètres de distorsion tangentielle.

Afin de comprendre intuitivement cette équation, considérons le cas où $k_2 = p_1 = p_2 = 0$. Si $k_1 = 0$, alors $(x_d, y_d)^\top = (x_n, y_n)^\top$ et il n'y a donc pas de distorsion. Si $k_1 > 0$, alors les coordonnées $(x_d, y_d)^\top$ s'éloignent du centre, c'est donc une distorsion en coussinet. Si $k_1 < 0$, alors les coordonnées $(x_d, y_d)^\top$ se rapprochent du centre, c'est donc

2. Brown s'intéresse à la correction des effets de la distorsion et exprime donc les coordonnées non distordues en fonction des coordonnées distordues.

une distorsion en barillet. Le cas de distorsion en moustache peut être obtenu quand k_1 et k_2 sont de signes opposés.

Pour intégrer la prise en compte de la distorsion dans le modèle de caméra présenté dans la partie précédente, il faut remplacer (x_n, y_n) par (x_d, y_d) dans l'équation (3.3). Notons qu'avec l'introduction de cette équation, il n'est plus possible de formuler matriciellement le modèle de caméra, car l'équation de distorsion n'est pas linéaire.

3.2 Modèle de scène

Après avoir présenté le modèle de caméra permettant de décrire le processus de formation des images, nous allons présenter le modèle de scène vu par la caméra.

Le modèle que nous proposons ici consiste principalement en une modélisation géométrique du sol et des obstacles fixes (typiquement des bâtiments) posés sur celui-ci. Ce modèle se destine principalement aux scènes en extérieur, mais est éventuellement utilisable sous certaines conditions pour des scènes en intérieur.

Comme dans tout modèle, un compromis entre expressivité du modèle et difficulté de construction a dû être réalisé. Nous avons privilégié la simplicité d'élaboration, afin d'être capable de le construire à partir d'informations simples disponibles durant les études de prédéploiement des caméras.

Nous présenterons donc dans ce qui suit le modèle de scène élaboré, puis nous décrirons un moyen semi-automatique afin de l'initialiser, dans le cadre d'une scène en extérieur, à l'aide de simples coordonnées GPS, grâce à la base de données OpenStreetMap.

3.2.1 Modélisation

Le sol

Nous faisons l'hypothèse que le sol est plan, et par conséquent, nous le modélisons par le plan d'équation $z = 0$.

Ce choix de modélisation simple est motivé par le fait qu'il s'applique à beaucoup de situations rencontrées en pratique (surveillance d'un parking ou d'un lieu public, ou des routes) sans nécessiter de renseignements de la part de l'utilisateur.

En effet, il aurait été possible d'utiliser un modèle plus complexe à base de maillage, afin de pouvoir prendre en compte les pentes ou les irrégularités du sol de la scène. Cependant il aurait fallu également proposer un moyen de construire ce maillage simplement.

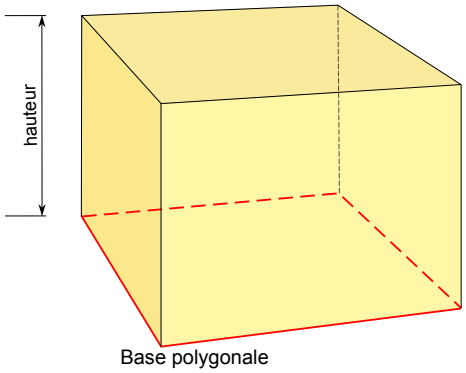


FIGURE 3.5 Illustration du modèle de bâtiment. Un bâtiment consiste en une base polygonale extrudée verticalement et une hauteur.

Une base de données topographiques (telle que celle proposée par l'IGN [3]) aurait pu être utilisé dans les cas d'application en extérieur, mais la résolution spatiale de ces bases n'est généralement pas adaptée aux scènes surveillées. L'IGN propose, par exemple, en libre accès une base topographique de l'élévation du sol avec une précision pour l'altitude de l'ordre du mètre en maillant la France avec des facettes de $75 \times 75m$. Cette résolution n'est cependant pas suffisante pour améliorer la qualité de la modélisation du sol, et est bien plus complexe que la modélisation plane du sol retenue.

Les bâtiments

Les bâtiments sont modélisés par des polygones extrudés verticalement posés sur le sol : ils sont donc constitués d'une base polygonale et d'une hauteur, comme illustré à la figure (3.5).

Cette modélisation permet de représenter une large gamme de bâtiments simples rencontrés en pratique. Tel qu'exprimé le modèle ne permet de modéliser directement que des bâtiments à toit plat (*i.e.* de hauteur uniforme). Il est cependant possible, en combinant plusieurs polygones extrudés verticalement avec des hauteurs différentes, de modéliser des formes un peu plus complexes présentant un toit de hauteur non uniforme. Mais cela nécessite de renseigner plus d'informations de la part de l'utilisateur.

De la même manière que pour la modélisation du sol, nous avons privilégié la simplicité du modèle par rapport à sa précision. En effet, dans le modèle retenu, l'utilisateur n'a qu'à renseigner les coordonnées de la base du bâtiment considéré et une hauteur. À noter que lorsque l'on dispose d'un plan de masse de la zone à surveiller, ce qui est le cas lors des études de prédéploiement des caméras, renseigner ces coordonnées peut être

3. http://professionnels.ign.fr/bdalti

grandement facilité, le plan de masse guidant la saisie de ces coordonnées.

Dans les cas simples, où seulement quelques bâtiments apparaissent dans le champ de vision des caméras, renseigner manuellement les coordonnées des bâtiments peut être envisageable. Cependant, cette démarche montre ses limites et devient fastidieuse et chronophage dès que le nombre de bâtiments s'élève à plus de quelques unités.

Aussi nous avons proposé d'utiliser la base de données géographique OpenStreetMap [4] afin d'automatiser le renseignement des bâtiments proches de la scène. Pour l'utilisateur, cela ne nécessite que de préciser les coordonnées GPS d'un point de la scène, comme nous allons le voir dans la partie suivante.

3.2.2 OpenStreetMap

Généralités

OpenStreetMap est un projet de cartographie libre du monde qui met ainsi à disposition de tous et de manière gratuite une très grande quantité de données géographiques. Parmi ces données géographiques, on retrouve des frontières (pays, régions, villes, *etc.*), des routes, divers points d'intérêts (station essences, magasins, *etc.*) et, ce qui nous intéresse particulièrement ici, des bâtiments.

Cette richesse d'information est possible grâce aux contributions faites par la communauté construite autour de ce projet. En effet, à l'instar du projet Wikipedia [5], toute personne peut contribuer à l'amélioration de la base de données OpenStreetMap.

L'utilisation principale de cette base de données géographique est de produire des cartes à partir des données géographiques collectées comme illustré à la figure 3.6. Cependant, d'autres utilisations sont rendues possibles grâce à la fonction d'export des données OpenStreetMap au format XML. Ainsi, des applications de planification de routes [6], de simulation de trafic (Zilske et al., 2011; Dallmeyer et al., 2014) ou même l'intégration de ces données dans un jeu de course [7] ont pu être développées.

Représentation d'un bâtiment

Fondamentalement, OpenStreetMap, encode les informations géographiques à l'aide des 4 primitives suivantes :

4. http://www.openstreetmap.org
5. https://www.wikipedia.org
6. http://wiki.openstreetmap.org/wiki/Routing
7. http://wiki.openstreetmap.org/wiki/Supertuxkart

FIGURE 3.6 Exemple de carte générée à partir de la base de données OpenStreetMap. Sur cette carte figurent les rues avec leur nom, et les bâtiments éventuellement avec leur numéro.

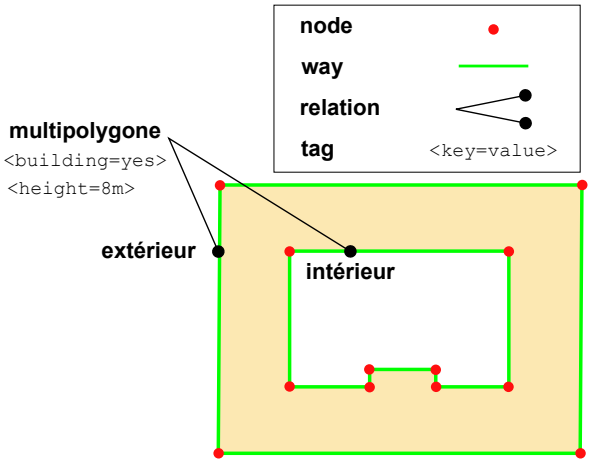
• **Un nœud (`node`)** représente une position d'un point sur terre. Il est décrit par une latitude, une longitude, un identifiant et optionnellement une élévation. C'est la primitive fondamentale permettant d'exprimer des coordonnées sur terre.

• **Un chemin (`way`)** est une ligne brisée composée d'un ou plusieurs nœuds. Lorsque le premier nœud et le dernier nœud d'un chemin sont identiques, le chemin est dit fermé, autrement c'est un chemin ouvert. Les chemins ouverts servent principalement à représenter les routes, alors que les chemins fermés permettent de délimiter des régions, en particulier les contours simples des bâtiments.

• **Une relation (`relation`)** permet de définir un lien logique ou géographique entre un ou plusieurs nœuds et/ou chemins. Pour ce qui nous intéresse, une relation permet de regrouper les contours d'un même bâtiment en précisant si le contour est intérieur ou extérieur quand le bâtiment a une forme complexe à trous.

• **Un tag (`tag`)** est une paire clé/valeur encodant une méta-information à propos d'une autre primitive. En ce qui nous concerne, cela permet de préciser qu'un contour est un contour de bâtiment, et éventuellement de quel type de bâtiment il s'agit (appartements, maison, hôtel, cathédrale, *etc.*).

Nous donnons, à titre d'exemple, la représentation d'un bâtiment complexe fictif à la figure 3.7. Dans cet exemple, le bâtiment est modélisé par une relation liant deux contours l'un pour la partie extérieure, l'autre pour le trou intérieur. Cette relation est annotée par les tags `<building=yes>` et `<height=8m>` afin de préciser que c'est un bâtiment générique de 8 mètres de haut.



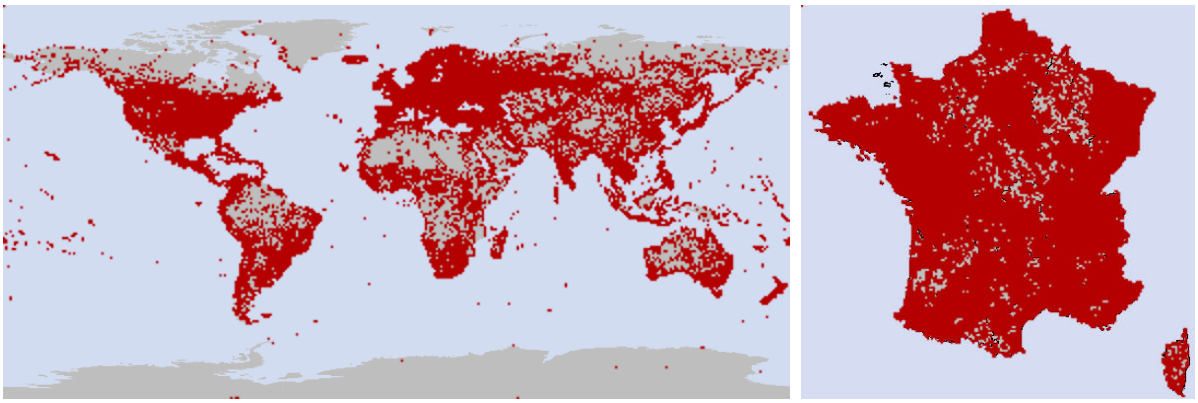

```
<osm>
        ...
        <node id="1" lat="..." lon="..."/>
        <node id="2" lat="..." lon="..."/>
        ...
        <node id="12" lat="..." lon="..."/>

        <way id="20">
                <nd ref="1"/>
                ...
                <nd ref="4"/>
        </way>

        <way id="21">
                <nd ref="5"/>
                ...
                <nd ref="12"/>
        </way>

        <relation id="30">
                <member type="way" ref="20" role="outer"/>
                <member type="way" ref="21" role="inner"/>
                <tag k="type" v="multipolygon"/>
                <tag k="building" v="yes"/>
                <tag k="height" v="8m"/>
        </relation>
</osm>
```

FIGURE 3.7 Modèle OpenStreetMap d'un bâtiment complexe avec listing XML abrégé correspondant.



(a) Monde (b) France

FIGURE 3.8 Répartition géographique des bâtiments répertoriés à l'échelle mondiale et à l'échelle française. Chaque point rouge, correspond à un bâtiment.

Le système de tags permet aussi de spécifier d'autres attributs, tels que les points d'entrées (`<entrance=node_id>`) ou le nombre d'étages (`<building:levels=n_étages>`).

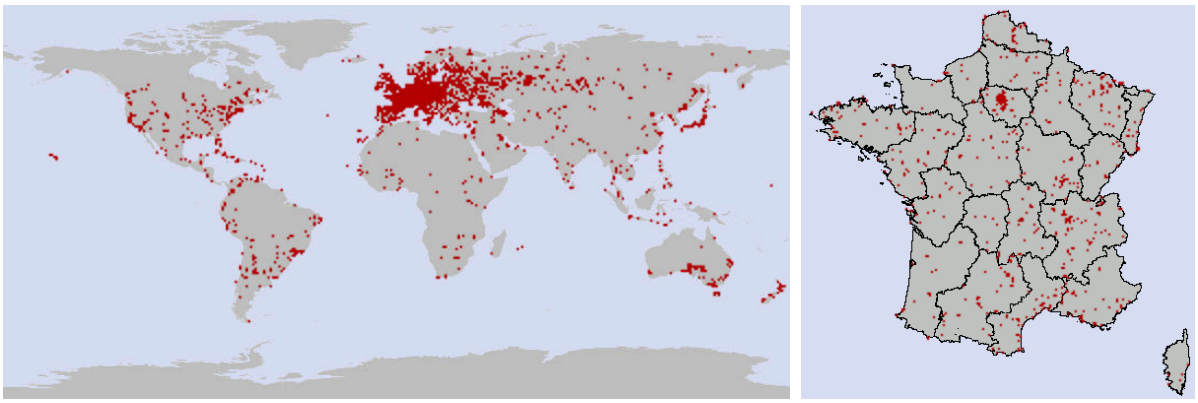
Il est important de noter que toutes les informations mentionnées dans ce qui précède ne sont pas nécessairement renseignées dans la base de données OpenStreetMap. Il n'y a en effet aucune garantie ni sur l'exactitude, ni sur l'exhaustivité des données renseignées. A minima, les bâtiments auront tous au moins un contour délimitant leur implantation sur le sol. Dans certains cas, la hauteur, ou le nombre d'étages seront renseignés, mais cela reste marginal.

Statistiques relatives aux bâtiments

Nous allons présenter dans ce qui suit un certain nombre de statistiques concernant les bâtiments afin d'évaluer la pertinence de cette base de données dans notre cas. Toutes ces statistiques sont extraites via les URL suivantes : http://taginfo.openstreetmap.org/keys/building#overview pour les statistiques mondiales, et http://taginfo.openstreetmap.fr/keys/building#overview pour la France seulement. Étant donné la vitesse d'évolution de la base de données OpenStreetMap, les chiffres donnés dans ce qui suit seront certainement obsolètes dans peu de temps. Aussi, nous précisons que ces statistiques ont été collectées en septembre 2014, et invitons le lecteur à consulter les nouvelles statistiques en allant consulter les adresses mentionnées ci-dessus.

À l'échelle mondiale, un peu plus de 120 millions de bâtiments sont répertoriés, et ce, sur l'ensemble des continents, comme en témoigne la carte de répartition à la figure 3.8a.

Si l'on s'intéresse plus particulièrement au cas de la France, on obtient la carte donnée



(a) Monde (b) France

FIGURE 3.9 Répartition géographique des hauteurs répertoriées à l'échelle mondiale et à l'échelle française.

à la figure 3.8b, avec près de 38 millions de bâtiments répertoriés (soit près d'un quart du nombre total de bâtiments sur l'ensemble du globe). Le cas de la France, particulièrement riche en terme de quantité de bâtiments recensés, est unique. Cette singularité s'explique par les contributions significatives faites grâce aux données issues du cadastre français. Cela a été possible grâce à la libération des données cadastrales par la direction générale des impôts et par la direction générale des finances publiques. De plus, nous constatons que des mises à jours régulières depuis 2009 intègrent les nouveaux bâtiments construits dans l'année et corrigent les erreurs sur les bâtiments existants.

Le fait que ces données soient issues du cadastre français apporte à la fois un certain gage de qualité (source des données officielle), et aussi de quantité comme en témoigne la couverture étendue sur le territoire français. Pour ces raisons, au moins pour une utilisation sur le territoire français, OpenStreetMap semble pertinent afin de construire un modèle de scène à partir d'une simple coordonnée GPS.

En revanche, il faut noter que très souvent les hauteurs des bâtiments ne sont pas renseignées comme en témoignent les cartes (3.9a) et (3.9b). En termes de chiffres, à l'échelle mondiale, seulement 1.6% des bâtiments ont une hauteur renseignée (environ 2 millions) ; pour la France ce pourcentage est estimé à 0.03% (environ 12000).

Actuellement, il est donc peu probable que la hauteur d'un bâtiment puisse être extraite automatiquement de la base de données OpenStreetMap. Cette donnée devra donc être obtenue par un autre moyen.

3.3 Intégration et utilisation en vidéosurveillance

Maintenant que nous avons présenté les modèles de caméra et de scène employés, voyons comment ceux-ci peuvent être intégrés et utilisés dans notre contexte de vidéosurveillance. Nous montrerons d'abord comment ces modèles sont construits par l'utilisateur, puis nous verrons quelques possibilités d'utilisation dans le cadre de la vidéosurveillance.

3.3.1 Les études de prédéploiement des caméras

Dans le contexte de la vidéosurveillance, les caméras ne sont pas placées au hasard. La position et l'orientation des caméras sont établies avant leur installation sur le site à surveiller. De plus ces données sont consignées dans un rapport et sont donc disponibles avant même l'installation des caméras. Nous proposons de mettre à profit ces informations afin de construire le modèle de scène et de caméra présentés dans les parties précédentes. L'intérêt principal de cette démarche est de pouvoir intégrer des informations sur la scène et les caméras sans demander de travail supplémentaire aux personnes réalisant le paramétrage des caméras de surveillance : toutes les informations sont importées directement à partir des études de prédéploiement des caméras.

Simulateur de caméras

Les études de prédéploiement des caméras sont réalisées à l'aide d'un logiciel de simulation de caméras. Ce simulateur interactif permet de placer des caméras, éditer leurs paramètres (placement, orientation, paramètres optiques), et visualiser ce que voit la caméra, ainsi que les zones effectivement surveillées. Une fois l'étude réalisée, l'utilisateur peut exporter son étude sous forme de rapport à destination des installateurs.

Une capture d'écran du simulateur employé par la société Foxstream est donnée à la figure 3.10. Sur cette figure, nous distinguons la fenêtre principale avec le plan de la zone et un volet de configuration des paramètres de la caméra, ainsi qu'une fenêtre de visualisation de ce que voit la caméra.

Sur le plan, la caméra placée est représentée par un disque vert ; le champ de vision est délimité par des traits noirs (forme trapézoïdale), et la zone effectivement couverte par la caméra est dessinée en bleu (cette zone tient compte des tailles minimales des objets à détecter ainsi que de leur surface apparente). Les bâtiments modélisés sont représentés en jaune pâle.

La fenêtre de la vue caméra affiche un rendu de la scène telle qu'elle est vue par la

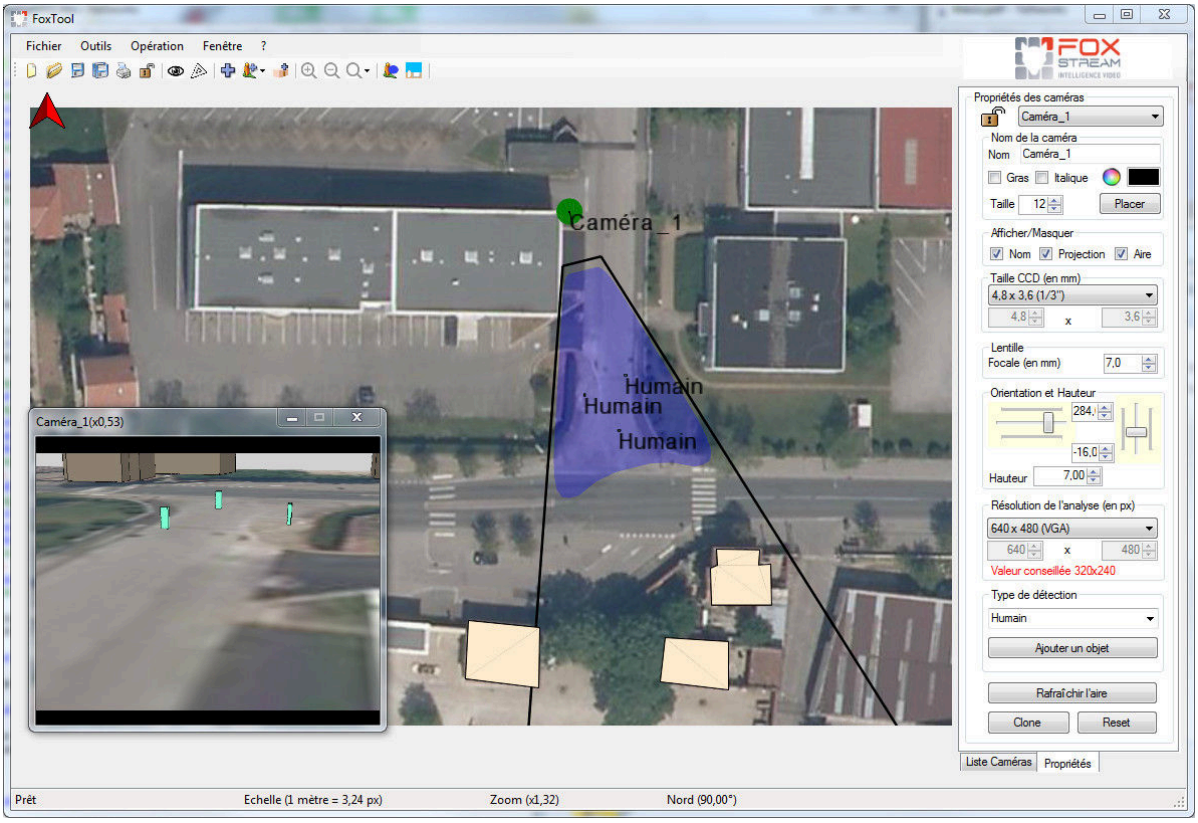


FIGURE 3.10 Capture d'écran du simulateur de caméra.

caméra. Cette vue tient compte des paramètres de la caméra et est mise à jour en temps réel afin de permettre aux utilisateurs de constater instantanément les effets de leurs modifications sur chacun des paramètres.

Paramétrisation simplifiée du modèle de caméra

Le modèle de caméra présenté en section 3.1 comporte 15 paramètres (3 de position, 3 d'orientation, 5 intrinsèques, et 4 de distorsion). Il est donc nécessaire d'aider l'utilisateur (qui n'est pas forcément un expert du domaine) à renseigner ces paramètres de manière intuitive. C'est pourquoi, le logiciel adopte une paramétrisation simplifiée du modèle de caméra [8].

Pour paramétrer une caméra, l'utilisateur spécifie la position de celle-ci en la déplaçant sur le plan, la hauteur pouvant être saisie numériquement par ailleurs. L'orientation de la caméra est spécifiée par ses angles de pan (orientation dans le plan horizontal) et de tilt (orientation dans le plan vertical). Pour renseigner ces angles, l'utilisateur peut soit spécifier directement les valeurs qu'il souhaite, soit désigner le point sur le plan qu'il souhaite observer. Concernant les paramètres optiques, l'utilisateur spécifie la focale de la caméra ainsi que la taille du capteur et la résolution d'acquisition.

Les valeurs des paramètres du modèle de caméra de la section 3.1 sont alors :

$$
\begin{aligned}
R &= \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix} \times R_y(\theta) \times R_z(-\psi) \\
T &= -RC \\
f_x &= f\frac{w}{l_x} \\
f_y &= f\frac{h}{l_y} \\
u_0 &= \frac{w}{2} \\
v_0 &= \frac{h}{2} \\
s &= 0 \\
k_1 &= k_2 = p_1 = p_2 = 0
\end{aligned}
\tag{3.6}
$$

Avec ψ l'angle de pan, θ l'angle de tilt, $R_z(\psi)$ la rotation d'axe z et d'angle ψ, $R_y(\theta)$

8. Toutefois, l'utilisateur expert pourra spécifier à sa guise l'ensemble des valeurs des paramètres du modèle décrit en section 3.1

la rotation d'axe y et d'angle θ. C sont les coordonnées du centre de la caméra exprimées dans le repère de la scène. w et h sont les résolutions horizontale et verticale de l'image acquise. l_x et l_y sont les dimensions horizontale et verticale du capteur d'image. Ces dimensions sont exprimées en millimètres tout comme f, la focale de la caméra. Dans notre paramétrisation simplifiée nous négligeons en première approximation les effets de distorsion, d'où les valeurs nulles pour le skew s et les coefficients de distorsion k_1, k_2, p_1, p_2.

Note : La paramétrisation donnée ici mène à des coordonnées images repérées par rapport au coin haut-gauche de l'image, et une profondeur (z_e) positive devant la caméra. C'est la convention utilisée par OpenCV. En revanche, pour effectuer les rendus 3d, nous utilisons OpenGL qui opère avec une autre convention : les coordonnées images sont repérées par rapport au coin bas-gauche de l'image, et la profondeur (z_e) est négative devant la caméra. Dans ce cas, il suffit de remplacer la matrice $\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix}$ par $\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix}$ dans l'équation donnant la valeur de R.

Paramétrisation du plan et import des données OpenStreetMap

Les études de prédéploiement sont réalisées à partir d'un plan de la zone à surveiller. L'utilisateur indique donc l'image à utiliser comme plan de zone et spécifie également l'échelle de celui-ci. Ceci permet de réaliser des mesures de distance directement sur le plan.

Pour des scènes en extérieur, l'utilisateur peut également renseigner la direction du nord et les coordonnées GPS de l'origine du repère de la scène. Ces indications permettent de localiser et orienter le plan. Il devient alors possible de calculer les coordonnées GPS en tout point du plan, et ainsi importer depuis la base de données OpenStreetMap les bâtiments existants dans cette zone.

Comme nous l'avons précisé lors de la présentation de la base de données OpenStreetMap, la hauteur des bâtiments n'est pas toujours renseignée. Nous avons donc enrichi le logiciel utilisé dans l'entreprise afin de pouvoir estimer la hauteur d'un bâtiment OpenStreetMap.

Quand la hauteur est spécifiée, nous utilisons la valeur fournie. Si la hauteur est absente, nous regardons si le nombre d'étages est fourni. Dans ce cas, nous estimons la hauteur totale du bâtiment en supposant que chaque étage fait 3m. Si aucune de ces

(a) Vue d'ensemble (b) Zoom sur la gare

FIGURE 3.11 Modélisation de la ville de Passau (Allemagne) à partir des données OpenStreetMap.

données ne sont fournies, nous attribuons une hauteur de 5m. L'utilisateur pourra éditer *a posteriori* ces valeurs en fonction de ses observations si besoin.

Nous donnons dans ce qui suit quelques exemples de plans obtenus par cette démarche. Le premier exemple que nous présentons est celui de la ville de Passau (Allemagne) à la figure 3.11. Dans cet exemple, les hauteurs des bâtiments n'étaient pas disponibles et ont donc été estimées à partir du nombre d'étages. Cet exemple montre qu'il est possible de créer simplement et rapidement un vaste modèle de scène (en moins de 5 minutes).

Le deuxième exemple (figure 3.12) montre le cas de bâtiments complexes où la hauteur était disponible. La forme obtenue n'est pas un toit plat grâce à la superposition de plusieurs "parties" de bâtiments ayant des hauteurs différentes. On voit, d'ailleurs, que la passerelle reliant les deux tours n'est pas modélisable correctement par notre modèle, étant donné qu'elle ne touche pas le sol.

Le troisième exemple (figure 3.13) montre les bâtiments situés aux abords de la société Foxstream (Vaulx-en-Velin, France). Dans cet exemple, aucune information de hauteur, ni de nombre d'étages n'étaient disponibles, c'est pourquoi la hauteur des bâtiments est uniforme. En revanche, cet exemple est intéressant, car il montre qu'il est possible de combiner différentes sources de données géographiques : le plan est issu d'une image satellite obtenue via http://www.geoportail.gouv.fr/, et les bâtiments sont issus de la base OpenStreetMap.

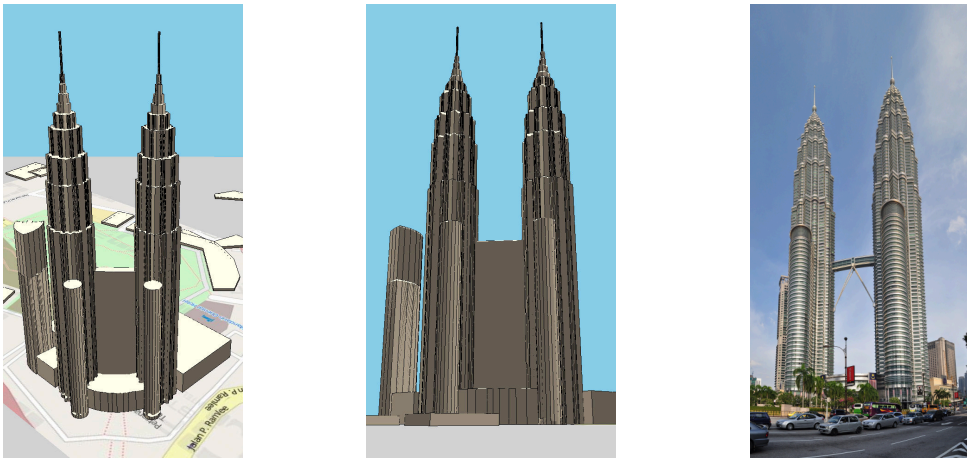
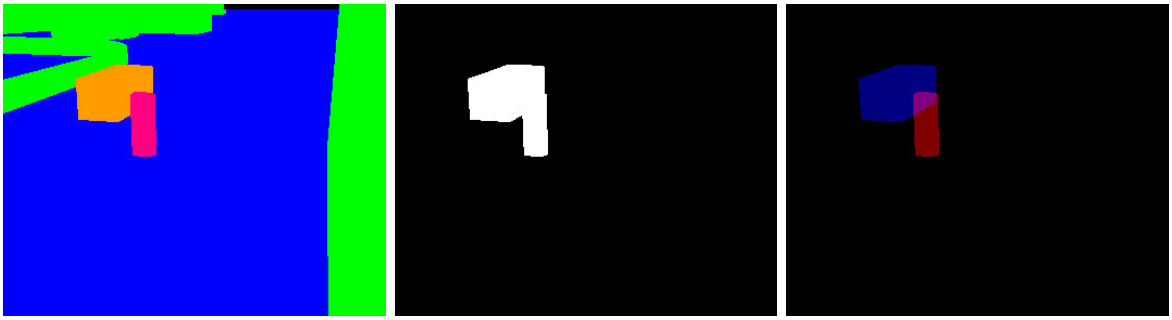


FIGURE 3.12 Tours jumelles Petronas (Kuala Lumpur, Malaisie). Rendus obtenus par FoxTool à gauche et au centre. Photo réelle à droite (Alexander Vonbun, 2009).

FIGURE 3.13 Bâtiments situés aux abords de la société Foxstream. (Vaulx en Velin, France)

(a) Normal

(b) Superposition des contours synthétiques sur image réelle



(c) Couleurs uniques

(d) Masque binaire

(e) Masques binaires multiplexés

FIGURE 3.14 Illustration de différents rendus synthétiques générés à partir des études de prédéploiement.

3.3.2 Rendus synthétiques

Les modèles de scène et de caméra développés se prêtent naturellement à la production de rendus synthétiques de la scène observée à partir d'une caméra. Aussi nous avons implémenté un module logiciel permettant d'effectuer de tels rendus en temps réel à l'aide de l'API OpenGL [9].

Nous avons déjà montré quelques possibilités de ce module lors de la présentation du logiciel de simulation et avec les exemples de scènes construites avec OpenStreetMap. Aussi nous ne reviendrons pas sur ces exemples. En revanche, nous allons montrer d'autres possibilités de rendus qui s'avèrent utiles lors de l'analyse (voir figure 3.14).

Le premier exemple de rendu (figure 3.14a) est celui du rendu dit "normal" qui

9. spécification normalisée d'un ensemble de fonctions permettant d'effectuer des rendus 3d - https://www.opengl.org/

correspond à ce que l'utilisateur a pu obtenir lors de son étude de prédéploiement. Par rapport aux données contenues dans l'étude de prédéploiement, nous avons ajouté deux pavés droits l'un représentant une voiture l'autre un piéton.

Sur la figure (3.14b), nous montrons qu'il est possible de générer des rendus des contours des objets de la scène et de les superposer à une image réelle. Dans cette image, les contours des bâtiments sont représentés en vert, les contours de la voiture et du piéton sont représentés en orange et rose. Ce type de visualisation est principalement destiné à vérifier la cohérence du modèle 3d vis-à-vis de la réalité.

Une autre possibilité intéressante afin de calculer les silhouettes des objets présents dans la scène est d'effectuer le rendu avec des couleurs uniques pour chaque objet (ou groupe d'objets) comme montré à la figure (3.14c). En effet, il devient alors possible d'identifier pour chaque pixel à quel objet il correspond à partir de sa couleur. Sur l'image d'exemple, les pixels bleus sont des pixels appartenant au sol, les pixels verts appartiennent à des bâtiments, les pixels oranges à une voiture, et les pixels roses à un piéton. Une application pratique de ce genre de rendu dans le simulateur de caméra est de pouvoir déterminer quel objet se trouve sous le pointeur de la souris : il suffit d'effectuer un rendu en couleurs uniques et de lire la couleur du pixel sous le curseur.

Un cas particulier de rendu en couleurs uniques est celui du rendu d'un masque binaire où les objets à visualiser sont dessinés en blanc, le reste en noir. L'exemple à la figure (3.14d) montre un masque binaire représentant la voiture et le piéton.

Enfin le dernier type de rendu que nous présenterons ici, est celui des masques binaires multiplexés. Le principe d'un tel rendu est de combiner plusieurs rendus de masques binaires sur une même image couleur. Chaque masque binaire réside alors dans un plan de bit [10] distinct de l'image. Prenons l'exemple donné à la figure (3.14e) où sont multiplexés le masque binaire de la voiture et celui du piéton. Cette image est une image couleur RGB 24 bits (8 bits par composante). Le masque de la voiture est dessiné sur le plan de bit 7 (bit le plus significatif de la composante bleu), et celui du piéton sur le plan de bit 23 (bit le plus significatif de la composante rouge). Ainsi la voiture apparait bleu, le piéton rouge et la zone d'intersection en violet (= rouge + bleu).

L'intérêt d'un tel rendu est de diminuer le volume de données à transférer entre la mémoire du processeur graphique et la mémoire centrale. En effet, lorsque l'on fait un rendu d'une image binaire seule, l'image obtenue est tout de même une image couleur (généralement sur 16, 24 ou 32 bits) mais avec seulement du blanc et du noir. Dans ces images, il y a donc une densité d'information utile très faible (1 seul bit par pixel, au lieu de 24 sur une image 24 bits). Ceci rallonge inutilement le temps de transfert de

10. un plan de bit est l'ensemble des bits à une position donné des valeurs des pixels de l'image

l'image entre le processeur graphique et la mémoire centrale. En multiplexant plusieurs images binaires sur une même image couleur, on réduit ainsi le nombre de transferts mémoire nécessaires. Par exemple pour une image 24 bits, on peut multiplexer jusqu'à 24 masques binaires, et donc diviser par 24 le nombre de transferts à réaliser. Ceci est d'autant plus intéressant dans notre cas, car ces transferts constituent le facteur limitant de la vitesse d'obtention d'un rendu.

3.4 Conclusion

Dans ce chapitre nous avons présenté les modèles de caméra et de scène utilisés. Ces modèles simples sont construits directement à partir des études de prédéploiement des caméras, et se prêtent naturellement à la production de rendus synthétiques de la scène vue par la caméra. Par ailleurs, de par leur nature géométrique, ces modèles permettent de raisonner en 3d comme nous allons le montrer dans le chapitre suivant en exposant une méthode permettant de prédire où sont projetées les ombres au cours de la journée dans une scène en extérieur et ainsi les filtrer du masque de mouvement.

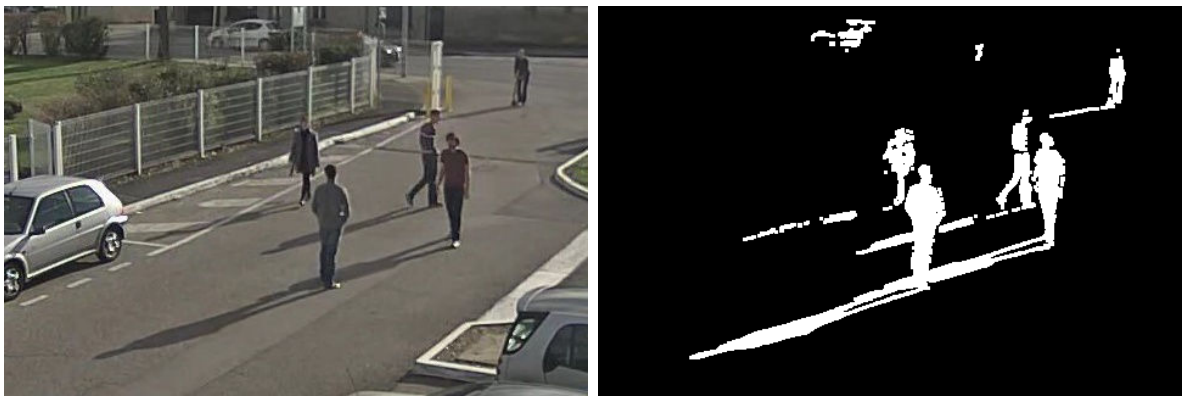
Chapitre 4

Ombres : prédiction et suppression

L'objectif de ce chapitre est d'exposer une méthode de suppression des ombres portées présentes dans le masque de mouvement. La méthode que nous proposons s'inscrit dans la continuité du modèle de scène présenté dans le chapitre 3 et tire parti des informations géométriques et géolocalisées qui s'y rapportent (modèle de scène 3d, coordonnées GPS, direction du Nord). En particulier, cette méthode permet de prédire les ombres projetées au sol dues au Soleil tout au long de la journée. En première application, nous proposons d'utiliser cette prédiction afin de supprimer les ombres portées des piétons présentes dans le masque de mouvement.

4.1 Introduction

Les détections d'objets obtenues par suppression de fond peuvent contenir également les ombres des objets segmentés (figure 4.1).



(a) Image couleur (b) Masque de mouvement

Figure 4.1 Exemple d'ombres segmentées avec les objets les projetant menant à une fusion d'objets normalement distincts.

Ces ombres nuisent à la qualité de la segmentation et réduisent l'applicabilité des méthodes de suppression de fond. Par ailleurs ces ombres perturbent également les descripteurs caractérisant les objets (taille, orientation, histogrammes de couleur, *etc.*).

C'est pourquoi nous proposons de supprimer les ombres portées du masque de mouvement afin d'améliorer la détection et le suivi des objets de la scène. Nous décrivons dans un premier temps la méthode de calcul des ombres portées des objets de la scène. Nous verrons ensuite comment calculer la position du Soleil au cours de la journée. Enfin nous présenterons une première méthode permettant de supprimer les ombres portées des piétons.

4.2 Ombres projetées

4.2.1 Ombre propre, ombre portée et pénombre

Une ombre est une zone sombre d'une surface qui n'est pas (ou peu) atteinte par la lumière. Elle résulte de l'interposition d'un objet opaque entre une source de lumière et une autre surface qui serait atteinte par la lumière s'il n'y avait pas d'obstacle au passage de la lumière. L'ombre se décompose alors en une partie "ombre propre" (zone d'ombre sur l'objet opaque lui-même) et une partie "ombre portée" (zones d'ombre projetées sur les autres surfaces de la scène). On distinguera également, dans les ombres portées, la zone de pénombre qui marque la frontière entre la zone d'ombre et la zone éclairée. La pénombre se manifeste dans le cas de sources de lumières étendues où seulement une partie de la source de lumière éclaire la surface, comme illustré à la figure (4.2).

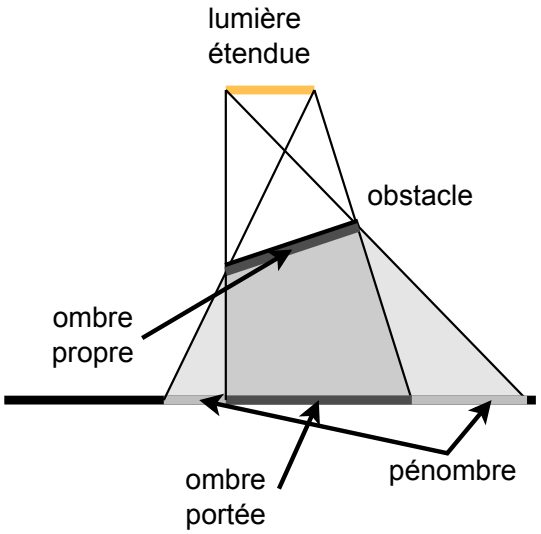


FIGURE 4.2 Ombre propre, ombre portée et pénombre

Dans ce qui suit, et pour le reste du manuscrit, nous faisons l'hypothèse simplificatrice de sources de lumières ponctuelles. Ce choix permet de négliger les zones de pénombres qui sont alors considérées comme totalement éclairées. Cette hypothèse est raisonnable pour les sources de lumières considérées dans ce manuscrit, principalement le Soleil. Celui-ci peut être en effet modélisé comme une source ponctuelle de lumière située à l'infini, c'est-à-dire comme une source de lumière directionnelle.

4.2.2 Méthodes classiques pour le calcul des ombres

Le calcul des ombres d'une scène peut être réalisé de plusieurs manières différentes en fonction du compromis entre précision et temps de calcul souhaité. Nous allons présenter rapidement les principales techniques de calcul des ombres, à savoir la technique du *shadow volume*, celle du *shadow mapping* et celle de la projection plane.

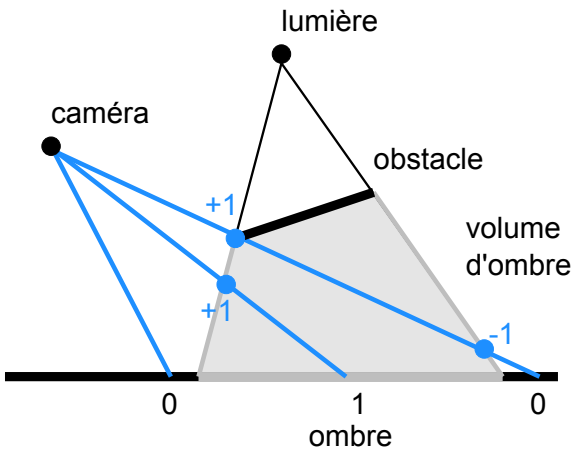


FIGURE 4.3 Illustration de la technique du shadow volume.

La technique du shadow volume (Crow, 1977), illustrée à la figure (4.3), consiste à calculer les volumes d'ombre de la scène. Ces volumes sont délimités par des rayons partant de la source lumineuse et passant par les bordures des objets de la scène. Une fois ces volumes calculés, pour savoir si un point est ombré ou éclairé, il suffit de savoir si ce point est dans un volume d'ombre ou non. Ce test d'appartenance à un volume d'ombre est résolu en comptant pour chaque rayon partant de la caméra le nombre de fois qu'il entre (+1) et qu'il sort (-1) d'un volume d'ombre avant d'arriver sur l'objet à dessiner : si ce compteur est nul, le rayon arrive sur une zone éclairée, sinon le rayon arrive sur une zone ombrée.

Cette technique géométrique nécessite cependant quelques améliorations afin de pouvoir gérer le cas où la caméra est dans un volume d'ombre.

La technique du shadow mapping (Williams, 1978), quant à elle, formule le problème de calcul des ombres sous la forme d'un problème de visibilité : les zones ombrées sont les zones qui ne sont pas visibles depuis la source lumineuse, comme illustré à la figure (4.4).

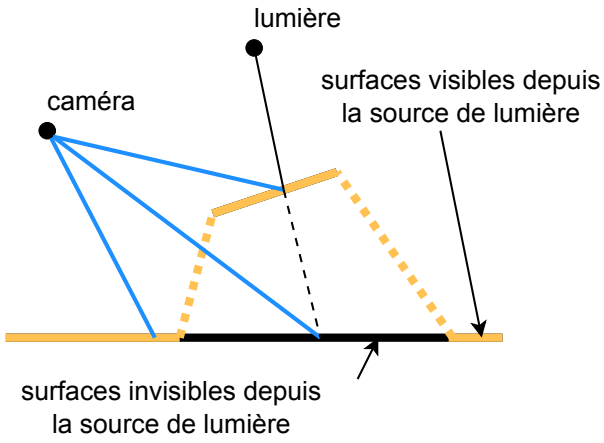


FIGURE 4.4 Illustration de la technique du shadow mapping

En pratique cet algorithme s'implémente de la manière suivante : un premier rendu de la scène vue depuis la source lumineuse est effectué. Lors de ce rendu seule la carte de profondeur est conservée (*depth buffer*). Ce *depth buffer* stocke donc la distance entre la source lumineuse et le premier obstacle rencontré par le rayon de lumière traversant un pixel donné. Ensuite, un second rendu de la scène vue depuis la caméra est effectué. Lors de ce rendu, les coordonnées des points à dessiner sont transformées afin d'être exprimées dans le repère de la source lumineuse. Ceci permet alors de pouvoir comparer la profondeur de chaque point avec celle correspondante dans le *depth buffer* issu du premier rendu : si les points sont situés avant le premier obstacle (valeur inférieure à celle contenue dans le *depth buffer*) le point est éclairé, sinon le point est ombré. Cette technique résout le problème de visibilité dans l'espace image en deux passes, mais nécessite des précautions particulières afin d'éviter les problèmes d'échantillonnage des ombres lors du premier rendu dans le *depth buffer*.

Enfin, Blinn (1988) propose de calculer les ombres portées en projetant sur le sol les facettes composant les objets opaques de la scène. Pour les sources lumineuses directionnelles, cette projection est effectuée parallèlement à la direction de la source lumineuse. Pour les sources ponctuelles cette projection est centrale dont le centre est la source lumineuse. Nous illustrons cette technique à la figure (4.5).

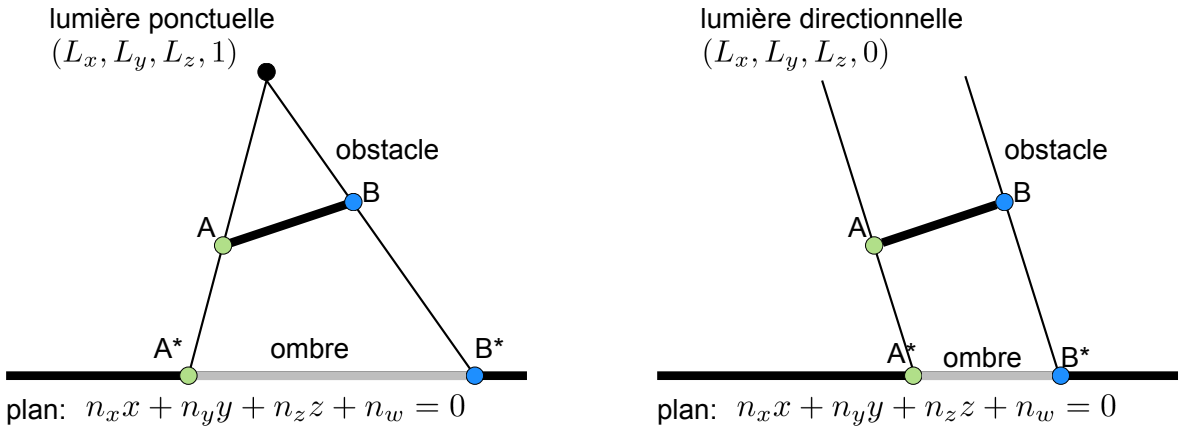


FIGURE 4.5 Illustration de la technique de projection plane des ombres. Les points projetés sont notés avec une étoile en exposant.

Si l'on note I la matrice identité, $\vec{L} = (L_x, L_y, L_z, L_w)^\top$ le vecteur de position de la lumière exprimé en coordonnées homogènes ($L_w = 1$ pour une lumière ponctuelle, ou $L_w = 0$ pour une lumière directionnelle, c'est-à-dire ponctuelle située à "l'infini") et $\vec{n} = (n_x, n_y, n_z, n_w)$ le vecteur normal du plan sur lequel s'effectue la projection (*ie* le plan d'équation $n_x x + n_y y + n_z z + n_w = 0$), la matrice de projection permettant d'obtenir l'ombre projetée d'un objet est :

$$\begin{pmatrix} n_y L_y + n_z L_z + n_w L_w & -n_y L_x & -n_z L_x & -n_w L_x \\ -n_x L_y & n_x L_x + n_z L_z + n_w L_w & -n_z L_y & -n_w L_y \\ -n_x L_z & -n_y L_z & n_x L_x + n_y L_y + n_w L_w & -n_w L_z \\ -n_x L_w & -n_y L_w & -n_z L_w & n_x L_x + n_y L_y + n_z L_z \end{pmatrix} \tag{4.1}$$

Cette technique a l'avantage d'être très simple mais est limitée aux surfaces de projection planes et ne gère pas les ombres propres.

4.2.3 Projection des ombres portées sur le sol

Dans notre cas, nous souhaitons calculer les ombres portées des objets de la scène. En effet, les ombres propres ne doivent pas être supprimées du masque de mouvement car elles sont considérées comme faisant partie de l'objet. De plus, nous disposons d'une modélisation géométrique simple de la scène constituée exclusivement de facettes planes. En particulier, nous avons modélisé le sol par un plan d'équation ($z = 0$). Dans ces conditions, nous avons choisi de calculer les ombres portées à l'aide de la méthode de projection plane des ombres.

Nous ferons également les hypothèses suivantes afin de simplifier le calcul de projec-

tion d'ombres :

1. Le Soleil est la seule source de lumière de la scène. Les autres sources lumineuses sont négligées.
2. Le Soleil est situé infiniment loin de la terre et peut donc être considéré comme une source de lumière directionnelle.
3. Seules les ombres projetées sur le sol sont considérées.
4. En accord avec le modèle de scène présenté précédemment, le sol est plan.

Sous ces hypothèses, le calcul des ombres portées revient à projeter les objets de la scène sur le sol ($z = 0$). La matrice de projection (4.1) se simplifie en :

$$\begin{pmatrix} L_z & 0 & -L_x & 0 \\ 0 & L_z & -L_y & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & L_z \end{pmatrix} \tag{4.2}$$

où (L_x, L_y, L_z) désigne la direction du Soleil vu depuis la scène. En outre, du fait de l'hypothèse de lumière directionnelle, la direction de projection ne dépend pas de la position dans la scène.

Notons que la méthode de projection est relativement générale et aurait pu être étendue à d'autres sources de lumières telles que les lampadaires (source ponctuelle), les projecteurs (source ponctuelle) ou la Lune (source directionnelle) qui elles se manifestent principalement la nuit. Nous nous sommes cependant restreint dans ces travaux au cas diurne exclusivement.

Nous expliquons dans la partie suivante, comment calculer la direction du Soleil en fonction de la position GPS de la scène, de la date et de l'heure courante, afin de pouvoir prédire la position des ombres des objets mobiles.

4.3 Direction du Soleil

L'objectif de cette partie est de présenter une méthode de calcul de la direction du Soleil pour un observateur fixe sur terre, en fonction de la date et de l'heure courante.

4.3.1 Coordonnées locales : azimut et altitude

Pour exprimer les directions d'observation des astres célestes depuis un point fixe sur terre, le système de coordonnées le plus couramment utilisé est celui des "coordonnées

horizontales" (également appelé système de "coordonnées locales"), où la direction d'une étoile (ici le Soleil) est exprimée à l'aide des angles d'azimut et d'altitude, comme montré à la figure (4.6).

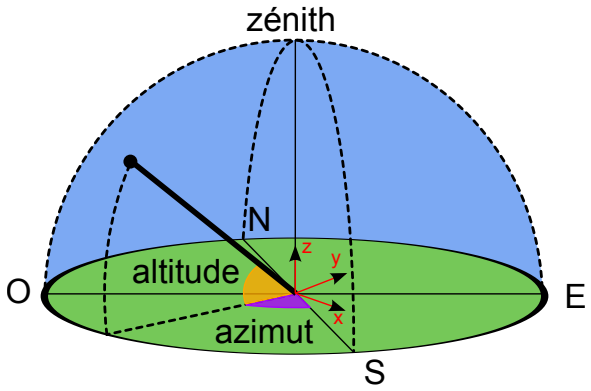


FIGURE 4.6 Système de coordonnées horizontales permettant d'exprimer la direction d'observation du Soleil. Nous avons également fait figurer le repère de la scène (en rouge) afin de faciliter la visualisation de la conversion donnée à l'équation (4.3).

Dans ce système, l'altitude est repérée par rapport au plan horizontal local d'observation et varie entre 0° (horizon) et 90° (zénith). L'azimut est, quant à lui, repéré par rapport au Sud (0°) et varie positivement vers l'Ouest(+90°), négativement vers l'Est (-90°). Notons que d'autres conventions sont également couramment utilisées pour paramétrer l'azimut, notamment en repérant la direction à partir du Nord.

Pour convertir ces angles d'azimut (Γ) et d'altitude (a) en une direction (L_x, L_y, L_z) exprimée dans le repère de la scène, on utilisera l'équation suivante :

$$\begin{aligned} L_x &= \cos(a)\cos(\Gamma_N + \pi - \Gamma) \\ L_y &= \cos(a)\sin(\Gamma_N + \pi - \Gamma) \\ L_z &= \sin(a) \end{aligned} \tag{4.3}$$

où Γ_N désigne l'angle entre l'axe des abscisses du repère de la scène et la direction du Nord.

4.3.2 Algorithmes de calcul

Les différents algorithmes de calcul de la course du Soleil se différencient selon leur précision, leur complexité et leur durée de validité. Nous proposons de présenter dans les paragraphes qui suivent les travaux les plus récents dans ce domaine et indiquerons l'algorithme retenu.

TABLE 4.1 Caractéristiques des différents algorithmes de calcul de la position du Soleil

Algorithme	Années de validité	Erreur maximale (en °)	Complexité
Reda and Andreas (2008)	-2000 – 6000	0.0003	>10000
Michalsky (1988)	1950 – 2050	0.01	530
Blanco-Muriel et al. (2001)	1995 – 2015	0.008	589
Grena (2008)	2003 – 2023	0.003	846
Grena (2012) (1)	2010 – 2110	0.2	428
Grena (2012) (2)	2010 – 2110	0.03	455
Grena (2012) (3)	2010 – 2110	0.009	572
Grena (2012) (4)	2010 – 2110	0.009	641
Grena (2012) (5)	2010 – 2110	0.003	929

L'algorithme actuel faisant référence en matière de calcul de la position du Soleil a été proposé par Reda and Andreas (2008) sur la base des algorithmes développés dans Meeus (1991). Cet algorithme, appelé SPA (*Sun Position Algorithm*), démontre une extrême précision (erreur angulaire maximale de 0.0003°) et une très large période de validité (2000 av. J.-C. – 6000). Cependant sa complexité est bien plus grande que les autres algorithmes (les développements limités utilisés font intervenir plus de 1000 coefficients) ce qui limite son utilisation aux applications nécessitant une extrême précision.

En ce qui concerne les algorithmes moins complexes, il y a eu chronologiquement celui développé par Michalsky (1988) qui est bien plus simple tout en ayant une erreur maximale de 0.01° sur la période 1950 – 2050. Plus récemment, l'algorithme (PSA) proposé par Blanco-Muriel et al. (2001) qui a amélioré la précision de calcul (erreur maximale de 0.008°) mais en se restreignant à une période de validité plus courte (1995 – 2015) et qui actuellement touche à sa fin. Pour la période actuelle, Grena (2008) a développé l'algorithme ENEA donnant une erreur maximale de 0.003° sur la période 2003 – 2023. Enfin, 5 nouveaux algorithmes ont été proposés dans Grena (2012). Ces algorithmes sont tous valides sur la période 2010 – 2110, et répondent à des compromis complexité / précision différents.

Nous avons regroupé dans la table (4.1) les différentes caractéristiques des algorithmes présentés. Dans cette table, l'erreur maximale correspond à

$$\Delta_{\max} = \sqrt{(\Delta\Gamma \cos a)^2 + \Delta a^2} \tag{4.4}$$

avec Γ et a respectivement l'azimut et l'altitude du Soleil. Les quantités $\Delta\Gamma$ et Δa représentent les écarts d'azimut et d'altitude entre l'algorithme étudié et l'algorithme de

référence (SPA), sauf pour SPA pour lequel nous rapportons l'erreur mentionnée par les auteurs.

Concernant la colonne "Complexité", nous reprenons l'évaluation de complexité effectuée par Grena (2012) où chaque classe d'opération se voit affectée d'un poids. En sommant les poids des différentes opérations constituant l'algorithme, on obtient une mesure de la complexité de l'algorithme. Les opérations d'addition, de multiplication, de conversion de type, de valeur absolue et de saut conditionnel se voient affectées d'un poids de 1. Les opérations de division de modulo et de racine carrée ont un poids de 10 ; et les opérations trigonométriques un poids de 30. Le choix de ces poids a été justifié empiriquement en comparant les temps d'exécution relatifs des différentes opérations par rapport à une simple addition (*i.e.* une division prend approximativement 10 fois plus de temps qu'une addition, et une fonction trigonométrique 30 fois plus qu'une addition).

Parmi ces algorithmes, nous avons choisi d'utiliser l'algorithme (Grena, 2012) (3) car il présente un bon compromis entre précision / complexité et restera valide pour encore une centaine d'années. La procédure de calcul (algorithme 1) prend en entrée la date, l'heure courante et la position GPS d'observation (latitude et longitude) et renvoie la direction du Soleil.

Algorithme 1 : Calcul de la direction du Soleil

Input : Année (y), mois (mo), jour (d), heure (h), minute (mi), seconde(s), coordonnées GPS (lat,lon) d'observation et direction du Nord (Γ_N)

Output : Direction du Soleil (L_x, L_y, L_z)

Function `ComputeSunDirection(`y,mo,d,h,mi,s,lat,lon,Γ_N`)`

`/* compute times` t `(UT) and` t_e `(independant of earth rotation) */`

$t \leftarrow$ `ComputeJD(`y,mo,d,h,mi,s`)` $-$ `ComputeJD(`*2060,1,1,0,0,0*`)`;

$t_e \leftarrow t + 1.574 \times 10^{-5} \times 67$;

`/* compute apparent ecliptic longitude (`λ`) and obliquity (`ϵ`) */`

$\lambda \leftarrow -1.388\,803 + 1.720\,279\,216 \times 10^{-2} \times t_e$
$+3.3366 \times 10^{-2} \times \sin(0.017\,201\,971\,5 \times t_e - 0.061\,72)$
$+3.53 \times 10^{-4} \times \sin(2 \times 0.017\,201\,971\,5 \times t_e - 0.1163)$;

$\epsilon \leftarrow 4.089\,567 \times 10^{-1} - 6.19 \times 10^{-9} \times t_e$;

`/* compute declination (d) and right ascension (ra) */`

$d \leftarrow \arcsin(\sin\lambda \sin\epsilon)$;

$ra \leftarrow \arctan(\frac{\sin\lambda \cos\epsilon}{\cos\lambda})$;

if $ra < 0$ **then**
  $ra \leftarrow ra + 2\pi$

`/* compute hour angle (H) */`

$H \leftarrow 1.752\,831\,1 + 6.300\,388\,099t + lon - ra$;

$H \leftarrow ((H + \pi) \bmod 2\pi) - \pi$;

`/* compute azimuth angle (`Γ`) */`

$\Gamma \leftarrow \arctan(\frac{\sin H}{\cos H \sin lat - \tan d \cos lat})$;

`/* compute parallax/refraction corrected altitude angle (`a`) */`

$sa_0 \leftarrow \sin lat \sin d + \cos lat \cos d \cos H$;

$ca_0 \leftarrow \sqrt{1 - sa_0^2}$;

$a \leftarrow \arcsin(sa_0) - 4.26 \times 10^{-5} ca_0$;

if $a > 0$ **then**
  $a \leftarrow a + \frac{2.8262 \times 10^{-4}}{\tan(a + \frac{0.003\,138}{a + 0.089\,19})}$

`/* compute sun direction */`

$L_x \leftarrow \cos a \cos(\Gamma_N + \pi - \Gamma)$;

$L_y \leftarrow \cos a \sin(\Gamma_N + \pi - \Gamma)$;

$L_z \leftarrow \sin a$;

return (L_x, L_y, L_z);

Function `ComputeJD(`y, mo, d, h, mi, s`)`

if $mo \leq 2$ **then**
  $mo \leftarrow mo + 12$;
  $y \leftarrow y - 1$;

$jd \leftarrow$ `INT(`*365.25(y - 2000)*`)` + `INT(`*30.6001(mo+1)*`)` + `INT(`*0.01y*`)`
$+d - 21958 + \frac{h}{24} + \frac{mi}{1440} + \frac{s}{86400}$;

return jd;

4.4 Intégration de la prédiction des ombres au modèle de scène et de caméra

Afin d'illustrer la manière dont nous avons intégré la prédiction des ombres au modèle de scène et de caméra, nous donnons la procédure simplifiée[1] de calcul du masque d'ombres projetées dans ce qui suit (algorithme 2).

Algorithme 2 : Procédure de rendu du masque d'ombres portées.

Input : Caméra ($\mathcal{C}$), Liste d'objets opaques ($\mathcal{O}$), Vecteur de direction du Soleil (L)

Output : Image des ombres projetées

Function `ComputeShadowMask`

- fill color buffer with black;
- `/* Build camera matrices according equations (3.4) and (3.6) */`
- $K \leftarrow$ `ComputeCameraIntrinsicMatrix`$(\mathcal{C})$;
- $M \leftarrow$ `ComputeCameraExtrinsicMatrix`$(\mathcal{C})$;
- `/* Build sun projection matrix according to equation (4.2) */`
- $S \leftarrow$ `ComputeShadowProjectionMatrix`(L);
- **foreach** *object* o *in* $\mathcal{O}$ **do**
  - $M_o \leftarrow$ `GetModelMatrix`(o);
  - $Tri_o \leftarrow$ `GetModelTriangles`(o);
  - `/* Compute projection matrix */`
  - $P \leftarrow K \times [I_3|0_3] \times M \times S \times M_o$;
  - **foreach** *triangle* tri *in* Tri_o **do**
    - `/* Project triangle vertices` (tri_1, tri_2, tri_3) `*/`
    - $v_1 \leftarrow P \times tri_1$;
    - $v_2 \leftarrow P \times tri_2$;
    - $v_3 \leftarrow P \times tri_3$;
    - `DrawWhiteTriangle` (v_1, v_2, v_3);

1. l'implémentation réelle gère en plus les subtilités liées à OpenGL et la distorsion de la caméra qui ne sont pas détaillées ici

(a) 18/07/2012 13h58m56s (b) 18/07/2012 16h19m56s

(c) 18/07/2012 18h21m29s (d) 19/07/2012 10h46m31s

FIGURE 4.7 Exemple de prédiction d'ombres (Foxstream) sur deux demi-journées. Les contours des bâtiments sont représentés en vert, et ceux de leurs ombres sont en rouge.

4.5 Validation

Dans cette partie nous allons valider expérimentalement notre approche de prédiction des ombres dans le cas d'objets fixes dont la position est connue *a priori* : les bâtiments.

Nous utilisons l'algorithme de rendu présenté précédemment (algorithme 2) et comparons les masques obtenus avec des images réelles.

Le premier exemple (figure 4.7) montre une série d'acquisitions réalisées entre le 18/07/2012 14h et le 19/07/2012 11h sur le parking bordant les locaux de la société Foxstream (GPS : N45.7539326 E4.925457). Dans cette séquence, nous constatons que les contours des bâtiments ainsi que leurs ombres sont globalement correctement prédits.

Nous avons également expérimenté cette prédiction des ombres dans un autre contexte (figure 4.8) où les données OpenStreetMap n'étaient pas disponibles et ont donc été ren-

seignées manuellement. Dans cet exemple, nous avons évalué la prédiction des ombres sur une plus grande durée (environ 8 mois). La scène observée se situe aux abords du barrage de Génissiat (GPS : N46.0524321 E5.814168). Les résultats obtenus confirment le bon comportement de la prédiction des ombres. À noter que, l'hypothèse de sol plan d'équation $z = 0$ a du être assouplie afin d'accommoder les changements de hauteur du niveau de l'eau au cours des mois : le barrage accumulant puis relâchant de l'eau en fonction des besoins (nous avons estimé des variations de hauteur d'eau allant jusqu'à 3m).

Outre ces illustrations qualitatives de prédiction des ombres, nous avons quantifié numériquement la qualité de prédiction pixel à pixel vis-à-vis de la réalité (voir table 4.2). Nous avons employé les métriques de recouvrement (CR pour *Coverage Ratio*), de Précision (P), de Rappel (R) et le F-score définis respectivement de la manière suivante :

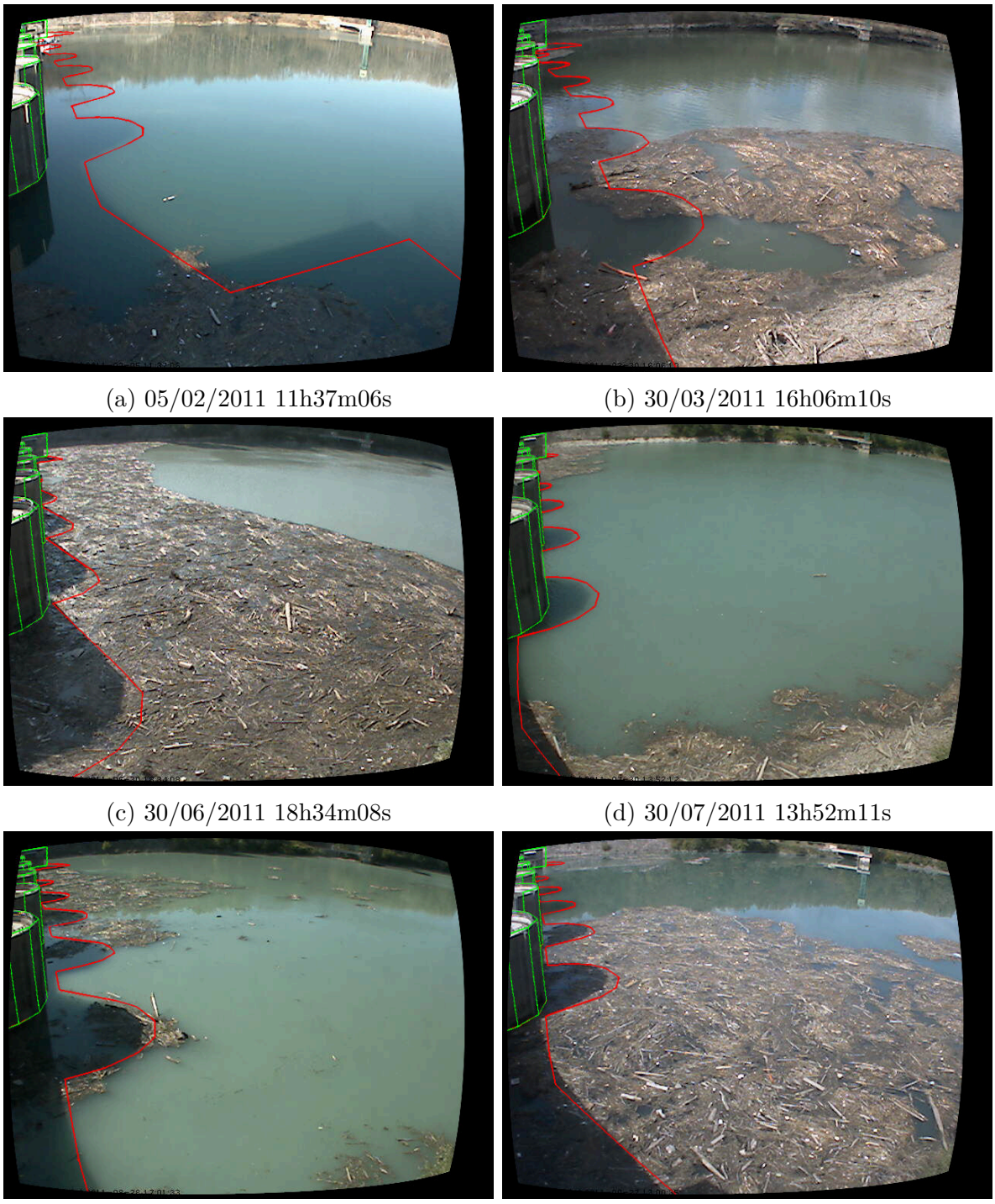
$$\begin{aligned} \mathrm{P} &= \frac{|\mathcal{P} \cap \mathcal{T}|}{|\mathcal{P}|} & \mathrm{R} &= \frac{|\mathcal{P} \cap \mathcal{T}|}{|\mathcal{T}|} \\ \mathrm{CR} &= \frac{|\mathcal{P} \cap \mathcal{T}|}{|\mathcal{P} \cup \mathcal{T}|} & \text{F-score} &= 2\frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}} \end{aligned} \tag{4.5}$$

où $\mathcal{P}$ désigne l'ensemble des pixels prédits comme étant de l'ombre, $\mathcal{T}$ l'ensemble des pixels étant de l'ombre (vérité terrain), et $|\mathcal{X}|$ désignant le cardinal de l'ensemble $\mathcal{X}$.

TABLE 4.2 Évaluation pixel à pixel de la qualité de la prédiction des ombres. Nous avons annoté manuellement les ombres d'une dizaine d'images afin de constituer la vérité terrain.

séquence	CR	P	R	F-score
Foxstream	85.3%	95.7%	88.4%	91.8%
Génissiat	82.2%	89.3%	91.4%	90.0%

Ces résultats quantitatifs confirment les observations faites sur les images superposant la prédiction des ombres et la réalité, et prouvent la faisabilité de la prédiction des ombres observées par une caméra fixe. Notons que les objets que nous avons utilisés pour valider cette approche sont fixes (bâtiments). Aussi nous avons souhaité étendre cette prédiction aux objets mobiles de la scène afin de pouvoir supprimer leur ombre du masque de mouvement. Nous présenterons dans la section suivante une première approche réalisant cette suppression pour le cas des piétons, qui est un cas très sensible en vidéosurveillance.



(a) 05/02/2011 11h37m06s (b) 30/03/2011 16h06m10s

(c) 30/06/2011 18h34m08s (d) 30/07/2011 13h52m11s

(e) 28/08/2011 17h01m33s (f) 27/09/2011 13h00m35s

FIGURE 4.8 Exemple de prédiction d'ombres (barrage de Génissiat) sur une durée de 8 mois. Les contours des bâtiments sont représentés en vert, et ceux de leurs ombres sont en rouge.

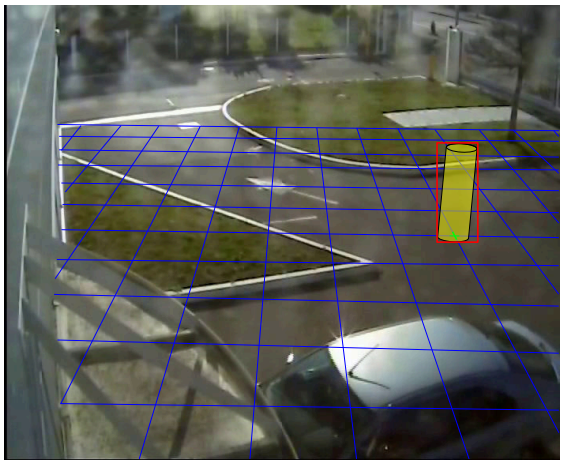


FIGURE 4.9 Illustration de la modélisation utilisée pour la suppression de l'ombre des piétons. Sur cette vue ont été superposées une image obtenue par la caméra, la grille de positions discrétisées (en bleu), une telle position (croix verte) ainsi que la boîte englobante 2d (en rouge) qu'aurait un piéton (en jaune) à cette position. Pour des raisons de lisibilité, les pas de discrétisation de la grille ont été choisis bien espacés. En pratique, ce pas est choisi beaucoup plus fin (de l'ordre du rayon du cylindre modélisant un piéton)

4.6 Une première méthode de suppression d'ombres pour les piétons

Dans ce qui suit, nous allons présenter une méthode de suppression des ombres des piétons du masque de mouvement. Dans ce contexte, et dans un souci de simplicité, nous modélisons les piétons par un cylindre de dimensions fixes (diamètre : 60cm ; hauteur : 1,75m) posé sur le sol. Cette modélisation cylindrique permet de ne pas se préoccuper de l'orientation des piétons.

La zone du sol surveillée par la caméra est discrétisée à l'aide d'une grille de positions $\mathcal{P}_{\delta_x,\delta_y}$ où δ_x et δ_y représentent les pas de discrétisation (grille en bleu sur la figure (4.9)). Nous appelons f l'application qui, à chaque position p de $\mathcal{P}_{\delta_x,\delta_y}$, associe la boîte englobante 2d notée b sur l'image caméra correspondant à un piéton placé en p. Une telle association $b = f(p)$ est illustrée à la figure (4.9) où la position p est représentée par la croix verte, le piéton par le cylindre jaune et la boîte englobante b correspondante en rouge. Nous notons $\mathcal{B} = f(\mathcal{P}_{\delta_x,\delta_y})$ l'ensemble des boîtes englobantes 2d générées à partir de la grille de discrétisation.

Le masque de mouvement est noté F et est défini tel que $F(q) = 1$ si le pixel q appartient à l'avant-plan, et $F(q) = 0$ sinon. Nous notons W_F la fonction associant à

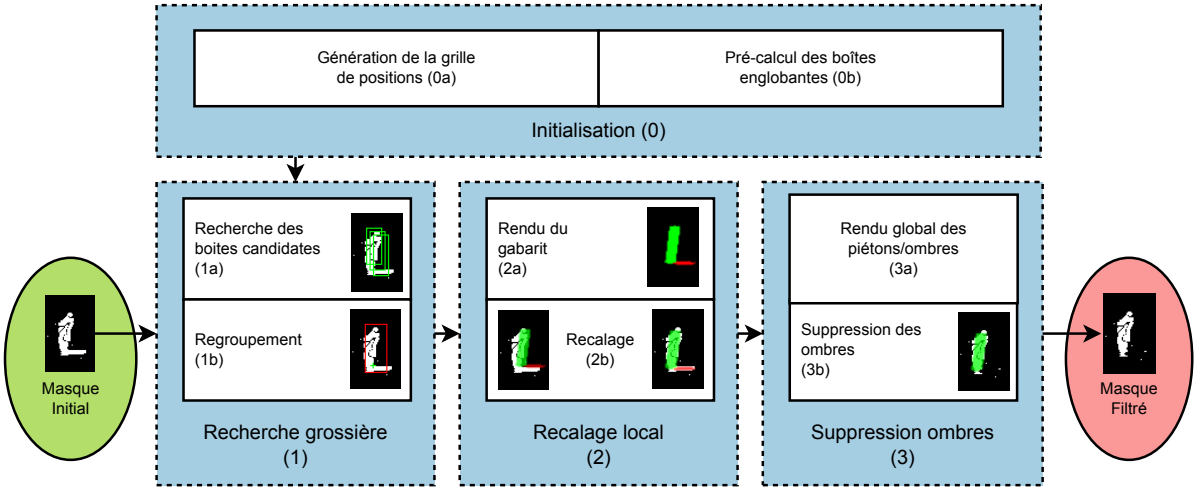


FIGURE 4.10 Aperçu des étapes de l'algorithme de suppression des ombres des piétons.

une boîte englobante 2d b la quantité suivante :

$$W_F(b) = \begin{cases} \frac{\sum_{q \in b} F(q)}{A(b)} & \text{si } A(b) > 0 \\ 0 & sinon \end{cases} \tag{4.6}$$

avec $A(b)$ l'aire de la boîte englobante 2d b. La quantité $W_F(b)$ représente le pourcentage de la boîte b contenant des pixels "actifs" (*i.e.* de valeur égale à 1) du masque de mouvement F. Nous appellerons cette quantité "score d'intersection" entre la boîte b et le masque de mouvement F.

Nous donnons un aperçu des étapes composant notre algorithme de suppression des ombres des piétons du masque de mouvement à la figure (4.10). Le principe de l'algorithme est d'estimer la position de tous les piétons de la scène (étapes 1 et 2), afin de pouvoir calculer (étape 3a) puis supprimer (étape 3b) leurs ombres projetées. Nous présenterons et illustrerons ces différentes étapes dans les parties suivantes.

4.6.1 Initialisation

Durant la phase d'initialisation, la grille de positions discrètes visibles par la caméra $\mathcal{P}_{\delta_x,\delta_y}$ (étape 0a) est générée. En chacune de ces positions, nous calculons la boîte englobante 2d correspondant à un piéton en cette position (étape 0b). Nous obtenons ainsi un ensemble de couples $\{(p, b = f(p))\}$ permettant de retrouver la position au sol p d'un piéton à partir de sa boîte englobante b. Les positions menant à une boîte englobante vide, par exemple celles pour lesquelles le piéton serait caché par un mur, sont retirées.

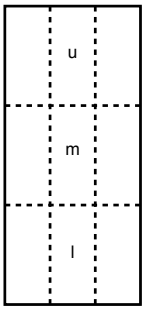


FIGURE 4.11 Découpage en 9 tuiles d'une boîte englobante 2d. Les tuiles supérieure (u), centrale (m) et inférieure (l) de la bande centrale sont testées afin de déterminer si un piéton est présent à cet endroit.

On voit ici un intérêt d'utiliser les informations d'OpenStreetMap afin de limiter l'espace de recherche aux positions visibles par la caméra.

Le choix des pas de discrétisation δ_x et δ_y est fait de manière à couvrir correctement la zone à surveiller, tout en évitant les calculs superflus. En pratique, nous adoptons les valeurs suivantes $\delta_x = \delta_y = 0.3$m, qui correspondent à la moitié de la taille au sol de l'objet à détecter.

Cette étape d'initialisation ne dépend pas du masque de mouvement à traiter et peut donc être effectuée hors-ligne, avant tout traitement.

4.6.2 Recherche grossière

Le but de cette étape est d'estimer rapidement le nombre K de piétons présents dans le masque de mouvement F et leurs positions au sol $C = \{c_k\}_{k=1..K}$.

Premièrement nous sélectionnons, parmi les boîtes précalculées $\mathcal{B}$, le sous-ensemble $\mathcal{B}'$ des boîtes précalculées qui contiennent un piéton (étape 1a). Les éléments b'_i de $\mathcal{B}'$ sont choisis de sorte que :

$$W_F(u_i) > \tau \quad W_F(m_i) > \tau \quad W_F(l_i) > \tau \tag{4.7}$$

avec u_i, m_i et l_i les rectangles du tiers supérieur, du tiers central et du tiers inférieur de la bande verticale centrale de la boîte englobante 2d b'_i (voir figure 4.11) ; et τ un seuil (ici $\tau = 0.3$). Nous justifions qualitativement ces équations à l'aide des éléments suivants : l'utilisation de boîtes rectangulaires alignées avec les axes de l'image se prêtent à des détections accélérées grâce aux images intégrales (Viola and Jones, 2004) ; les boîtes englobantes précalculées sont plus grandes que les objets qu'elles contiennent, autrement dit, les silhouettes des piétons ne remplissent pas toute la surface de la boîte englobante

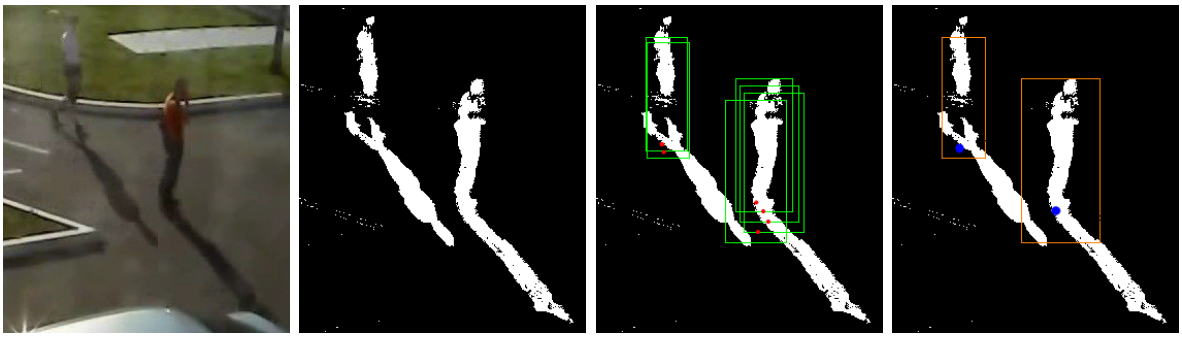


Image RGB | Masque de mouvement | Avant regroupement | Après regroupement

FIGURE 4.12 Illustration de la procédure de meanshift. Les positions au sol (points) sont regroupées à l'aide de l'algorithme de meanshift. Chaque regroupement représente idéalement un unique piéton. Les boîtes englobantes 2d relevant d'un même piéton sont alors fusionnées (union) afin de constituer le rectangle délimitant ce piéton.

2d mais se situent principalement sur la bande centrale verticale de cette boîte englobante (d'où le découpage vertical en tiers). Par ailleurs, nous souhaitons ne conserver que les boîtes dont la silhouette de piéton couvre toute la hauteur de la boîte englobante (d'où le découpage horizontal en tiers).

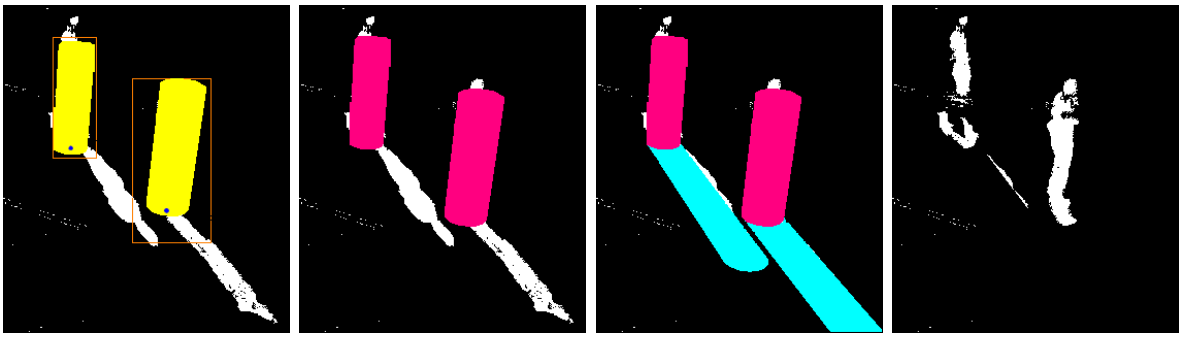
À l'issue de cette sélection, plusieurs boîtes englobantes sélectionnées peuvent en fait correspondre à un même et unique piéton (voir figure 4.12). Nous proposons donc de regrouper les boîtes b'_i sélectionnées à l'aide d'une procédure de *meanshift* (étape 1b) en raisonnant sur les positions au sol p'_i associées obtenues grâce aux correspondances établies durant la phase d'initialisation, et en leur affectant le poids suivant favorisant les boîtes les plus remplies :

$$\frac{1}{3}(W_F(u_i) + W_F(m_i) + W_F(l_i)) \tag{4.8}$$

Cette procédure permet ainsi d'estimer conjointement le nombre K de piétons présents ainsi que leurs positions en regroupant les positions au sol proches.

4.6.3 Recalage local et suppression

Les estimations des positions faites à l'étape précédente peuvent mener à des rendus de piétons décalés de quelques pixels par rapport au masque de mouvement (4.13). En effet, la nature discrète de la grille de position $\mathcal{P}_{\delta_x,\delta_y}$ ainsi que la procédure de regroupement par meanshift favorisent l'apparition de ces écarts. Nous proposons donc de recaler localement chacune des positions estimées en utilisant le masque de mouvement comme guide. Nous procédons par *template matching* en effectuant un rendu d'un gabarit de



Rendu d'un gabarit | Recalage du gabarit | Prédiction des ombres | Ombres supprimées

FIGURE 4.13 Illustration de la procédure de recalage local et de suppression des ombres

piéton (étape 2a) et en le déplaçant dans un voisinage de la position estimée (étape 2b). Une fois les positions corrigées, il suffit de placer des gabarits de piétons à ces positions, et d'effectuer un rendu des ombres résultantes (étape 3a). Si l'on note S le masque d'ombre ainsi généré, le masque de mouvement F est alors nettoyé (étape 3b) de ses ombres par opérations logiques :

$$F = F \wedge \neg S \tag{4.9}$$

Nous illustrons ces dernières étapes à la figure (4.13).

4.7 Évaluation

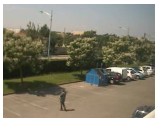
4.7.1 Séquences d'évaluation

Nous avons sélectionné plusieurs séquences vidéo afin de tester l'approche de suppression d'ombres. Les caractéristiques de ces séquences ont été répertoriées dans la table (4.3). À noter qu'à part une des séquences qui a été capturée par nos soins, les trois autres font parties de challenges internationaux (Ferryman and Shahrokni, 2009; Vacavant et al., 2013).

Sur ces séquences, plusieurs images ont été annotées manuellement afin d'obtenir des masques de vérité pour les ombres et l'avant-plan. La seule exception concernant ces annotations porte sur la séquence PETSS09.S2L2 où l'annotation des ombres était impossible, car les ombres des piétons tombent dans des régions déjà ombrées par d'autres obstacles de la scène. C'est pourquoi dans cette séquence, seule l'annotation concernant les objets de premier-plan a été faite.

Les modèles de scène utilisés ont été générés à partir des données OpenStreetMap

TABLE 4.3 Caractéristiques des séquences de test. Les données manquantes (date et heure) ont été estimées à partir des ombres visibles et sont notées en italique.

	BMC.V01	**SudEst**	**PETS01.D3**	**PETS09.S2L2**
				
séquence				
résolution	320 × 240	704 × 576	768 × 576	768 × 576
durée	21m58s	40s	3m33s	1m02s
#images annotées	40	50	50	10
scène				
localisation	N45.75381° E4.92536°	N45.75380° E4.92524°	N51.43823° W0.94443°	N51.43824° W0.94440°
heure GMT	*07-01 10 :00*	04-17 07 :17	*03-07* 14 :50	*02-11* 14 :55

autour des positions GPS spécifiées. En ce qui concerne les paramètres de caméra, quand une calibration était disponible (séquence PETS09.S2L2 par exemple), celle-ci a été utilisée. Dans le cas contraire, une calibration approchée a été réalisée à l'aide du logiciel de simulation et de placement des caméras. Pour la date et l'heure, nous avons souvent dû les estimer à partir des ombres de la scène car ces données n'étaient pas renseignées. Seule la séquence capturée par nos soins (séquence SudEst) a pu être datée lors de son enregistrement.

Les masques de mouvement ont été obtenus à l'aide du modèle de fond ViBe (Barnich and Van Droogenbroeck, 2011).

4.7.2 Métriques d'évaluation

Les métriques que nous utilisons servent à évaluer deux éléments : d'une part, la qualité de la détection des ombres et, d'autre part, l'amélioration de la segmentation des objets d'avant-plan une fois les ombres supprimées.

Concernant l'évaluation de la qualité de la segmentation des ombres présentes dans le masque de mouvement nous utilisons les métriques définies par Prati et al. (2003) de

la manière suivante :

$$\eta = \frac{\mathrm{TP}_S}{\mathrm{TP}_S + \mathrm{FN}_S} \tag{4.10}$$

$$\xi = \frac{\overline{\mathrm{TP}_F}}{\mathrm{TP}_F + \mathrm{FN}_F} \tag{4.11}$$

η est appelé "taux de détection d'ombre" (ou *shadow detection rate*) et quantifie la part d'ombre détectée par rapport à la part d'ombre totale. TP_S est le nombre de pixels d'ombre correctement classés comme ombre (*i.e.* vrais positifs) et FN_S est le nombre de pixels d'ombre classés à tort comme n'étant pas de l'ombre (*i.e.* faux négatifs).

ξ est appelé "taux de discrimination d'ombre" (ou *shadow discrimination rate*) et quantifie la part de pixels d'avant-plan correctement détectés en pénalisant explicitement les erreurs de classification d'ombre qui modifieraient les détections et les formes des objets d'avant-plan. En effet, une erreur de classification d'ombre pour un pixel du fond n'affecte pas la segmentation des objets du premier plan alors qu'une erreur de classification d'ombre sur un pixel du premier plan affecte la segmentation finale des objets. Dans cette équation, $\overline{\mathrm{TP}_F}$ désigne le nombre de pixels du premier plan correctement classés comme premier plan moins le nombre de pixels de premier plan classés comme ombre. TP_F et FN_F désignent respectivement le nombre de pixels du premier plan correctement classés comme premier plan (vrais positifs), et le nombre de pixels du premier plan incorrectement classés comme ne faisant pas partie du premier plan (faux négatifs).

En ce qui concerne l'évaluation de la segmentation des objets d'avant-plan, nous utilisons les métriques classiques de "précision" (P), "rappel" (R) et de "F-Score" définies ainsi :

$$\mathrm{P} = \frac{\mathrm{TP}_F}{\mathrm{TP}_F + \mathrm{FP}_F} \tag{4.12}$$

$$\mathrm{R} = \frac{\mathrm{TP}_F}{\mathrm{TP}_F + \mathrm{FN}_F} \tag{4.13}$$

$$\text{F-score} = 2\frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}} \tag{4.14}$$

où TP_F désigne le nombre de pixels de l'avant-plan correctement classé comme tel (vrais positifs), FN_F le nombre de pixels de l'avant-plan classés à tort comme faisant partie du fond (faux négatifs) et FP_F le nombre de pixels du fond incorrectement classés comme avant-plan (faux positifs).

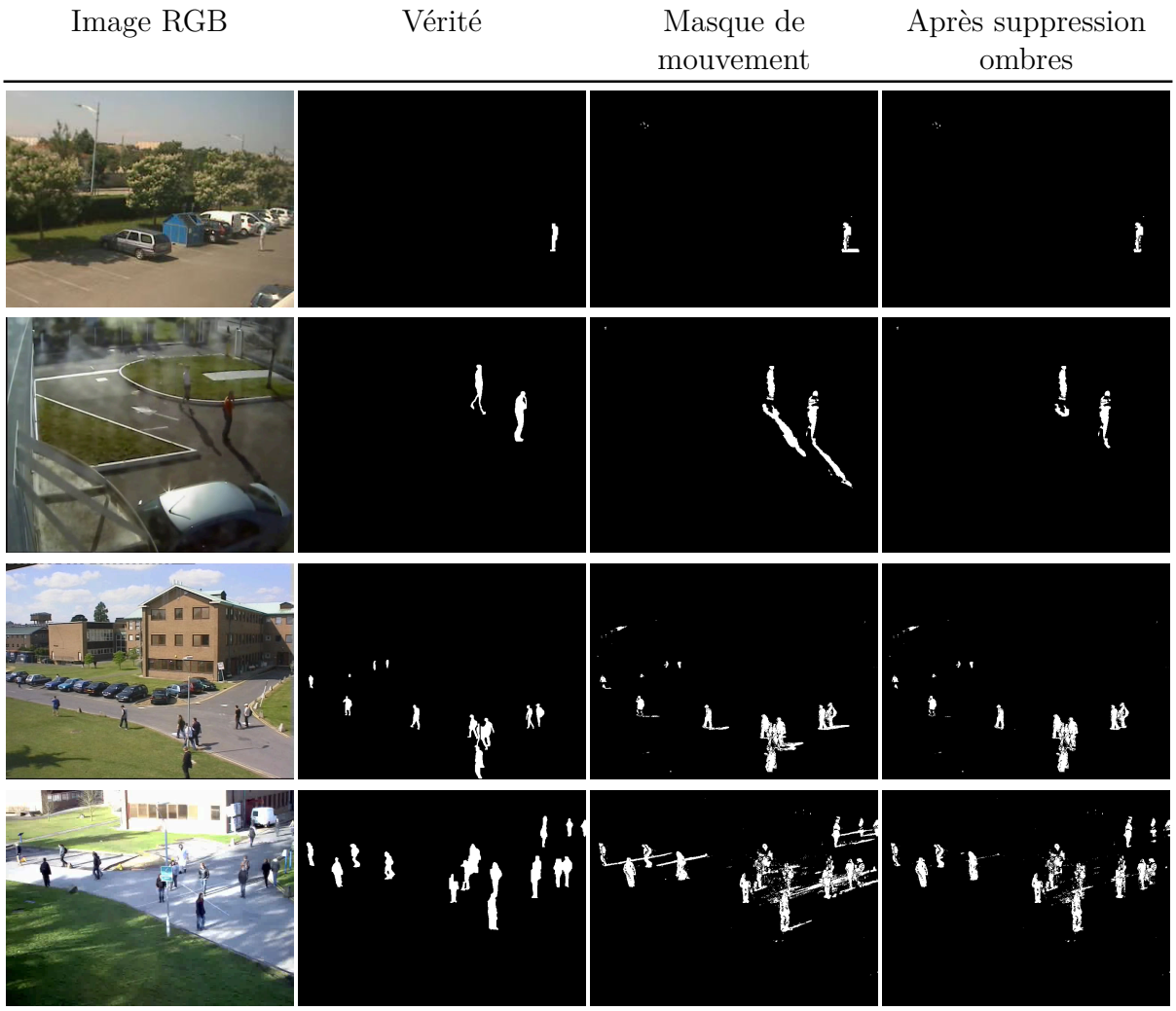


FIGURE 4.14 Exemples de résultats de suppression d'ombres pour chacune des séquences.

4.7.3 Résultats

Nous donnons pour chacune des séquences quelques images représentatives de la suppression d'ombre obtenue par notre méthode (figure 4.14).

Les résultats quantitatifs associés sont donnés, en terme de segmentation d'ombre à la figure (4.15), et en terme d'amélioration de la segmentation des objets à la figure (4.16).

Sur ces graphiques, nous avons reporté les résultats obtenus par notre méthode ("Rogez"), et deux autres méthodes courantes de suppression d'ombre ("Cucchiara" (Cucchiara et al., 2003) et "Sanin" (Sanin et al., 2012)). Pour rappel, la méthode de "Cucchiara" est une méthode de suppression d'ombre utilisant principalement un critère de

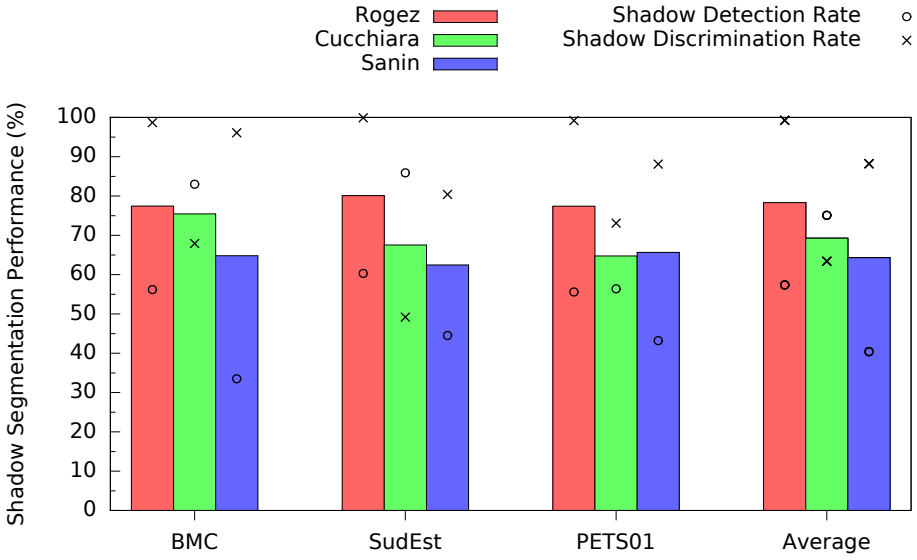


FIGURE 4.15 Performances de segmentation des ombres. La hauteur des barres correspond à la moyenne du taux de détection et de discrimination des ombres.

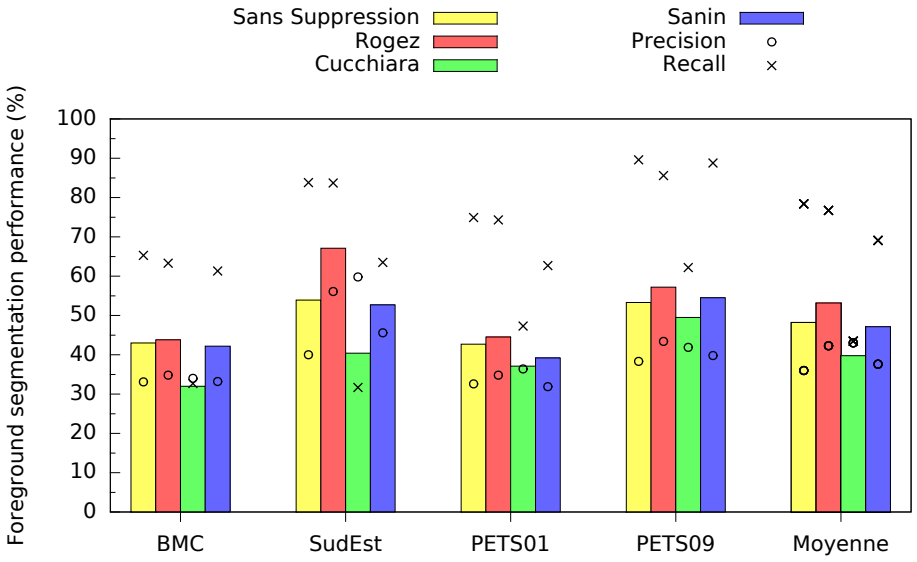


FIGURE 4.16 Amélioration de la qualité de la segmentation après suppression des ombres. La hauteur des barres représente le F-Score.

couleur pour détecter l'ombre ; tandis que la méthode de "Sanin" inclut également un critère portant sur la texture de la surface à analyser. Aucune de ces approches n'impose de contraintes géométriques.

Concernant la segmentation des ombres (figure 4.15), nous constatons que le comportement de notre algorithme est relativement indépendant de la séquence vidéo considéré : le taux de détection des ombres est de l'ordre de 60% en moyenne, le taux de discrimination de l'ordre de 99%. Par ailleurs, notre algorithme se montre globalement plus performant que les autres méthodes génériques de suppression d'ombre.

Ces bonnes performances en terme de détection et suppression d'ombre se traduisent en terme d'amélioration de la segmentation des objets mobiles (voir figure 4.16). Dans ce graphique, nous observons une certaine amélioration de la qualité de la segmentation : le F-score après suppression d'ombre est plus grand que sans suppression de l'ombre. On notera également que cette amélioration dépend largement de la taille des ombres sur la séquence considérée : Sur la séquence "BMC", les ombres sont petites et leur suppression n'améliore donc que légèrement la segmentation finale ; en revanche, sur la séquence "SudEst", les ombres sont aussi grandes que les objets qui les projettent et leur suppression induit une nette amélioration. Ce graphique confirme également que la méthode proposée se montre plus performante que les autres méthodes de suppression d'ombre.

4.7.4 Sensibilité aux paramètres de la caméra

La méthode de suppression d'ombre que nous proposons repose sur l'utilisation d'une caméra calibrée. Aussi nous avons souhaité évaluer l'influence de variations des paramètres de calibration sur les performances de notre méthode. À noter que pour faciliter cette étude, nous avons restreint la scène à une scène sans obstacles fixes dans le champ de vision de la caméra (pas de bâtiments).

Nous avons procédé en changeant la valeur d'un paramètre à la fois, les autres étant maintenus à leur valeur de référence, et nous avons mesuré l'évolution des métriques de performance.

Plus formellement nous définissons la quantité suivante :

$$\Delta(Y) = \max\{|Y(X+\epsilon) - Y(X)|, |Y(X-\epsilon) - Y(X)|\} \tag{4.15}$$

avec $\epsilon > 0$ l'incertitude sur le paramètre X, et Y la métrique de performance étudiée.

Les résultats de cette étude de sensibilité sont donnés dans la table (4.4).

Nous avons également classé les différents paramètres en fonction de leur influence

TABLE 4.4 Influence des paramètres de la caméra sur les métriques de performance. Se référer aux parties (3.1 et 3.3.1) pour les notations.

Paramètre	Incertitude	ΔP	ΔR	ΔF-score	$\Delta\eta$	$\Delta\xi$	Rang
position (x)	±10m	0.005	0.001	0.004	0.019	0.001	11
position (y)	±10m	0.000	0.000	0.000	0.000	0.000	12
hauteur (z)	±2m	0.167	0.008	0.133	0.595	0.010	2
pan (ψ)	±10°	0.079	0.003	0.059	0.244	0.006	6
tilt (θ)	±10°	0.147	0.011	0.115	0.516	0.013	3
focale (f)	±2mm	0.124	0.011	0.091	0.409	0.013	4
taille capteur (l_x)	±2mm	0.048	0.020	0.039	0.104	0.024	5
taille capteur (l_y)	±2mm	0.167	0.014	0.132	0.619	0.016	1
k_1	±0.05	0.017	0.007	0.013	0.046	0.007	7
k_2	±0.05	0.022	0.001	0.013	0.055	0.002	10
p_1	±0.05	0.016	0.005	0.009	0.060	0.006	9
p_2	±0.05	0.021	0.005	0.014	0.049	0.006	8

sur les métriques de performance. Ainsi les paramètres influençant le plus les résultats sont ceux qui impactent directement la hauteur apparente des piétons sur l'image, à savoir la taille verticale du capteur de la caméra, la hauteur à laquelle elle est placée, l'angle de tilt et la focale de la caméra. C'est pourquoi, nous préconisons de définir ces paramètres avec soin. Les autres paramètres ont moins d'influence sur les résultats et servent principalement à améliorer la cohérence de la vue par rapport au reste de la scène.

4.8 Conclusion

Dans ce chapitre nous avons présenté une méthode de prédiction des ombres projetées dues au Soleil au cours de la journée. Cette méthode a été validée sur le cas d'objets fixes (bâtiments) en démontrant des performances prometteuses (F-Score de l'ordre de 90%) sur les séquences considérées.

Du fait de ces résultats, nous avons proposé d'étendre cette méthode au cas d'objets mobiles, plus particulièrement au cas des piétons, qui constituent une cible majeure pour la vidéosurveillance. Ainsi nous avons proposé une méthode permettant de détecter et filtrer les ombres des piétons visibles dans le masque de mouvement. Cette méthode améliore la qualité de la segmentation faite par le modèle de fond et démontre des performances supérieures aux autres méthodes classiques de suppression d'ombres sur les séquences considérées.

En revanche, comme nous avons travaillé avec un modèle d'objet simple (cylindre de taille fixe), cette méthode ne peut s'appliquer en l'état qu'à une seule catégorie d'objets (les piétons). Nous proposons de lever cette contrainte en adoptant une modélisation non pas cylindrique des objets, mais à base de pavés droits (aussi appelés cuboïdes) de dimensions variables. Ce changement de modèle, plus générique mais aussi plus complexe, implique potentiellement une plus grande sensibilité aux erreurs et un coût d'exécution plus grand. Aussi, il n'est pas envisageable d'opérer ce changement sans introduire un mécanisme facilitant ce processus. C'est pourquoi nous proposons d'intégrer cette estimation des positions et dimensions des piétons dans une démarche de suivi des objets. Ainsi nous pourrons bénéficier de l'apport de la cohérence temporelle issue du suivi et réduire l'espace d'estimation des paramètres.

Chapitre 5

Suivi multi-objets

Le suivi multi-objets permet d'obtenir les trajectoires des objets mobiles de la scène vue par une caméra. L'objet de ce chapitre est d'exposer la méthode de suivi multi-objets que nous avons développée au cours de cette thèse.

Cette méthode s'appuie sur les travaux de Di Lascio et al. (2013) qui formulent le problème de suivi multi-objets en un problème récursif d'associations entre détections et objets suivis. La particularité de la méthode qu'ils proposent est, premièrement, de prendre explicitement en compte les défauts qui peuvent survenir lors de la phase de détection, en particulier les défauts courants liés à l'utilisation d'un masque de mouvement (fusion, morcellement, non détection). Deuxièmement, leur algorithme gère les occultations inter-objets via l'utilisation de groupes qui permettent notamment de suivre collectivement les objets occultants et occultés. Troisièmement, ils modélisent l'évolution d'un objet par un automate fini permettant ainsi de synthétiser de manière compacte et intelligible l'état courant d'un objet ainsi que les transitions possibles pour celui-ci. Cette modélisation permet en outre d'adapter le comportement de l'algorithme en fonction de l'état de chaque objet. Pour ces raisons, ainsi que pour les performances annoncées par les auteurs, nous avons choisi de baser nos travaux sur cet algorithme.

Toutefois, l'algorithme proposé par Di Lascio et al. (2013) est spécifique à une catégorie d'objets (piétons, valises) et nécessite un apprentissage spécifique pour chacune des classes d'objets à suivre. C'est pourquoi nous avons souhaité généraliser cette approche à tout objet se déplaçant au sol, cas le plus probable dans le cadre de la vidéosurveillance. Nous avons également intégré les modèles de scène et de caméra développés précédemment afin de raisonner en 3d et ainsi être capable d'estimer la position et les dimensions 3d des objets suivis. Grâce à cette intégration de la 3d nous avons également amélioré la gestion de la formation et de la destruction des groupes.

Nous présentons dans ce qui suit la méthode développée et évaluerons l'apport de

nos contributions par rapport à l'algorithme original.

5.1 Introduction

Le suivi d'objets consiste, dans notre cas, à extraire les trajectoires des objets mobiles d'une scène observée par une caméra. Bien que de formulation simple, ce problème d'estimation des trajectoires n'en reste pas moins un problème complexe et n'est d'ailleurs pas encore complètement résolu comme en témoignent les nombreuses publications à ce sujet (voir bibliographie au chapitre 2).

La méthode que nous proposons réalise l'estimation "en ligne" des trajectoires des objets mobiles, c'est-à-dire que la trajectoire des objets est construite de proche en proche, au fur et à mesure que de nouvelles images sont acquises par la caméra. Dans ce contexte, l'estimation des trajectoires consiste alors à détecter et ré-identifier dans chaque nouvelle image les objets suivis à l'image précédente. Les positions des objets ré-identifiés peuvent ainsi être mises à jour. Des nouvelles trajectoires sont initialisées pour les objets nouvellement détectés et inversement les trajectoires des objets sortis de la scène sont détruites.

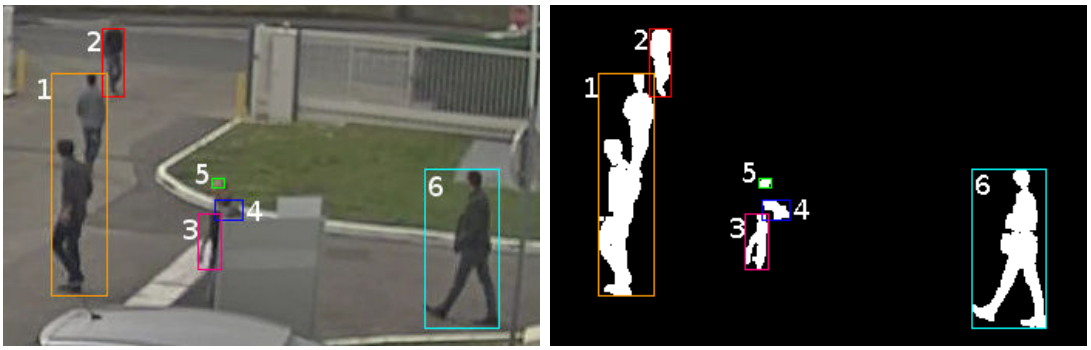
5.1.1 Définitions

Avant d'aller plus loin, nous souhaitons préciser la définition des termes suivants :

- **Blob :** composante connexe issue du masque de mouvement.
- **Objet :** modélisation d'une entité réelle dont on souhaite réaliser le suivi. Par exemple, un piéton ou un véhicule. À chaque objet nous associons, un identifiant unique (ID), un modèle (voir section 5.2.1) ainsi qu'un état (voir section 5.4.2).
- **Groupe :** modélisation d'un ensemble d'entités réelles qui ne sont plus suivies individuellement, mais collectivement. Comme pour les objets, un groupe dispose d'un ID, d'un modèle et d'un état. Il contient de plus, les identifiants des objets qu'il regroupe afin de permettre la ré-identification de ces objets lorsque le groupe se sépare.

5.1.2 Détections et difficultés associées

Notre algorithme de suivi travaille à partir des détections issues du masque de mouvement (*i.e.* blobs). Idéalement, et afin de simplifier la ré-identification des objets d'une image sur l'autre, il conviendrait qu'il n'y ait qu'une détection par objet réel et que tous les objets soient détectés. Cependant, en pratique, ce cas idéal n'est pas toujours observé. En particulier, les erreurs décrites ci-dessous sont couramment rencontrées :



(a) Image RGB (b) Masque de mouvement

FIGURE 5.1 Erreurs courantes de détection affectant le suivi (blobs 1,3,4 et 5).

• **La non détection :** le blob n'a pas été détecté alors que l'objet réel est visible par la caméra. Ce problème peut se manifester notamment pour des objets de petite taille, ou se camouflant avec le fond.

• **La fausse détection :** un blob a été détecté sans pour autant correspondre à un objet réel. Ces détections erronées peuvent se manifester notamment dans les zones dynamiques du fond, ou à cause de changements des conditions d'éclairage de la scène.

• **La fusion de blobs :** il se peut qu'un blob ne corresponde pas à un unique objet, mais à un ensemble d'objets. C'est le cas par exemple du blob (1) de la figure (5.1) qui regroupe deux piétons.

• **Le morcellement de blobs :** il se peut qu'un objet ne corresponde pas à un unique blob, mais qu'il soit morcelé en plusieurs petits blobs. C'est ce que l'on observe pour les blobs (3,4 et 5) à la figure (5.1) qui constituent chacun un morceau du piéton à cet endroit.

Partant du constat que ces erreurs sont inévitables, car le détecteur idéal n'existe pas, il convient d'intégrer dans l'algorithme de suivi un mécanisme permettant de gérer ces sources d'erreurs.

5.1.3 Aperçu de la méthode

L'algorithme de suivi que nous proposons est constitué de trois étapes principales, comme illustré à la figure (5.2).

La première étape consiste à extraire les blobs du masque de mouvement[1]. Pour chaque composante connexe extraite du masque de mouvement, nous calculons éga-

1. À proprement parler, l'extraction des blobs ne fait pas partie du tracking. Nous la mentionnons ici car c'est l'étape préliminaire de détection que nous utilisons.

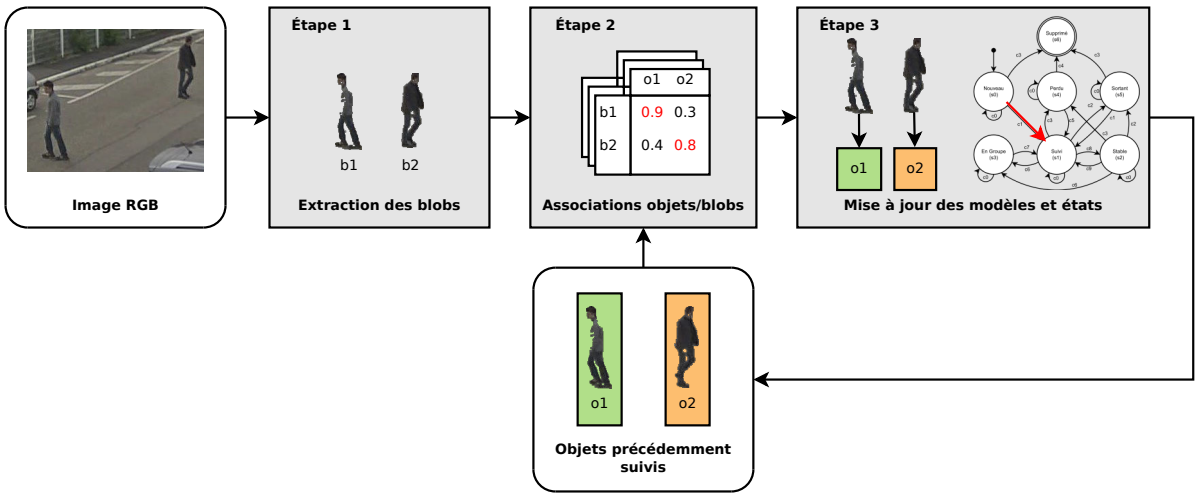


FIGURE 5.2 Aperçu de l'algorithme de suivi

lement un ensemble de caractéristiques de position (centroïde, position réelle au sol), de forme (boîte englobante, aire, taille réelle) et d'apparence (histogramme de couleur RGB), qui servent à mesurer la similarité entre un modèle d'objet et un blob.

La deuxième étape réalise les associations entre les objets précédemment suivis et les blobs nouvellement détectés. Le processus d'association est réalisé en trois passes consécutives (détaillées en section 5.3) et s'appuie sur une mesure de similarité explicitée en section 5.3.1. À l'issue de ces trois passes, les objets non associés sont considérés "perdus" et chaque blob non associé donne naissance à un nouvel objet. Cette étape d'association gère en outre les cas d'occultation inter-objets, ainsi que les formations et séparations de groupes qui en résultent.

La troisième étape réalise la mise à jour des modèles et des états des objets selon la procédure détaillée en section 5.4. D'une part, les modèles d'objets sont mis à jour grâce aux caractéristiques des blobs auxquels ils ont été associés à l'étape précédente. D'autre part, l'état de chaque objet est mis à jour conformément à l'automate fini présenté en section 5.4.2 en réalisant la transition applicable à la situation de l'objet.

5.2 Modèle et état d'un objet

Comme nous l'avons précisé en introduction, chaque objet suivi est composé d'un identifiant unique, d'un modèle permettant de caractériser et ré-identifier celui-ci au cours de la séquence vidéo et d'un état qui permet d'adapter les différents traitements (stratégie d'association, mesure de similarité) pour permettre un traitement optimal de chaque objet selon sa situation courante.

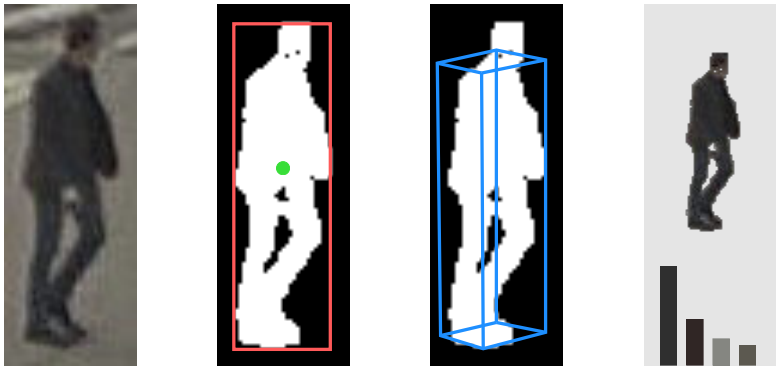


FIGURE 5.3 Illustration des principales caractéristiques du modèle d'objet

Nous présentons dans ce qui suit le modèle d'objet utilisé (5.2.1) ainsi que les différents états possibles d'un objet (5.2.2).

5.2.1 Modèle d'objet

Le modèle d'objet que nous utilisons regroupe un ensemble de caractéristiques qui permettent à la fois de représenter l'objet à l'instant considéré, mais aussi de prédire sa position et ses dimensions futures afin de faciliter sa ré-identification à l'image suivante.

Ainsi le modèle d'objet que nous utilisons (voir illustration à la figure 5.3) regroupe les caractéristiques suivantes :

• **position, taille et surface apparente 2d :** c'est-à-dire les coordonnées (x, y) du centroïde de l'objet, les dimensions de sa boîte englobante 2d (w, h) et son aire A ;

• **position, orientation et taille 3d :** encodées sous la forme d'un cuboïde 3d posé sur le sol à la position $(x_c, y_c, 0)$, de dimensions (w_c, d_c, h_c) et dirigé dans la direction (θ_z) ;

• **apparence :** synthétisée sous la forme d'un histogramme de couleurs RGB. Nous utilisons un histogramme 3d et quantifions les composantes R, G et B sur 4 bits chacunes.

La plupart des caractéristiques formant le modèle d'objet peuvent être initialisées directement à partir des caractéristiques du blob ayant conduit à la création de l'objet en question. Ainsi, à la création d'un nouvel objet, nous extrayons du blob la position 2d du centroïde du blob, sa boîte englobante 2d, sa surface apparente ainsi que son histogramme de couleur. En revanche, l'initialisation du cuboïde est plus délicate car il s'agit d'inférer une forme 3d à partir d'une projection 2d.

Nous formulons ce problème d'estimation des paramètres du cuboïde en un problème d'optimisation d'énergie. Nous appelons $r(x_c, y_c, w_c, d_c, h_c, \theta_z)$ la fonction qui dessine la silhouette du cuboïde défini par les paramètres donnés, telle qu'elle serait vue par la caméra ; m est le masque de mouvement courant ; R_1 la boîte englobante de la silhouette, R_2 la boîte englobante du blob, et $I_{R_1 \cup R_2}$ la fonction indicatrice de la région $R_1 \cup R_2$. Nous définissons alors l'énergie suivante :

$$E(x_c, y_c, w_c, d_c, h_c, m) = ||(r(x_c, y_c, w_c, d_c, h_c, \theta_z) - m).I_{R_1 \cup R_2}||_{L_2} \tag{5.1}$$

Cette énergie correspond à la distance L_2 entre la silhouette prédite du cuboïde et le masque de détection sur la région $R_1 \cup R_2$. L'estimation des paramètres du cuboïde est alors réalisée via la minimisation suivante :

$$[x_c, y_c, w_c, d_c, h_c, \theta_z]_0 = \text{argmin}(E(x_c, y_c, w_c, d_c, h_c, \theta_z)) \tag{5.2}$$

Ne disposant pas des informations de gradient d'énergie pour guider l'optimisation, nous utilisons la méthode de Nelder-Mead (Nelder and Mead, 1965), qui ne requiert que les valeurs de la fonction à optimiser (pas ses dérivées). Cette méthode d'optimisation itérative délimite l'espace de recherche à l'aide d'un simplexe [2] que l'on déforme, déplace et réduit progressivement afin que ses sommets convergent en un point où la fonction est minimale. Cette approche est heuristique et ne garantit pas la convergence vers le minimum global. C'est pourquoi nous adoptons la stratégie donnée par l'algorithme 3 pour pallier ces inconvénients et garantir un temps d'exécution borné ($M_1 \times M_2$ itérations).

Pour faciliter l'estimation initiale, nous réduisons le nombre de paramètres à optimiser en forçant une orientation $\theta_z = 0$ et en imposant une base carrée pour le cuboïde ($w_c = d_c$). Ces contraintes sont relâchées pour les estimations ultérieures afin de permettre au système de converger vers une solution plus proche de la réalité.

5.2.2 État d'un objet

À chaque objet nous associons un état qui permet de prendre en compte l'historique d'un objet et ainsi d'adapter les différents traitements (stratégie d'association, mesure de similarité, mise à jour) en fonction de cet état.

Nous décrivons les différents états possibles d'un objet dans ce qui suit ; les conditions de transitions entre ces différents états, formalisées sous la forme d'un automate fini,

2. enveloppe convexe d'un ensemble de (n+1) points dans un espace à n dimensions. Exemples courants : segment de droite (1D), triangle (2D), tétraèdre (3D).

Algorithme 3 : Stratégie d'optimisation utilisée pour l'estimation des cuboïdes.

Input : vecteur de paramètres initial v_0
Output : vecteur de paramètres optimal v_{best}
Function `RunOptimization`

```
v ← v_0;
for i ← 1 to M_1 do
    /* Créé un simplexe V aléatoirement autour du point v_best     */
    V ← InitSimplexAround(v_best);
    /* Exécute M_2 itérations de Nelder-Mead                       */
    for j ← 1 to M_2 do
        V ← NelderMead(V);
    /* Mémorise le centre v du simplexe V                          */
    v ← Barycenter(V);
    /* Retient la solution ayant un coût minimal                   */
    if Cost(v) < Cost(v_best) then
        v_best ← v;
return v_best;
```

seront précisées ultérieurement (section 5.4.2).

• **NOUVEAU (s_0) :** c'est l'état initial de tout objet et correspond au cas où l'objet vient d'être créé et n'est pas encore totalement entré dans la scène (par exemple parce qu'entrant par un bord de l'image). Cet état permet, d'une part, de retenir provisoirement les objets qui sont en train d'entrer dans la scène sans pour autant être totalement visibles et, d'autre part, de permettre de filtrer les détections en marge de la scène (au bord de l'image ou dans le ciel par exemple) qui ne passeront jamais à l'état SUIVI (s_1).

• **SUIVI (s_1) :** un objet dans cet état est entièrement dans la scène, mais sa stabilité n'est pas encore établie. Nous reviendrons sur cette notion de stabilité lorsque nous expliquerons les conditions de transition vers l'état STABLE (s_2) à la section 5.4.2.

• **STABLE (s_2) :** l'objet est entièrement dans la scène et est considéré stable. Cet état se distingue de l'état SUIVI (s_1) car il traduit un degré de confiance dans le suivi accru par rapport aux objets à l'état SUIVI (s_1). Cette distinction n'est pas fondamentale pour le fonctionnement de l'algorithme de suivi lui-même, mais permet à d'éventuels traitements de plus haut niveau d'être informé de cette meilleure confiance dans le suivi d'un objet.

• **EN GROUPE (s_3) :** l'objet n'est plus suivi individuellement, seul le groupe auquel il appartient est suivi. Cet état permet de conserver temporairement les objets qui ne peuvent plus être suivi individuellement afin de permettre de réaffecter correctement ceux-ci lorsque le groupe se scindera.

• **PERDU** (s_4) **:** l'objet n'a pas été associé, par exemple à cause d'une occultation par un obstacle ou une mauvaise détection. Cependant, son modèle est maintenu en mémoire, afin de permettre de le retrouver s'il venait à réapparaître ultérieurement.

• **SORTANT** (s_5) **:** l'objet est en train de sortir de la scène (par exemple en s'approchant du bord). À la différence de l'état PERDU (s_4), qui traduit la perte d'un objet généralement due à un problème de détection, l'état SORTANT (s_5) traduit l'intention d'un objet suivi de sortir de la scène. Il faut également le distinguer de l'état NOUVEAU (s_0), car bien qu'à la bordure de la scène considérée, l'objet SORTANT (s_5) est entré dans la scène et s'apprête à en sortir.

• **SUPPRIMÉ** (s_6) **:** l'objet n'est plus suivi et peut être oublié. C'est l'état final de tout objet d'intérêt.

5.3 Association objets / blobs

L'algorithme d'association objets/blobs utilise la notion de matrice de similarité S, dont les éléments s_{ij} sont les scores de similarité entre l'objet i et le blob j. La figure 5.4 montre des exemples de telles matrices de similarité dans différents cas.

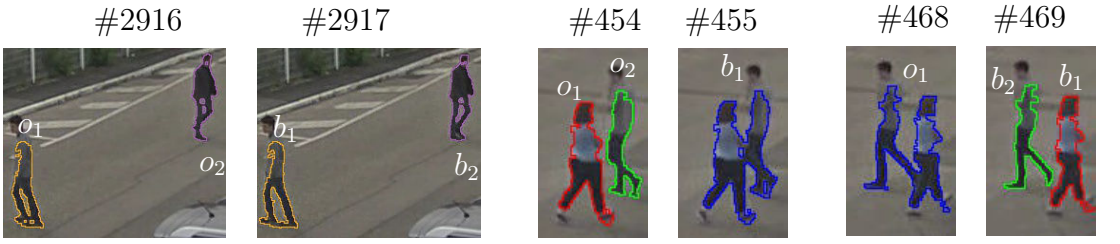
Dans le premier cas (figure 5.4a), les maxima de similarité permettent de réaliser sans ambiguïté les associations objets/blobs : l'objet o_1 est associé au blob b_1 et l'objet o_2 est associé au blob b_2. Ce sont des associations simples entre un objet et un blob.

Dans le deuxième cas (figure 5.4b), les maxima de similarité ne permettent pas de réaliser une association simple entre un objet et un blob. En effet, les objets o_1 et o_2 sont tous les deux très similaires au blob b_1. Ce cas se manifeste typiquement au début d'une occultation.

Dans le troisième cas (figure 5.4c), nous montrons la situation complémentaire à la précédente où les blobs b_1 et b_2 sont tous les deux très similaires à l'objet o_1. Ce cas se manifeste typiquement à la fin d'une occultation.

L'algorithme d'association que nous avons développé se déroule en trois passes successives, chacune tentant d'associer les objets et blobs n'ayant pas encore été associés. Un aperçu du déroulement de ces trois passes est donné à la figure 5.5.

L'intérêt de ce déroulement en trois passes est le suivant : la première passe (ϕ_1) permet de reconduire les associations relativement sûres et ainsi diminuer le nombre d'objets et blobs en entrée de la seconde passe (ϕ_2), plus complexe et donc plus sujette à faire de mauvaises associations. La deuxième passe (ϕ_2) gère les associations multiples, et utilise le critère de recouvrement au lieu du critère de forme afin de mieux détecter les formations et destructions de groupes. La troisième passe (ϕ_3) tente d'associer les



S	b_1	b_2
o_1	**0.970**	0.659
o_2	0.641	**0.952**

(a) Association simple 1 objet 1 blob

S	b_1
o_1	**0.940**
o_2	**0.913**

(b) Début d'occultation

S	b_1	b_2
o_1	**0.922**	**0.947**

(c) Fin d'occultation

FIGURE 5.4 Exemples de matrices de similarité (S) entre objets à l'instant $(t-1)$ et blobs à l'instant (t). Les contours du masque de mouvement à l'instant indiqué sont dessinés en couleur. Les nombres en gras dans les matrices de similarité indiquent que les objets et blobs s'y rapportant ont été associés.

objets restants, quitte à utiliser un seuil d'association moindre, afin d'éviter la création erronée de nouveaux objets ou le marquage erroné d'objets comme non associés.

Avant de présenter dans le détail chacune des trois passes d'associations, nous commençons par préciser le mode de calcul du score de similarité entre un objet et un blob.

5.3.1 Mesure de similarité

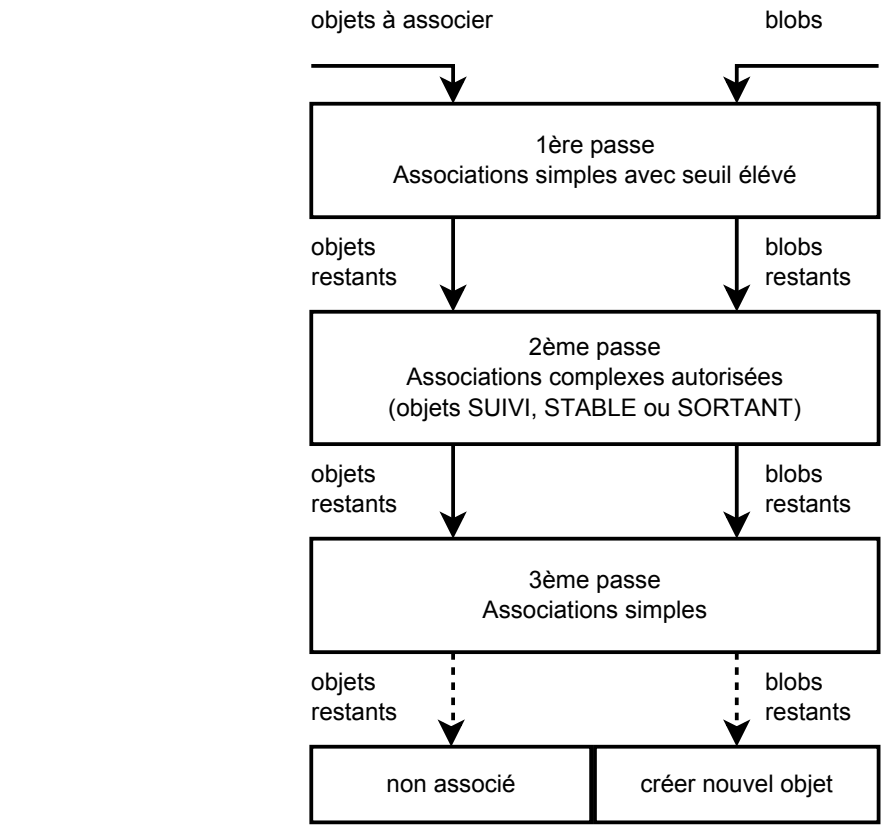
La phase d'association entre objets suivis à l'instant $t-1$ et blobs nouvellement détectés à l'instant t fait intervenir une mesure de similarité entre un objet i et un blob j. Cette mesure prend des valeurs comprises entre 0 (objet et blob très différents) et 1 (objet et blob très similaires) et permet d'évaluer la probabilité d'associer l'objet i au blob j. Notons que, pour faciliter cette comparaison, nous compensons le retard temporel des objets sur les blobs (les objets suivis relèvent de l'image à l'instant $t-1$, alors que les blobs sont des détections à l'instant t) en prédisant les modèles des objets à l'instant t à partir du modèle à l'instant $t-1$ grâce à un filtre de Kalman.

La mesure de similarité développée fait intervenir quatre critères de similarité, chacun comparant un aspect particulier de l'objet et du blob considérés.

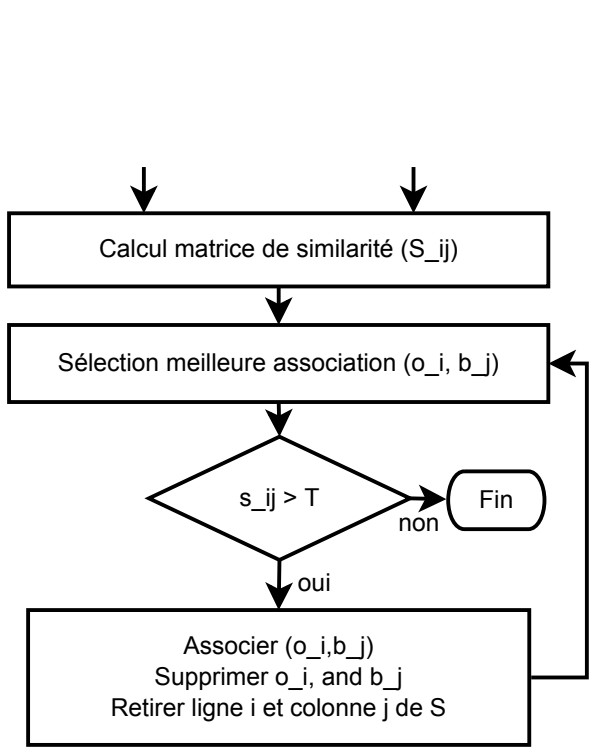
Le premier critère porte sur la distance d_{ij} entre le centroïde prédit de l'objet i et le centroïde du blob j :

$$s_{ij}^{p} = \begin{cases} 1 - d_{ij}/d_{\max} & \text{si } d_{ij} \leq d_{\max} \\ 0 & \text{sinon} \end{cases} \tag{5.3}$$

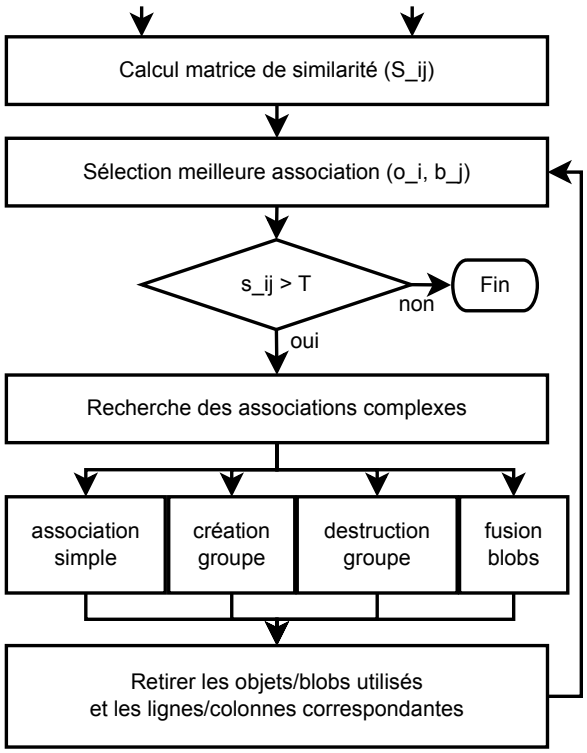
avec $d_{\max}$ la distance maximale admissible entre objets et blobs. $d_{\max}$ joue le rôle de



(a) Aperçu global de la méthode d'associations



(b) Algorithme glouton pour les associations simples



(c) Algorithme glouton pour les associations complexes

FIGURE 5.5 Algorithmes d'association

facteur de normalisation et permet ainsi de borner l'indice de similarité s^p_{ij} entre 0 (centroïdes trop éloignés) et 1 (centroïdes confondus). Ainsi, cet indice de similarité varie continûment entre 0 et 1 et évalue la proximité spatiale entre l'objet i et le blob j.

Le deuxième critère porte sur la forme, via les aires et hauteurs réelles, de l'objet i et du blob j :

$$s^s_{ij} = 1 - \sqrt{\frac{(\Delta A_{ij})^2 + (\Delta h_{ij})^2}{2}} \tag{5.4}$$

où ΔA_{ij} et Δh_{ij} représentent la différence relative entre l'aire (respectivement la hauteur réelle) de l'objet i et du blob j. Nous utilisons comme mesure de différence relative la définition suivante :

$$\Delta X_{ij} = \frac{X_i - X_j}{\max(X_i, X_j)} \tag{5.5}$$

Ainsi si les aires et hauteurs réelles de l'objet i et du blob j sont proches, leur différence relative sera nulle et donc s^s_{ij} sera proche de 1.

Le troisième critère porte sur l'apparence. Plus précisément, il mesure le recouvrement entre l'histogramme de couleur de l'objet i et l'histogramme de couleur de la région dans l'image courante correspondant au blob j. Il est calculé de la manière suivante :

$$s^a_{ij} = BC_{i,j} = \frac{1}{B}\sum_{b=1}^{B}\sqrt{q_i(b)q_j(b)} \tag{5.6}$$

Ici, B est le nombre de classes d'un histogramme de couleur, et $BC_{i,j}$ est le coefficient de Bhattacharyya (Bhattacharyya, 1943) entre l'histogramme de couleur q_i du modèle d'objet i et l'histogramme de couleur q_j du blob j. Pour des histogrammes présentant un fort recouvrement, s^a_{ij} tendra vers 1. En revanche, pour des histogrammes n'ayant pas un bon recouvrement s^a_{ij} tendra vers 0.

Le dernier critère porte sur le recouvrement de la silhouette prédite du cuboïde représentant l'objet i, et le blob j :

$$s^o_{ij} = \begin{cases} 1 & \text{si la silhouette prédite du cuboïde de l'objet } i \text{ intersecte le blob } j \\ 0 & \text{sinon.} \end{cases} \tag{5.7}$$

À l'inverse des autres critères qui varient continûment entre 0 et 1, le critère de recouvrement est binaire et ne prend donc que les valeurs 0 ou 1. Nous avions également testé une version de ce critère, variant continûment entre 0 et 1, basée sur le taux de recouvrement entre silhouette et blob. Cependant, la version binaire de ce critère donne de meilleurs résultats du fait de sa capacité à mieux faire ressortir les cas d'associations

complexes (début ou fin d'occultation, fusion de blobs) que ne le faisait la version basée sur le taux de recouvrement.

Finalement, nous définissons la similarité entre un objet i et un blob j par

$$s_{ij} = \sqrt{\frac{\alpha_p(s_{ij}^p)^2 + \alpha_s(s_{ij}^s)^2 + \alpha_a(s_{ij}^a)^2 + \alpha_o(s_{ij}^o)^2}{\alpha_p + \alpha_s + \alpha_a + \alpha_o}} \tag{5.8}$$

Les coefficients α définissent les poids de chacun des critères de similarité et permettent d'adapter la mesure de similarité en fonction du contexte comme nous le verrons dans la section (5.3).

Par construction, s_{ij} est compris entre 0 et 1 et mesure la similarité entre l'objet i et le blob j.

5.3.2 Première passe : associations sûres

La première passe (notée ϕ_1) consiste à associer les objets et blobs ayant une forte similarité. Durant cette phase, tous les objets hormis ceux dans l'état EN GROUPE ou SUPPRIMÉ sont considérés. Les poids des différents critères de similarité sont choisis ainsi :

- ($\alpha_p = 1$, $\alpha_s = 0$, $\alpha_a = 0$, $\alpha_o = 0$) pour un objet NOUVEAU, car seule sa position est jugée fiable ;
- ($\alpha_p = 0$, $\alpha_s = 1$, $\alpha_a = 1$, $\alpha_o = 0$) pour un objet PERDU, car ni sa position, ni son recouvrement ne peuvent être considérés fiables ;
- ($\alpha_p = 1$, $\alpha_s = 1$, $\alpha_a = 1$, $\alpha_o = 0$) pour un objet SUIVI, STABLE ou SORTANT.

La stratégie d'association est de type glouton (voir algorithme figure 5.5b), et le seuil minimal d'association T_1 est choisi élevé (0.9 en pratique).

5.3.3 Deuxième passe : associations complexes

La deuxième passe (notée ϕ_2) tente d'associer les blobs et les objets dans l'état SUIVI, STABLE ou SORTANT restants en prenant en compte les possibilités d'associations multiples (voir algorithme figure 5.5c). Les poids des différents critères sont les suivants ($\alpha_p = 1, \alpha_s = 0, \alpha_a = 1, \alpha_o = 1$). Le seuil minimal d'association T_2 est fixé inférieur à T_1 (0.66 en pratique). Le raisonnement permettant de prendre en compte les associations multiples est le suivant : une fois la meilleure association sélectionnée (o_i, b_j), on recherche dans la ligne i, d'autres blobs distincts de b_j dont la similarité avec l'objet o_i est supérieure au seuil T_2. L'ensemble de ces blobs est noté $\mathcal{B}$. On procède de même pour la colonne j, où l'on recherche d'autres objets, distinct de o_i dont la similarité avec

le blob b_j est supérieure au seuil T_2. L'ensemble de ces objets est noté $\mathcal{O}$. On a alors plusieurs cas :

- **Si $\mathcal{B} = \mathcal{O} = \emptyset$:** on est dans le cas d'association simple 1 objet, 1 blob : l'association (o_i, b_j) est validée (comme montré à la figure 5.4a).

- **Si $\mathcal{B} \neq \emptyset$ et $\mathcal{O} = \emptyset$:** on est dans le cas où l'objet o_i peut être associé à plusieurs blobs. Ce cas peut correspondre, soit à un défaut de segmentation (*ie* où l'objet o_i est morcelé en plusieurs blobs) auquel cas il convient de fusionner les blobs correspondants en un seul blob et associer ce dernier avec l'objet o_i, soit à la fin d'une occultation (*ie* si o_i est un groupe), auquel cas le groupe o_i est détruit et on procède à une association gloutonne simple (voir algorithme figure 5.5b, avec un seuil $T_g = 0.66$ et des poids $\alpha_p = 0$, $\alpha_s = 1$, $\alpha_a = 1$ et $\alpha_o = 0$) entre les membres du groupe et les blobs de $\mathcal{B} \cup b_j$ (comme montré à la figure 5.4c).

- **Si $\mathcal{O} \neq \emptyset$ et $\mathcal{B} = \emptyset$:** on est dans le cas où le blob b_j peut être associé à plusieurs objets. Nous considérons que ceci marque le début d'une occultation et, dans ce cas, il convient de créer un groupe comportant les objets $\mathcal{O} \cup o_i$ et d'associer celui-ci au blob b_j (comme montré à la figure 5.4b).

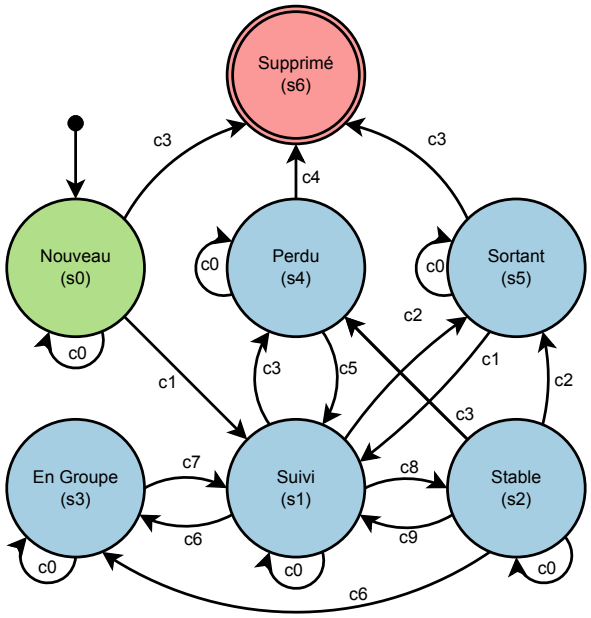
- **Si $\mathcal{O} \neq \emptyset$ et $\mathcal{B} \neq \emptyset$:** par souci de simplicité, nous négligeons le fait qu'il y ait d'autres blobs que b_j et nous nous ramenons au cas de création d'un groupe.

5.3.4 Troisième passe : associations restantes

Pour la troisième et dernière passe (notée ϕ_3), tous les objets non encore associés, hormis ceux dans l'état EN GROUPE ou SUPPRIMÉ, sont considérés afin d'être associés aux blobs restants. Les poids des différents critères de similarité et la stratégie d'association (algorithme figure 5.5b) sont les mêmes que pour la première passe (ϕ_1), seul le seuil minimal d'association change : il est choisi plus bas que T_1 (en pratique $T_3 = 0.75$) afin de permettre d'effectuer les associations qui n'auraient pas été faites durant les deux passes précédentes.

5.3.5 Création des nouveaux objets

Enfin, à l'issue de ces trois phases d'associations, pour chaque blob restant (non associé), un nouvel objet est créé. Les objets restants sont marqués comme non associés.



Id	Condition de transition
c_0	Aucune des autres transitions ne s'applique
c_1	L'objet est complètement dans la scène
c_2	L'objet n'est pas complètement dans la scène
c_3	L'objet n'a pas été associé à un blob
c_4	L'objet a disparu depuis trop longtemps ($> T_f$)
c_5	L'objet a été associé
c_6	L'objet entre dans un groupe
c_7	L'objet sort d'un groupe
c_8	L'objet a été associé durant la 1re passe
c_9	L'objet n'a pas été associé durant la 1re passe

FIGURE 5.6 Schématisation de l'automate fini utilisé. L'état initial et l'état final sont respectivement NOUVEAU (s_0) et SUPPRIMÉ (s_6). Les transitions sont notées de c_0 à c_9.

5.4 Mise à jour des objets

5.4.1 Mise à jour du modèle d'objet

Concernant la mise à jour des objets, il s'agit d'utiliser les caractéristiques des blobs détectés afin de mettre à jour les modèles des objets suivis.

Nous utilisons un filtre de Kalman pour réaliser cette mise à jour pour toutes les caractéristiques de type position, taille et orientation. Pour ce faire, nous faisons l'hypothèse de déplacement à vitesse constante et nous considérons que l'objet est rigide. Pour les autres caractéristiques (aire, histogrammes de couleur) un simple remplacement des valeurs est effectué.

5.4.2 Mise à jour de l'état d'un objet et automate fini

L'évolution d'un objet est modélisée à l'aide de l'automate fini représenté figure (5.6).

Nous détaillons, à présent, les transitions possibles à partir de chacun des états :

• **NOUVEAU (s_0) :** un objet dans cet état n'est, par définition, pas encore entré totalement dans la scène. Trois transitions sont possibles pour un objet dans un tel état :
– si l'objet disparait de la scène (c_3), il passe dans l'état SUPPRIMÉ (s_6) ;

– si l'objet entre entièrement dans la scène (c_1), il passe alors dans l'état SUIVI (s_1) ;
– si aucune des deux transitions précédentes ne s'applique (c_0), l'objet reste dans l'état NOUVEAU (s_0).

• **SUIVI (s_1) :** un objet dans cet état est, par définition, totalement dans la scène mais sa stabilité n'est pas encore établie. Cinq transitions sont possibles pour un objet dans un tel état :
– si l'objet a été inclus dans un groupe (c_6), celui-ci passe à l'état EN GROUPE (s_3) ;
– si l'objet n'a pas été associé à un blob (c_3), celui-ci passe à l'état PERDU (s_4) ;
– si l'objet n'est plus entièrement dans la scène (c_2), il passe à l'état SORTANT (s_5) ;
– si l'objet a été associé durant la première phase d'association (c_8), il passe à l'état STABLE (s_2) ;
– si aucune des transitions précédentes ne s'applique (c_0), l'objet reste dans l'état SUIVI (s_1).

• **STABLE (s_2) :** un objet dans cet état est, par définition, totalement dans la scène et est considéré stable (car associé durant la première passe avec seuil élevé). Cinq transitions sont possibles pour un objet dans un tel état :
– si l'objet a été inclus dans un groupe (c_6), celui-ci passe à l'état EN GROUPE (s_3) ;
– si l'objet n'a pas été associé à un blob (c_3), celui-ci passe à l'état PERDU (s_4) ;
– si l'objet n'est plus entièrement dans la scène (c_2), il passe à l'état SORTANT (s_5) ;
– si l'objet n'a pas été associé durant la première phase d'association (c_9), il passe à l'état SUIVI (s_1) ;
– si aucune des transitions précédentes ne s'applique (c_0), l'objet reste dans l'état STABLE (s_2).

• **EN GROUPE (s_3) :** un objet dans cet état n'est, par définition, plus suivi individuellement mais est inclus dans un groupe. Deux transitions sont possibles pour un objet dans un tel état :
– si le groupe auquel appartient l'objet a été scindé (c_7), l'objet retourne à l'état SUIVI (s_1) ;
– sinon (c_0), l'objet reste dans l'état EN GROUPE (s_3).

• **PERDU (s_4) :** un objet dans cet état n'a, par définition, pas été associé dernièrement. Trois transitions sont possibles pour un objet dans un tel état :
– si l'objet n'est pas réapparu dans la scène depuis trop longtemps (c_4), c'est-à-dire non associé consécutivement plus de T_f fois, il passe à l'état SUPPRIMÉ (s_6) ;
– au contraire si l'objet est retrouvé (c_5), il passe à l'état SUIVI (s_1) ;
– si aucune des transitions précédentes ne s'applique (c_0), l'objet reste dans l'état PERDU (s_4).

• **SORTANT (s_5) :** un objet dans cet état n'est, par définition, plus entièrement dans la scène après avoir été dans la scène. Trois transitions sont possibles pour un objet dans un tel état :
– si l'objet n'est plus détecté (c_3), il passe à l'état SUPPRIMÉ (s_6) ;
– si l'objet entre à nouveau à l'intérieur de la scène (c_1), il passe dans l'état SUIVI (s_1) ;
– si aucune des transitions précédentes ne s'applique (c_0), l'objet reste dans l'état SORTANT (s_5).

• **SUPPRIMÉ (s_6) :** un objet dans cet état n'est plus suivi et peut être oublié. Aucune transition n'est applicable.

Nous illustrons dans la figure (5.7) un exemple d'évolution d'un objet de sa première à sa dernière détection, afin d'illustrer les différents états et transitions en jeu.

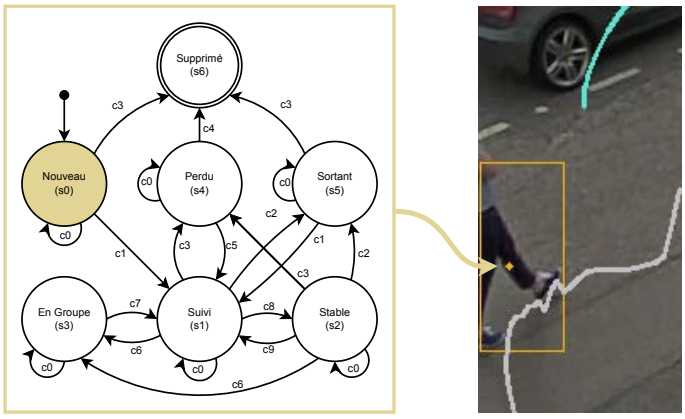
5.4.3 L'objet est-il complètement dans la scène ?

Nous utilisons dans la définition des états et des transitions l'expression "être complètement dans la scène" sans vraiment préciser ce que cela implique. Dans l'algorithme original (Di Lascio et al., 2013), cela signifie que l'objet doit être entièrement dans l'image : sa boîte englobante 2d ne doit pas toucher les bords (avec éventuellement une marge). Ce critère est simple et ne fait pas d'hypothèses sur la scène observée.

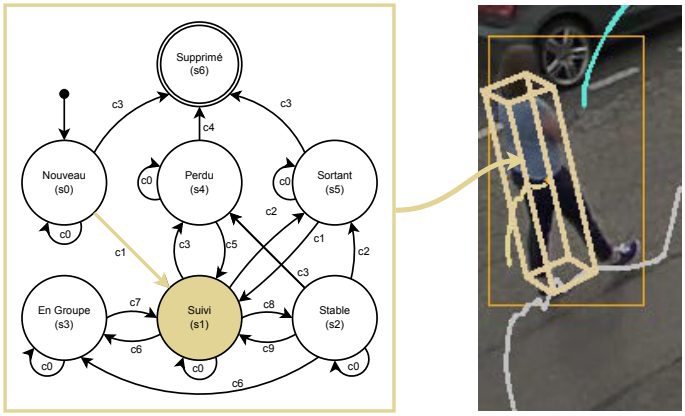
Dans notre cas cependant, nous avons à disposition un modèle de scène 3d (voir chapitre 3). C'est pourquoi nous avons revisité ce critère "être dans la scène" à l'aide de notre modèle de scène. Ainsi, à partir du modèle de scène, nous générons un rendu du masque de sol visible par la caméra (figure 5.8) ; nous testons si la boîte englobante 2d est bien entièrement dans l'image (comme l'algorithme original) et nous testons également si les sommets inférieurs de la boîte englobante 2d (ce qui correspond grossièrement à la position des points de contact de l'objet avec le sol) sont également sur le sol, c'est-à-dire correspondent à des pixels non nuls du masque de sol.

(a) Image caméra (b) Masque sémantique (c) Masque de sol

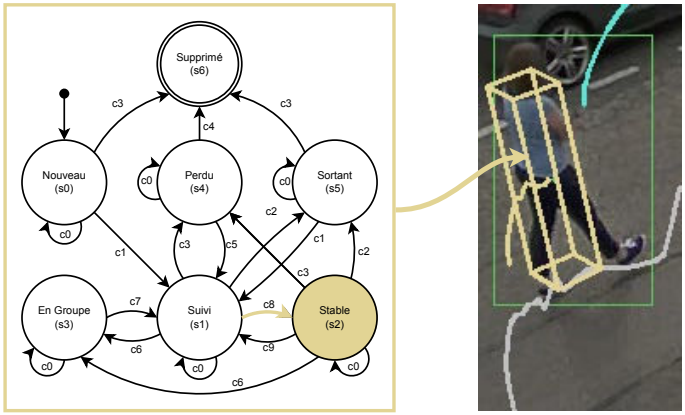
FIGURE 5.8 Exemple de carte sémantique et de masque de sol générés à partir du modèle de scène



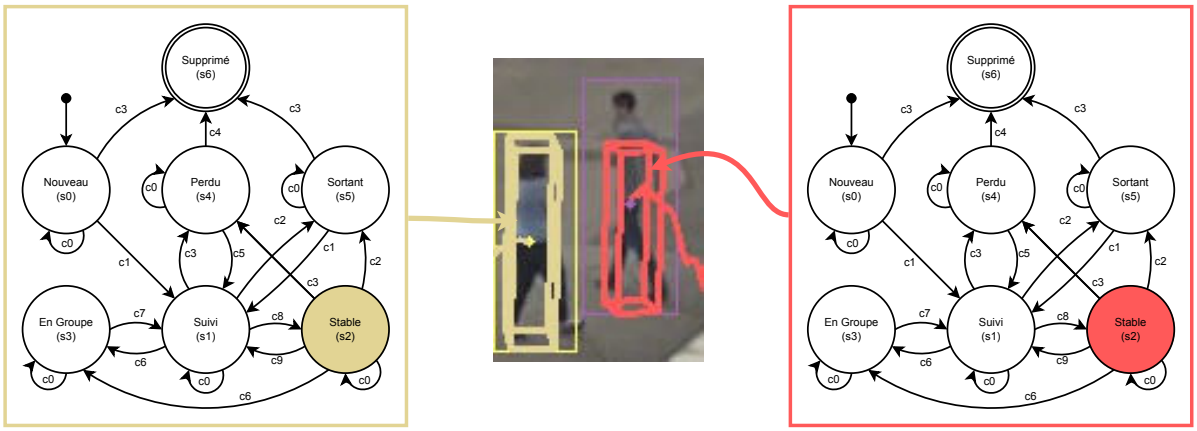
(a) L'objet est détecté pour la première fois et n'est pas encore totalement entré dans la scène. Il est dans l'état NOUVEAU.



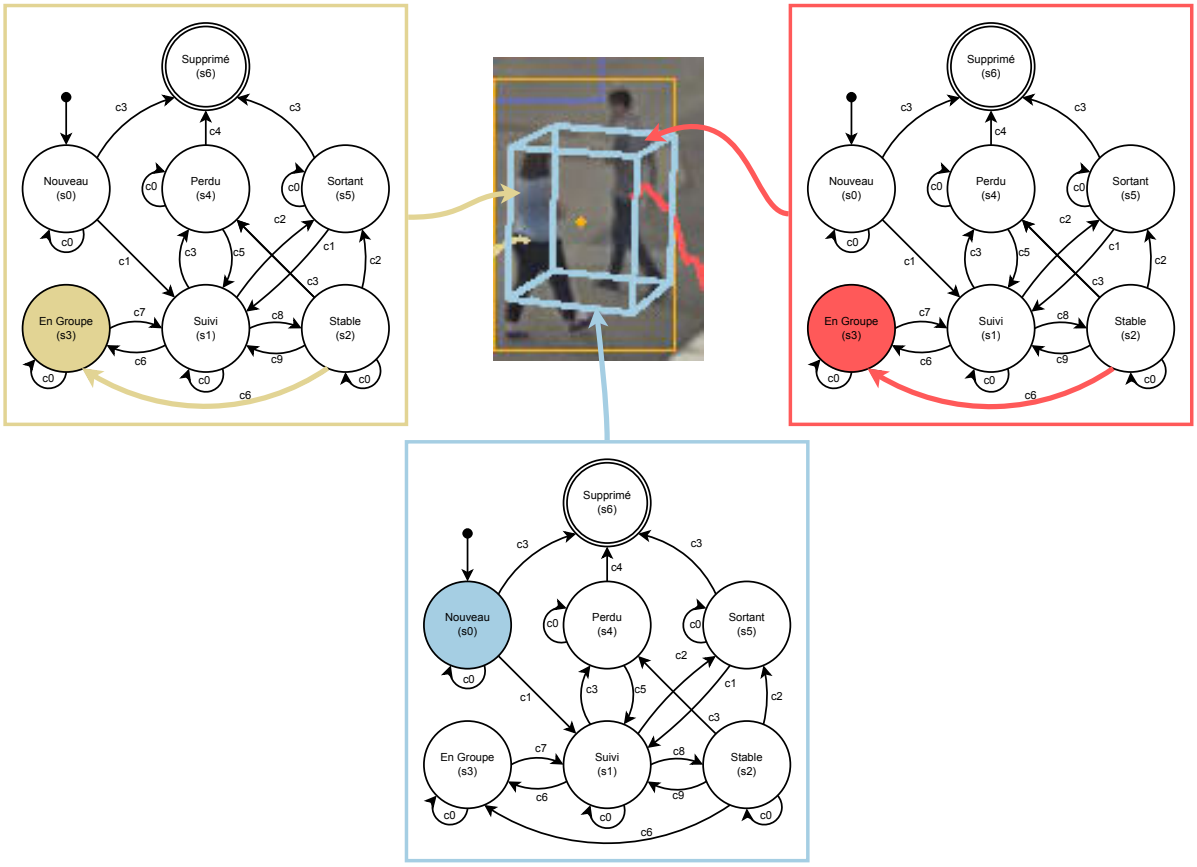
(b) L'objet vient d'entrer totalement dans la scène. Il passe alors dans l'état SUIVI.



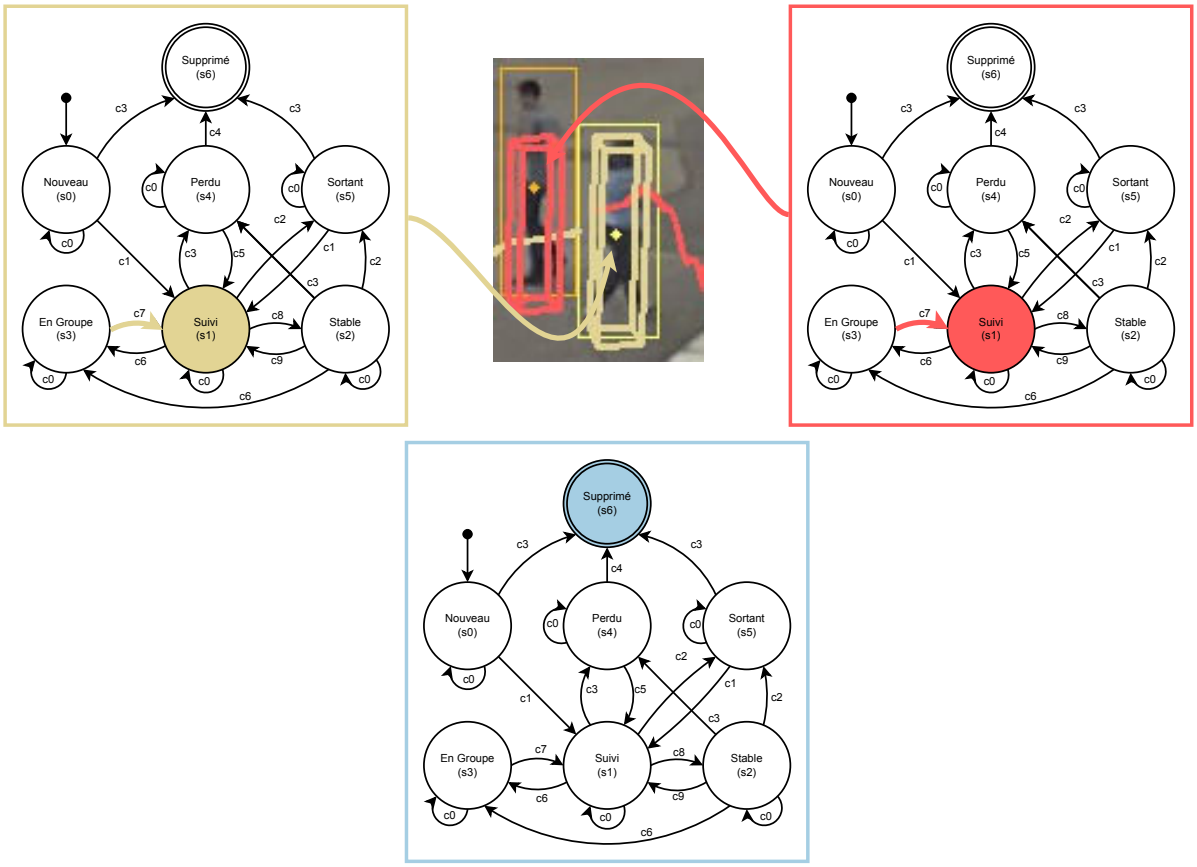
(c) L'objet est associé durant la première passe d'association, et est donc considéré stable. Il passe alors dans l'état STABLE.



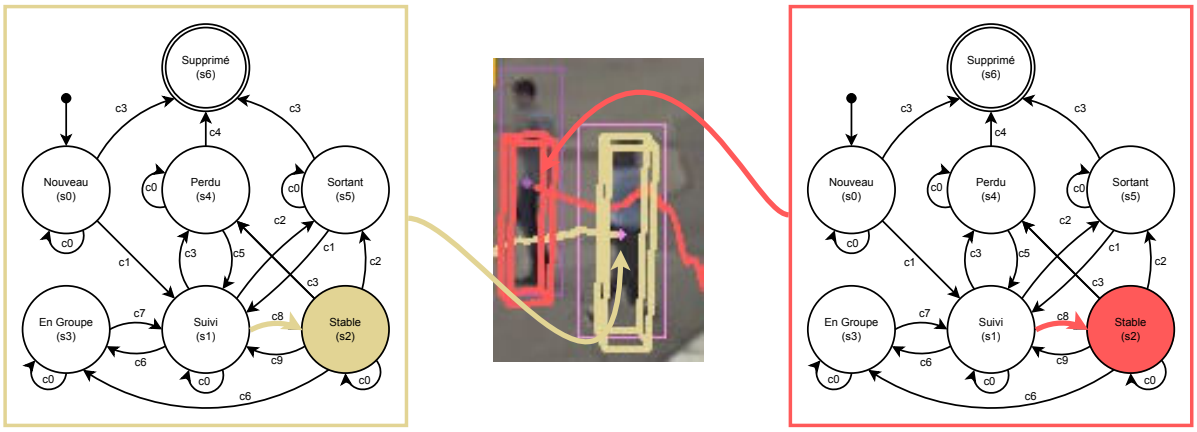
(d) Deux objets à l'état STABLE se rapprochent...



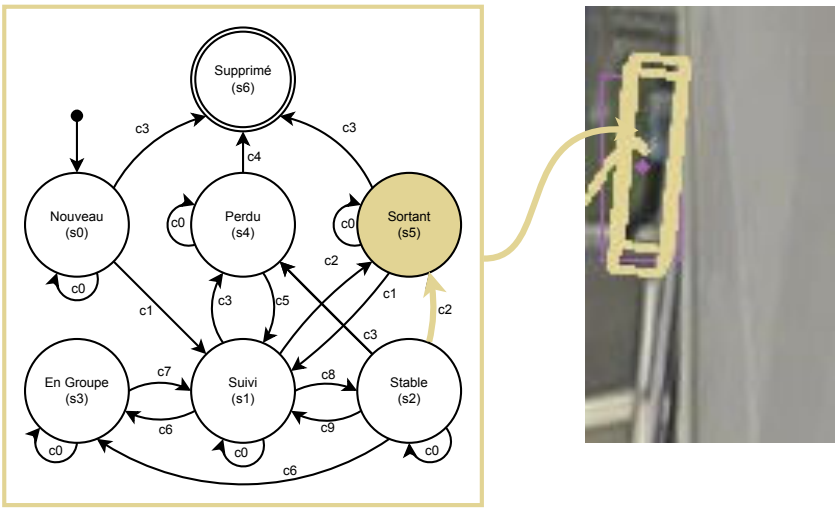
(e) ... et fusionnent : un nouveau groupe est créé et les objets passent alors dans l'état EN GROUPE.



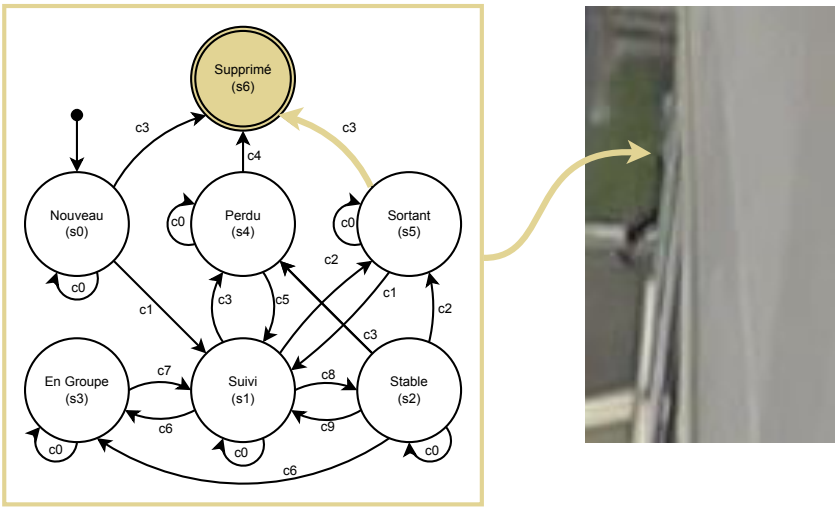
(f) Le groupe se sépare, passe à l'état SUPPRIMÉ et est détruit. Après avoir été correctement ré-identifiés, les objets membres du groupe retournent à l'état SUIVI...



(g) ... puis STABLE.



(h) L'objet s'apprête à sortir de la scène en passant derrière un bâtiment. Il passe dans l'état SORTANT.



(i) L'objet n'est plus détecté et passe donc dans l'état SUPPRIMÉ.

FIGURE 5.7 Exemple d'évolution d'un objet de sa première à sa dernière détection.

Ce critère permet, d'une part, d'autoriser les objets à rentrer et sortir de la scène en dehors des bords de l'image (par exemple, lorsque les objets passent derrière un bâtiment) et, d'autre part, cela filtre (ou tout du moins retient dans l'état NOUVEAU) les objets qui ne sont pas intéressants (cas d'une fenêtre qui s'ouvre ou de branches d'arbres mues par le vent, *etc.*).

5.5 Évaluation

5.5.1 Métriques d'évaluation

À l'aide des métriques d'évaluation, nous souhaitons évaluer certaines propriétés qu'un "bon algorithme" de suivi doit posséder, à savoir :

- À chaque instant, l'algorithme de suivi doit trouver tous les objets présents et estimer leur position le plus précisément possible.
- L'algorithme de suivi doit également préserver les identités des objets suivis tout au long de la séquence, c'est-à-dire que chaque objet doit conserver le même identifiant de son entrée dans la scène jusqu'à sa sortie.

Les métriques standards permettant d'évaluer un algorithme de suivi selon ces critères sont les métriques "CLEARMOT" décrites par Bernardin and Stiefelhagen (2008).

Avant de poursuivre la description de ces métriques, nous définissons les notations suivantes : $\mathcal{G}_t = \{g_t^i\}_{i=1..N_t}$ est l'ensemble des N_t objets véritablement présents dans la scène à l'instant t (vérité terrain) et $\mathcal{H}_t = \{h_t^i\}_{i=1..M_t}$ est l'ensemble des M_t objets suivis selon l'algorithme de suivi évalué à l'instant t (hypothèses).

Il s'agit alors de trouver, à chaque instant t, les correspondances entre les éléments de $\mathcal{G}_t$ et ceux de $\mathcal{H}_t$ et de compter :

- le nombre de correspondances trouvées noté c_t
- parmi ces correspondances, celles pour lesquelles l'identité de l'objet (selon l'algorithme de suivi) a changé depuis l'instant précédent. Leur nombre est noté is_t.
- le nombre d'éléments de $\mathcal{G}_t$ non associés. Ce nombre correspond au nombre d'objets non détectés et est noté fn_t
- le nombre d'éléments de $\mathcal{H}_t$ non associés. Ce nombre correspond au nombre de fausses détections et est noté fp_t
- l'erreur totale commise sur la position de chacun des couples objet/hypothèse établis à l'instant t. On note cette erreur d_t.

Nous pouvons alors définir les métriques suivantes :

- ***Multiple Object Tracking Precision*** **(MOTP) :**

$$\text{MOTP} = \frac{\sum_t d_t}{\sum_t c_t} \tag{5.9}$$

Cette quantité mesure l'erreur moyenne commise sur les positions des objets suivis. Elle permet d'évaluer la précision du suivi indépendamment des autres sources d'erreur (changement d'identité, non détection, fausse détection). Pour un algorithme idéal, MOTP = 0, et sa valeur est croissante à mesure que l'erreur d'estimation des positions augmente.

Notons que les mises en correspondance ne sont pas autorisées si la distance entre un objet g_t^i et une hypothèse h_t^j est trop grande, c'est-à-dire, supérieure à un seuil T définit par l'utilisateur. Ainsi, la valeur de MOTP est bornée entre $[0; T]$.

Afin de faciliter la comparaison entre méthodes, cette valeur MOTP est souvent reformulée pour varier entre 0 et 1 avec une valeur de 1 pour un algorithme idéal :

$$\text{MOTP} = 1 - (\frac{\sum_t d_t}{\sum_t c_t})/T \tag{5.10}$$

C'est cette définition que nous utilisons dans la suite de ce manuscrit.

- ***Multiple Object Tracking Accuracy*** **(MOTA) :**

$$\text{MOTA} = 1 - \frac{\sum_t(\text{fn}_t + \text{fp}_t + \text{is}_t)}{\sum_t N_t} \tag{5.11}$$

Cette quantité rassemble en un indicateur unique l'ensemble des types d'erreurs commises par l'algorithme de suivi (non détection, fausse détection, changement d'identité). Pour un algorithme idéal, MOTA = 1 et sa valeur est décroissante à mesure que le nombre d'erreurs augmente. Ce score peut être négatif.

- ***False Positive*** **(FP) :**

$$\text{FP} = \sum_t \text{fp}_t \tag{5.12}$$

Cette quantité correspond au nombre total de fausses détections (objets suivis ne correspondant pas à un objet réellement existant). Pour un algorithme idéal, FP = 0.

- ***False Negative*** **(FN) :**

$$\text{FN} = \sum_t \text{fn}_t \tag{5.13}$$

Cette quantité correspond au nombre total d'omissions (objet réellement existant non suivi). Pour un algorithme idéal, FN = 0.

- ***ID Switch* (IDs) :**

$$\text{IDs} = \sum_t \text{is}_t \tag{5.14}$$

Cette quantité correspond au nombre total de changements d'identité au cours de la séquence. Pour un algorithme idéal, IDs = 0.

Nous complétons ces métriques à l'aide de celles proposées par Li et al. (2009) qui classifient les trajectoires selon le pourcentage de la trajectoire effectivement suivi :

- ***Mostly Tracked* (MT) :** si plus de 80% des points d'une trajectoire ont été correctement estimés, la trajectoire est considérée comme "majoritairement suivie".

- ***Partially Tracked* (PT) :** si entre 20% et 80% des points d'une trajectoire ont été correctement estimés, la trajectoire est considérée comme "partiellement suivie".

- ***Mostly Lost* (ML) :** si moins de 20% des points d'une trajectoire ont été correctement estimés, la trajectoire est considérée comme "majoritairement perdue".

5.5.2 Séquences de test

Afin de tester l'algorithme de suivi, nous avons sélectionné un ensemble de séquences vidéos comportant des objets mobiles. Les caractéristiques de ces séquences sont synthétisées dans le tableau 5.1. Les séquences synthétiques ont été réalisées à l'aide du logiciel d'édition 3d Muvizu [3]. Ce logiciel permet de créer simplement des séquences vidéo comportant des objets ou des personnages animés. Ainsi nous avons réalisé quelques séquences courtes permettant de valider qualitativement le comportement de notre algorithme dans des cas simples et contrôlés, chacune évaluant un aspect particulier de notre implémentation. Par exemple, la séquence *muvizu1* met en scène une occultation totale d'un piéton par un autre piéton. La séquence *muvizu2* met en scène le croisement d'un piéton avec une voiture.

L'annotation des séquences vidéos a été réalisée à l'aide d'un outil spécifique développé en interne par l'entreprise Foxstream [4]. Une capture d'écran de cet outil est donné à la figure 5.9.

3. http://muvizu.com
4. OpenTest : développé par L. Robinault

	PETS.S2L1	**PETS01.D1**	**parking1**
			
Résolution	768 × 576	768 × 576	1280 × 960
# Images	795	2688	3600
Img / s	7	25	25
Types objets	piétons	piétons et véhicules	piétons (et véhicules)
# Objets	19	13	27

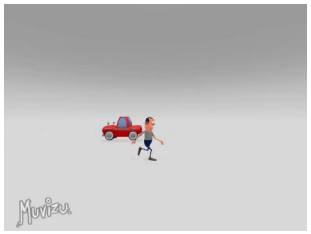
	muvizu1	**muvizu2**
		
Résolution	480 × 360	480 × 360
# Images	250	250
Img / s	25	25
Types objets	piétons	piétons et véhicules
# Objets	2	2

TABLE 5.1 Caractéristiques principales des séquences d'évaluation du suivi.

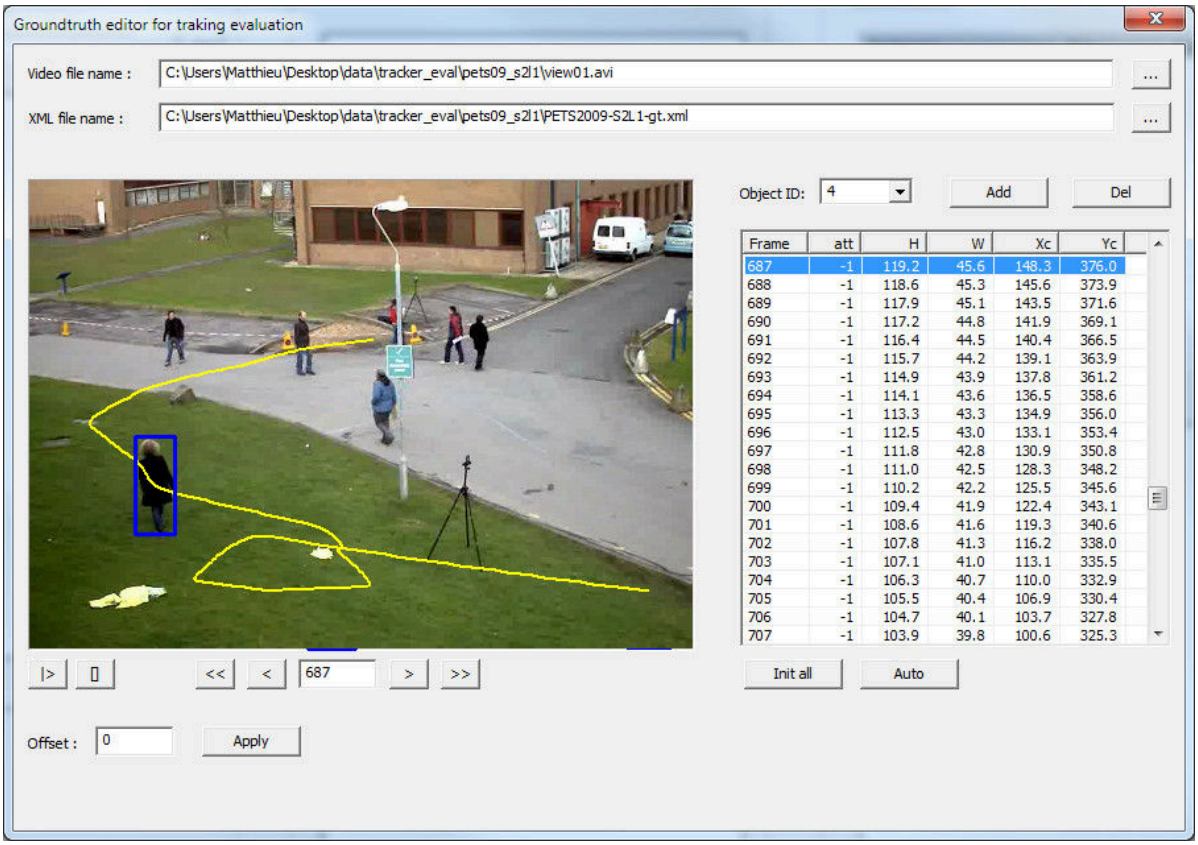


FIGURE 5.9 Capture d'écran du logiciel d'annotation de trajectoire utilisé.

L'intérêt de cet outil est de n'avoir à annoter que quelques images clés pour chaque objet, le logiciel se chargeant alors d'effectuer l'interpolation de la trajectoire et des dimensions des boîtes englobantes 2d représentant les objets pour les images situées entre les images clés. L'interpolation réalisée est linéaire, il faut donc veiller à bien choisir ses images clés en conséquence pour éviter les erreurs lors de l'interpolation. Ainsi, il devient plus facile d'annoter les séquences et d'avoir pour chaque image les boîtes englobantes 2d de chacun des objets annotés.

Grâce à ce logiciel, nous avons annoté la séquence parking1 (seulement les piétons, afin de pouvoir comparer avec la méthode de DiLascio). Pour la séquence PETS.S2L1 nous avons réutilisé les annotations fournies par A. Milan [5].

5.5.3 Évaluation

Nous avons implémenté l'algorithme décrit dans (Di Lascio et al., 2013). Cette implémentation, que nous espérons fidèle à la méthode originale, constitue notre référence,

5. http://www.milanton.de/data.html

et nous permet d'évaluer l'apport de nos modifications.

Le seuil d'association est fixé à 0.66 pour la méthode originale et, pour notre méthode, nous utilisons les valeurs fixes suivantes : $T_1 = 0.9, T_2 = T_g = 0.66, T_3 = 0.75$. De la même manière que pour la méthode originale, nous fixons T_f de manière à oublier les objets perdus depuis plus de 1 seconde.

Les masques de mouvement sont obtenus à l'aide du modèle de fond ViBe (Barnich and Van Droogenbroeck, 2011), suivi d'opérations morphologiques filtrant les pixels isolés.

Les résultats quantitatifs sont synthétisés dans la table 5.2.

TABLE 5.2 Tableau comparatif des différentes métriques de performance. Les scores en gras indiquent les meilleurs résultats par séquence et par métrique. Ici GT $= \sum_t N_t$, c'est-à-dire le nombre total d'éléments contenus dans la vérité terrain sur l'ensemble de la séquence. Les autres indicateurs ont été définis précédemment.

Vidéo	Méthode	MOTA	MOTP	GT	FP	FN	IDs	MT	PT	ML
PETS.S2L1	DiLascio13	0.759	0.594	4644	176	762	**179**	13	6	0
	Rogez	**0.774**	**0.617**	4644	**114**	**737**	199	**14**	**5**	0
parking1	DiLascio13	0.705	0.510	15439	226	4196	**135**	12	15	0
	Rogez	**0.729**	**0.516**	15439	**200**	**3754**	228	12	15	0

D'après ces résultats, nos modifications permettent d'améliorer la plupart des métriques de performances. Cette amélioration est principalement due à une meilleure gestion des groupes.

Nous donnons un exemple illustrant cette meilleure gestion des groupes grâce au critère de recouvrement à la figure 5.10.

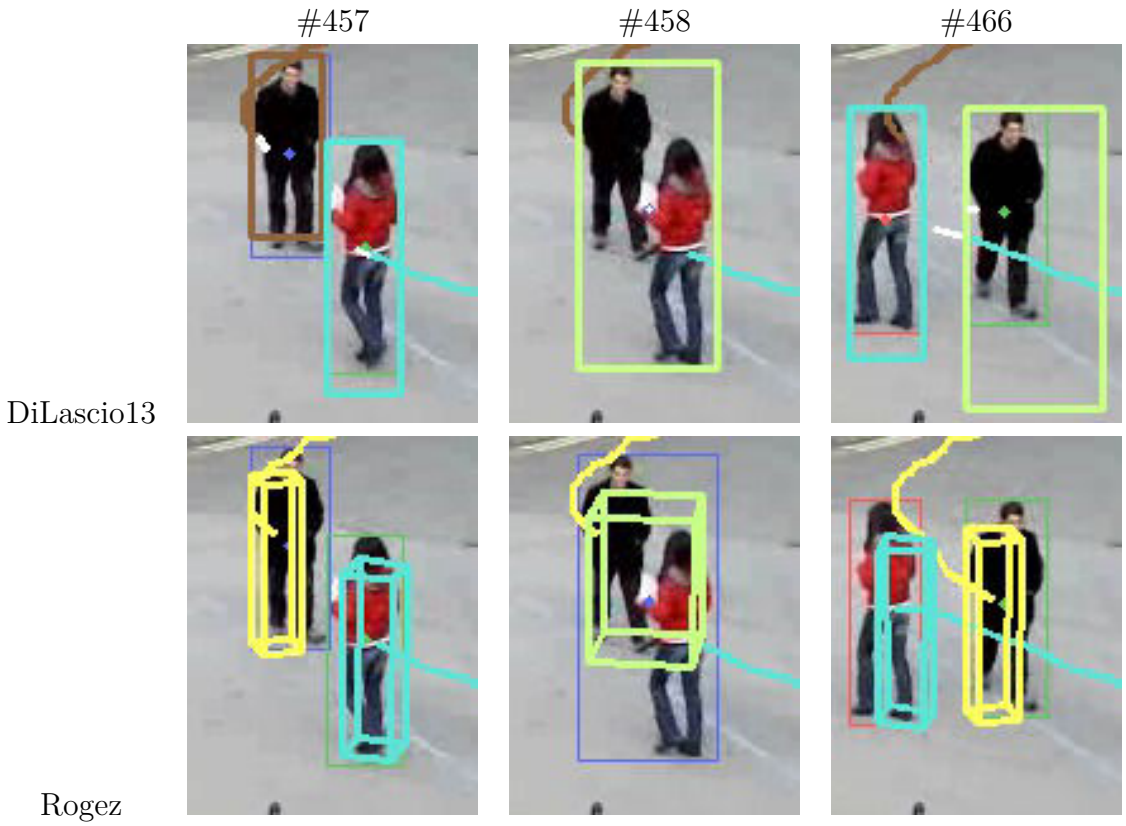


FIGURE 5.10 Exemple de création/destruction de groupe mieux réalisée par notre méthode. Notre méthode utilise un critère de recouvrement au lieu du critère de forme et arrive donc à détecter correctement la formation puis destruction du groupe. La méthode de Di Lascio se base sur la hauteur des objets et n'arrive pas à créer le groupe (mais crée un nouvel objet).

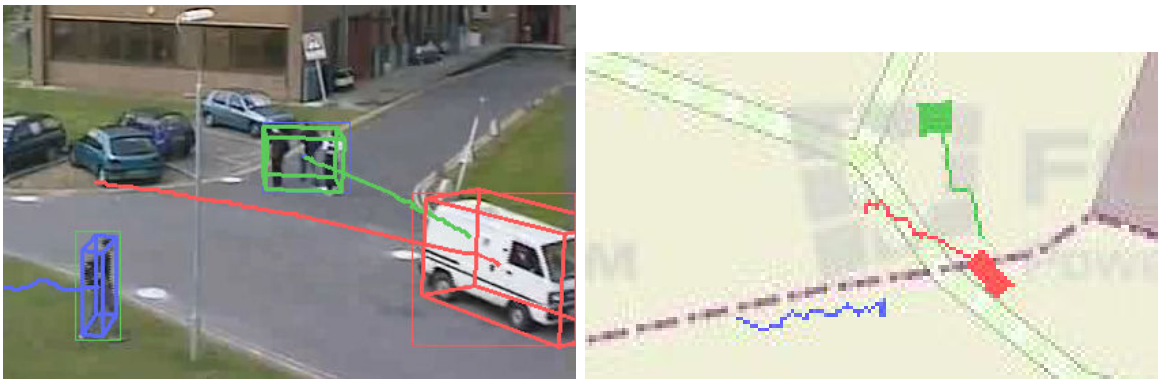
Dans cet exemple, deux piétons se croisent de sorte que la hauteur du blob regroupant les deux piétons soit bien plus grande que celle de chacun pris séparément, ce qui met en défaut le critère de forme utilisé par DiLascio pour la gestion des groupes. En revanche, le critère de recouvrement que nous proposons s'adapte bien à cette situation et permet la création du groupe. De la même manière à la fin du croisement, notre algorithme est capable de détecter la séparation du groupe et de ré-identifier les membres le composant.

Par ailleurs, il est intéressant de constater que, malgré la généralisation à tout type d'objets se déplaçant au sol, notre méthode arrive à améliorer les performances de l'algorithme initial. Nous illustrons qualitativement cette généralisation à la figure 5.11. Sur cette figure nous pouvons observer que notre algorithme s'adapte à n'importe quel type d'objets réel ou synthétique comme, par exemple, des voitures, des camionnettes, ou même une tondeuse.

FIGURE 5.11 Exemples de généralisation à d'autres types d'objets se déplaçant au sol.

À noter cependant que cette généralisation au travers de cuboïdes 3d a un coût d'exécution non négligeable : alors que l'algorithme original ne requiert que 16 ms en moyenne par image, ce temps peut aller jusqu'à 250 ms pour notre algorithme (temps variable selon le nombre d'objets et la résolution des images) à cause l'estimation des cuboïdes.

Grâce à la modélisation de la scène présentée au chapitre 3 et à l'estimation des cuboïdes 3d, il est possible d'offrir aux opérateurs une vue de la scène enrichie par les cuboïdes et trajectoires estimés des objets suivis comme nous le montrons à la figure 5.11. Nous pouvons également changer de point de vue et adopter une vue de dessus comme montré à la figure 5.12.



(a) Vue caméra avec cuboïdes et trajectoires des centroïdes

(b) Vue plan OpenStreetMap avec cuboïdes et trajectoires 3d

FIGURE 5.12 Exemple de plan "OpenStreetMap" enrichi par les trajectoires et les cuboïdes des objets suivis.

En revanche, comme nous travaillons avec une seule caméra, l'ambiguïté de profondeur persiste : il n'est pas rare d'observer des estimations erronées de cuboïdes mais dont les projections 2d sur l'image caméra correspondent bien à des silhouettes d'objets. La sévérité de l'ambiguïté de profondeur dépend principalement de l'angle de "tilt" de

la caméra : pour une caméra regardant à l'horizon, il est très difficile d'estimer correctement la profondeur d'un objet, alors que pour une caméra haute regardant vers le sol, l'estimation de la profondeur n'est plus problématique.

Il est à noter que notre méthode n'est pas capable d'améliorer les cas où les objets allant former un groupe sont de tailles très inégales (exemple à la figure 5.13). Dans ce cas, l'objet de dimension prépondérante risque d'être associé dès la première passe (le petit objet étant en fait négligeable), ce qui empêche la création du groupe.

(a) frame #1811 (b) frame #1812

FIGURE 5.13 Cas limite où des objets de tailles très inégales fusionnent : la création du groupe n'a pas été détectée et le piéton est "absorbé" par le camion. Le cuboïde du camion n'englobe pas tout de suite le piéton du fait du lissage opéré par le filtre de Kalman sur les dimensions du cuboïde.

5.6 Conclusion

Dans ce chapitre nous avons décrit l'algorithme de suivi multi-objets que nous proposons afin d'obtenir les trajectoires des objets mobiles. Cet algorithme étend les travaux de Di Lascio et al. (2013) en généralisant le suivi à tout objet se déplaçant au sol, en améliorant la détection de création/destruction des groupes et en intégrant le contexte spatial 3d de la scène.

Nous avons évalué nos contributions sur des séquences synthétiques et réelles : notre algorithme démontre des performances au moins égales, voire légèrement meilleures, à celles de l'algorithme original tout en généralisant le suivi à n'importe quel type d'objet se déplaçant au sol.

Cependant, la qualité du suivi est fortement conditionnée par la qualité des détections

sur lequel il s'appuie. Nous avons déjà présenté au chapitre 4 des éléments permettant de supprimer les ombres du masque de mouvement afin d'en améliorer la qualité. Cependant la réponse que nous avons proposée souffre notamment du manque de cohérence temporelle. C'est pourquoi, à la suite de ce travail sur le suivi, nous souhaitons réviser la prise en compte des ombres mais aussi, plus généralement, réexaminer la question de l'amélioration de la segmentation du mouvement connaissant les objets suivis dans la scène. C'est ce "rebouclage" (ou rétro action du module de suivi sur le module de détection) que nous allons présenter dans le chapitre suivant.

Chapitre 6

Améliorations des détections par rétro-action du module de suivi

Dans le chapitre précédent, nous avons développé un module de suivi d'objets utilisant des détections issues d'un modèle de fond. L'objet de ce chapitre est de réinvestir la connaissance des objets suivis dans le modèle de fond afin d'améliorer la qualité des détections. Ainsi, nous présentons trois types de rétro-actions du module de suivi sur le module de détection : premièrement, nous proposons d'utiliser la connaissance des positions et formes des objets suivis afin d'adapter localement les paramètres du modèle de fond aux pixels appartenant à des objets pour détecter ceux-ci. Deuxièmement, nous ajoutons un traitement optionnel permettant d'intégrer subitement un objet dans le fond. Ceci permet, par exemple, de basculer rapidement dans le fond une voiture venant de se garer. Troisièmement, nous intégrons les travaux de prédictions des ombres dans l'estimation des cuboïdes des objets suivis et proposons de filtrer ces ombres dans le masque de mouvement. Nous présentons dans ce qui suit ces diverses améliorations.

6.1 Rappels sur l'algorithme ViBe original

L'algorithme de soustraction de fond ViBe (Barnich and Van Droogenbroeck, 2011) s'articule en 3 étapes principales et utilise 4 paramètres globaux (voir tableau 6.1) :

• **L'initialisation du modèle** lors de laquelle, pour chaque pixel, l'apprentissage initial des N valeurs de fond observables est réalisé. À l'inverse de la plupart des algorithmes courants de soustraction de fond, cette étape est réalisée à partir d'une seule image. La procédure d'initialisation d'un pixel est donnée dans le listing 4.

• **La classification binaire objet/fond** détermine si un pixel donné appartient au fond, ou non. Lors de cette étape, la valeur courante d'un pixel est comparée à un

Algorithme 4 : Procédure d'initialisation du modèle de fond d'un pixel

Input : pixel (x, y) à initialiser, collection $(samples)$ de N échantillons au pixel considéré, image rgb courante (rgb).
Output : Les N échantillons du pixel (x, y) sont initialisés.
Procedure `InitBackgroundModel(`x,y,$samples$,N,rgb`)`

```
for i ← 1 to N do
    (x_n, y_n) ← GetRandomNeighbor(x,y);
    samples(i) ← rgb(x_n, y_n);
```

ensemble de N valeurs précédemment observées à ce pixel. Si la valeur courante est en accord (c'est à dire à une distance $< R$) avec au moins $\sharp_{min}$ valeurs, le pixel est considéré comme appartenant au fond. Sinon, il appartient à un objet. La procédure est donnée dans le listing 5.

Algorithme 5 : Algorithme de classification d'un pixel

Input : Couleur du pixel à classer (pix_color), collection $(samples)$ de N échantillons au pixel considéré, distance maximale admissible pour considérer deux couleurs proches (R), seuil de classification $(\sharp_{min})$.
Output : true si le pixel fait parti du fond, false sinon.
Function `IsBackground(`$pix\_color, samples, N, R, \sharp_{min}$`)`

```
count ← 0;
for i ← 1 to N do
    if Dist(pix_color, samples(i)) < R then
        count ← count + 1;
        if count ≥ ♯_min then
            /* Le pixel appartient au fond                    */
            return true;
/* Le pixel n'appartient pas au fond                          */
return false;
```

• **La mise à jour du modèle** (dans le cas où le pixel considéré appartient au fond) permet d'adapter progressivement les changements d'apparence que peut subir le fond au cours du temps. Cette mise à jour est réalisée à la fois pour le modèle du pixel considéré, mais aussi pour un voisin de celui-ci choisi aléatoirement. De plus, ces mises à jour du pixel considéré et du pixel voisin sont conditionnelles, chacune avec une probabilité $\frac{1}{\phi}$. La procédure est donnée dans le listing 6.

Algorithme 6 : Procédure de mise à jour d'un pixel

Input : Pixel courant (x, y), modèle de fond (bgm), masque d'avant-plan (fgm), image RGB courante (rgb), nombre (N) d'échantillons par pixel, distance maximale admissible pour considérer deux couleurs proches (R), seuil de classification $(\sharp_{min})$, paramètre contrôlant la "fréquence" de mise à jour du modèle de fond (ϕ).

Output : Le modèle de fond (bgm) et le masque d'avant-plan (fgm) sont mis à jour.

Procedure `UpdatePixel`$(x, y, bgm, fgm, rgb, N, R, \sharp_{min}, \phi)$

```
if IsBackground(rgb(x,y), bgm(x,y), N, R, ♯_min) then
    /* Background                                          */
    fgm(x,y) ← 0;
    /* Mise à jour du modèle de fond                       */
    if Rnd() mod φ = 0 then
        /* Met à jour le pixel courant                     */
        (x_n, y_n) ← GetRandomNeighbor(x,y);
        i ← GetRandomSampleIndex();
        bgm(x,y)(i) ← rgb(x_n, y_n);
    if Rnd() mod φ = 0 then
        /* Met à jour un pixel voisin choisi aléatoirement */
        (x_n, y_n) ← GetRandomNeighbor(x,y);
        i ← GetRandomSampleIndex();
        bgm(x_n, y_n)(i) ← rgb(x,y);
else
    /* Foreground                                          */
    fgm(x,y) ← 1;
```

TABLE 6.1 Tableau récapitulatif des paramètres de ViBe

Paramètre	Valeur par défaut	Description
N	20	Nombre d'échantillons mémorisés par pixel.
R	20	Distance maximale autorisée pour considérer que la couleur d'un pixel est similaire à la couleur d'un pixel du fond.
$\sharp_{min}$	2	Nombre minimal d'échantillons du fond devant être similaires au pixel courant pour considérer que celui-ci fait partie du fond.
ϕ	16	Espérance de la fréquence de mise à jour du modèle de fond. Une valeur faible provoquera une mise à jour plus fréquente du fond, alors qu'une valeur plus élevée mettra à jour le fond moins souvent.

Cet algorithme présente plusieurs avantages parmi lesquels une initialisation rapide du modèle de fond (une seule image suffit) et une politique aléatoire de mise à jour du modèle de fond permettant, entre autres, une adaptation du modèle de fond sans *a priori* temporel sur la séquence considérée (contrairement aux autres modèles s'appuyant sur une fenêtre d'observation temporelle ou conservant les P dernières valeurs observées). De plus, le fait de mettre également à jour un pixel voisin choisi aléatoirement permet de propager spatialement les valeurs observées dans le fond et ainsi de "rogner" les objets fantômes qui pourraient être présents dans la scène. Enfin, la simplicité de mise en œuvre de l'algorithme, son nombre restreint de paramètres (4) et les bonnes performances affichées (Brutzer et al., 2011) sont autant de critères nous ayant poussé à considérer cet algorithme en particulier.

Nous avons donc identifié trois points d'amélioration de l'algorithme initial que nous allons détailler dans les parties suivantes, à savoir, la prise en compte du contexte local grâce à l'introduction des cartes de paramètres, la possibilité de blocage de la propagation spatiale du modèle de fond et la possibilité de réinitialiser totalement un pixel au lieu de le mettre à jour. Ces modifications de l'algorithme original se veulent compatibles avec l'algorithme initial (*i.e.* il est possible de retrouver le comportement original) et permettent d'adapter le comportement de l'algorithme en fonction de l'application visée, comme nous le verrons par la suite.

6.2 Extensions de l'algorithme de soustraction de fond ViBe

6.2.1 Prise en compte locale du contexte : introduction des cartes de paramètres

ViBe s'appuie sur 4 paramètres globaux que nous avons présentés dans la table 6.1. Comme indiqué, ces paramètres sont globaux et ne tiennent pas compte de la connaissance des objets suivis. Nous proposons d'introduire des cartes de paramètres dont les valeurs, pour chaque pixel, peuvent être modifiées : nous permettons ainsi une adaptation locale de ces paramètres par un processus de plus haut niveau, typiquement un module ayant la connaissance des objets suivis. Plus précisément, nous permettons de spécifier pour chaque pixel (x, y) les valeurs $R(x, y)$, $\sharp_{min}(x, y)$ et $\phi(x, y)$ à utiliser. Nous appelons ainsi "carte du paramètre X" l'ensemble des valeurs du paramètre X sur l'ensemble des pixels (x, y) de l'image. Pour des raisons de simplicité, nous avons laissé invariant le paramètre N qui régit le nombre d'échantillons à mémoriser par pixel.

Grâce à ces cartes de paramètres, nous pouvons ainsi favoriser la détection des objets dans les zones occupées par des objets suivis, en faisant varier les valeurs de R et $\sharp_{min}$ et en accélérant (resp. ralentissant) la fréquence de mise à jour du fond (ϕ) dans les zones occupées par du fond (resp. des objets suivis).

À l'inverse, si l'on souhaite revenir au comportement original de l'algorithme, il suffira de spécifier la valeur par défaut du paramètre sur l'ensemble de la carte.

6.2.2 Blocage du rognage sur les objets

Nous avons mentionné précédemment que ViBe incorporait un mécanisme de propagation spatiale du modèle de fond en mettant à jour non seulement le modèle de fond du pixel courant, mais aussi celui d'un voisin proche choisi aléatoirement. Ce mécanisme est désirable afin d'améliorer la robustesse de l'algorithme, mais peut cependant contribuer à la disparition d'objets d'intérêt, notamment ceux qui seraient stationnaires pour une courte durée. Nous proposons donc de bloquer ce mécanisme de propagation spatiale dans les zones contenant des objets suivis en introduisant une carte de paramètre $m_{update\_neighbor}(x, y)$ qui spécifie si la mise à jour du pixel voisin au pixel (x,y) doit être faite ou non : si $m_{update\_neighbor}(x, y) = 0$, la mise à jour du voisin n'est pas faite, sinon la mise à jour est réalisée comme précédemment.

6.2.3 Réinitialisation d'un pixel

Enfin, nous avons introduit également la possibilité de réinitialiser un pixel (x, y) donné en introduisant une carte de paramètre $m_{reset}(x, y)$ qui détermine si le pixel doit être réinitialisé ($m_{reset}(x, y) \neq 0$) ou non ($m_{reset}(x, y) = 0$). Cette réinitialisation localisée d'un pixel remplace les procédures de classification et de mise à jour et ainsi, permet de basculer instantanément un pixel dans le fond. Ce mécanisme peut être utile notamment dans le cas de voitures se garant dans un parking. En effet, dans ce cas, il suffira de réinitialiser les pixels correspondant au véhicule s'étant garé pour qu'il intègre instantanément le fond. Ainsi, ce véhicule ne sera plus détecté. Cela simplifie la détection des autres objets se déplaçant dans le parking ainsi que leur suivi en réduisant les risques d'occultation.

6.2.4 L'algorithme ViBe étendu

Nous donnons la procédure de traitement d'un pixel selon l'algorithme ViBe étendu (algorithme 7). Les différences avec l'algorithme original ont été mises en évidence. Les procédures identiques à celles de l'algorithme original n'ont pas été rappelées.

6.2.5 Génération des cartes de paramètres

Nous venons de présenter différentes extensions de l'algorithme ViBe, qui permettent de "piloter" le comportement de celui-ci à partir de cartes de paramètres. L'intérêt de cette approche est de pouvoir ainsi spécifier le comportement de l'algorithme en générant des cartes de paramètres adaptées. Cette génération externe des cartes de paramètres peut ainsi, par exemple, tirer parti de la connaissance des objets suivis.

C'est ce que nous proposons dans l'architecture présentée à la figure 6.1.

Les cartes de paramètres sont alors générées selon les besoins suivants :

- **Détection améliorée des objets suivis :** il s'agit dans ce cas de choisir des paramètres différents de détection selon que le pixel considéré correspond à un objet suivi par le tracker ou non. Nous utilisons alors l'équation suivante pour générer les cartes des paramètres R, $\sharp_{min}$, ϕ et $m_{update\_neighbor}$:

$$(R, \sharp_{min}, \phi, m_{update\_neighbor}) = \begin{cases} (R^{obj}, \sharp_{min}^{obj}, \phi^{obj}, 0) & \text{si le pixel appartient à un objet suivi} \\ (R^{bg}, \sharp_{min}^{bg}, \phi^{bg}, 1) & \text{sinon} \end{cases} \quad (6.1)$$

Le choix des valeurs pour ces paramètres est justifié dans la section 6.4.1. Nous illustrons

Algorithme 7 : Algorithme ViBe étendu (pour un pixel)

Input : Pixel courant (x, y), modèle de fond (bgm), masque d'avant-plan (fgm), image RGB courante (rgb), nombre (N) d'échantillons par pixel, carte (R) de distance maximale admissible pour considérer deux couleurs proches, carte $(\sharp_{min})$ du seuil de classification, carte (ϕ) du paramètre contrôlant la "fréquence" de mise à jour du modèle de fond, carte $(m_{update\_neighbor})$ contrôlant la mise à jour du pixel voisin, carte (m_{reset}) de réinitialisation du fond.

Output : Le modèle de fond (bgm) et le masque d'avant-plan (fgm) sont mis à jour.

Procedure

`ProcessPixel`$(x, y, bgm, fgm, rgb, N, R, \sharp_{min}, \phi, m_{update\_neighbor}, m_{reset})$

```
/* Réinitialise le pixel                                              */
if m_reset(x,y) ≠ 0 then
    InitBackgroundModel(x,y,bgm(x,y),N,rgb);
    fgm(x,y) ← 0;
    return ;
/* Met à jour le pixel                                                */
if IsBackground(rgb(x,y), bgm(x,y), N, R(x,y), ♯_min(x,y)) then
    /* Background                                                     */
    fgm(x,y) ← 0;
    if Rnd() mod φ(x,y) = 0 then
        /* Met à jour le pixel courant                                */
        (x_n, y_n) ← GetRandomNeighbor(x,y);
        i ← GetRandomSampleIndex();
        bgm(x,y)(i) ← rgb(x_n, y_n);
    if (m_update_neighbor(x,y) ≠ 0) ∧ (Rnd() mod φ(x,y) = 0) then
        /* Met à jour un pixel voisin choisi aléatoirement            */
        (x_n, y_n) ← GetRandomNeighbor(x,y);
        i ← GetRandomSampleIndex();
        bgm(x_n, y_n)(i) ← rgb(x,y);
else
    /* Foreground                                                     */
    fgm(x,y) ← 1;
```

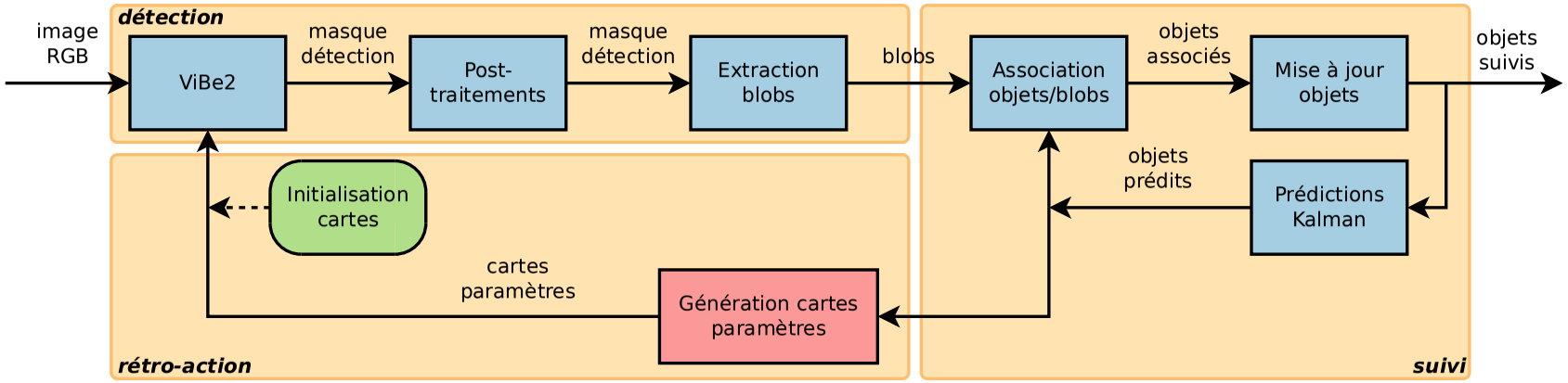


FIGURE 6.1 Architecture du pipeline avec rétro-action réalisant la détection et le suivi d'objets. La chaîne principale des traitements est montrée en bleu et la boucle de rétro-action intégrant la connaissance des objets suivis est montrée en rouge. Une étape d'initialisation des cartes de paramètres (en vert) est également nécessaire.

cette génération "binaire" de cartes de paramètres à la figure 6.2b où les zones en blanc correspondent aux zones où il y a des objets suivis (les paramètres prennent donc leur valeur X^{obj}) et les zones en noir correspondent aux zones il n'y a pas d'objets suivis (les paramètres prennent leur valeur X^{bg}).

• **Oubli des véhicules stationnés :** nous générons la carte du paramètre m_{reset} de manière à intégrer les objets stationnés depuis plus de T_s frames dans une zone de parking. Pour détecter un tel comportement, nous incrémentons un compteur pour chaque objet lorsque celui-ci est immobile et est situé dans une zone de parking (cette zone est spécifiée par l'utilisateur via un masque). Si l'objet se remet à bouger, le compteur est remis à zéro. Lorsque ce compteur atteint T_s, l'objet a été immobile pendant plus de T_s frames et peut donc être intégré au fond. L'équation régissant la génération de la carte de paramètre est donc :

$$m_{reset} = \begin{cases} 1 & \text{si le pixel appartient à un objet stationné depuis plus de } T_s \text{ frames} \\ & \text{dans une zone de parking.} \\ 0 & \text{sinon} \end{cases} \tag{6.2}$$

Nous illustrons une telle carte de paramètre à la figure 6.2c où la voiture (cuboïde jaune dans l'image 6.2a) est candidate pour être intégrée dans le fond.

• **Initialisation des cartes :** les cartes de paramètres sont initialisées en considérant qu'il n'y a pas d'objets suivis. Cela revient donc à générer des cartes de paramètres uniformes avec les valeurs suivantes [1] :

$$(R, \sharp_{min}, \phi, m_{update\_neighbor}, m_{reset}) = (R^{bg}, \sharp^{bg}_{min}, \phi^{bg}, 1, 1) \tag{6.3}$$

Il conviendra également de noter que la démarche présentée dans le cadre du modèle de fond ViBe avec des modules spécifiques permettant d'en améliorer les performances est relativement générale. Il est tout à fait envisageable d'adapter cette idée à d'autres modèles de fond (par exemple, pour le modèle basé sur les mélanges de gaussiennes, on pourra adapter le paramètre de mise à jour des poids et/ou le seuil de sélection des distributions). À noter également que le cas présenté intègre les voitures garées dans le fond, mais il est tout à fait envisageable de ne pas souhaiter ce comportement et donc de générer les cartes de paramètres en conséquence.

1. En réalité, seule la valeur $m_{reset} = 1$ est nécessaire, car dans ce cas, on réinitialise tous les pixels et donc les autres paramètres n'interviennent pas.

(a) image courante (b) carte de paramètre permettant d'améliorer la détection des objets suivis (c) carte de paramètre de réinitialisation du fond

FIGURE 6.2 Exemples de cartes de paramètres.

6.3 Suppression des ombres des objets suivis

Au chapitre 4, nous avons présenté une première méthode de suppression des ombres sans cohérence temporelle (*ie* sans connaissance des objets suivis). Maintenant que nous disposons d'un module de suivi, nous souhaitons réévaluer la suppression des ombres des objets mobiles du masque de mouvement. Nous proposons, d'une part, d'intégrer la possibilité qu'une ombre soit présente dans le masque de mouvement et donc d'adapter l'inférence des cuboïdes à ces cas. D'autre part, nous intégrons la suppression des ombres dans la boucle de rétro-action via la génération d'une carte d'ombres et un module de suppression à partir de cette carte.

6.3.1 Révision de l'inférence des cuboïdes

La procédure d'inférence des cuboïdes des objets suivis présentée dans la section 5.2.1 suppose qu'il n'y ait pas d'ombre dans le masque de mouvement. Nous souhaitons supprimer cette contrainte en permettant la présence d'une ombre portée attachée à l'objet suivi. L'idée permettant de parvenir à notre but est d'ajouter une variable booléenne de présence ou d'absence d'une ombre portée dans l'énergie à optimiser (équation 6.4) et d'intégrer dans le rendu des silhouettes la projection de leurs ombres (prédites grâce aux travaux présentés au chapitre 4) le cas échéant.

$$E(x_c, y_c, w_c, d_c, h_c, \theta_z, has\_shadow, m) = \\ ||(r(x_c, y_c, w_c, d_c, h_c, \theta_z, has\_shadow) - m).I_{R1 \cup R2}||_{L2} \quad (6.4)$$

Ceci permet de déterminer la présence ou l'absence d'ombre via l'optimisation de l'énergie et d'avoir une estimation de la position et des dimensions du cuboïde en tenant compte de l'éventuelle ombre portée.

L'ajout de cette variable booléenne augmente la taille de l'espace à parcourir pour la recherche d'un minimum, c'est pourquoi nous avons également adapté la procédure d'optimisation en conséquence afin de mieux guider l'optimisation de cette nouvelle énergie (voir figure 6.3).

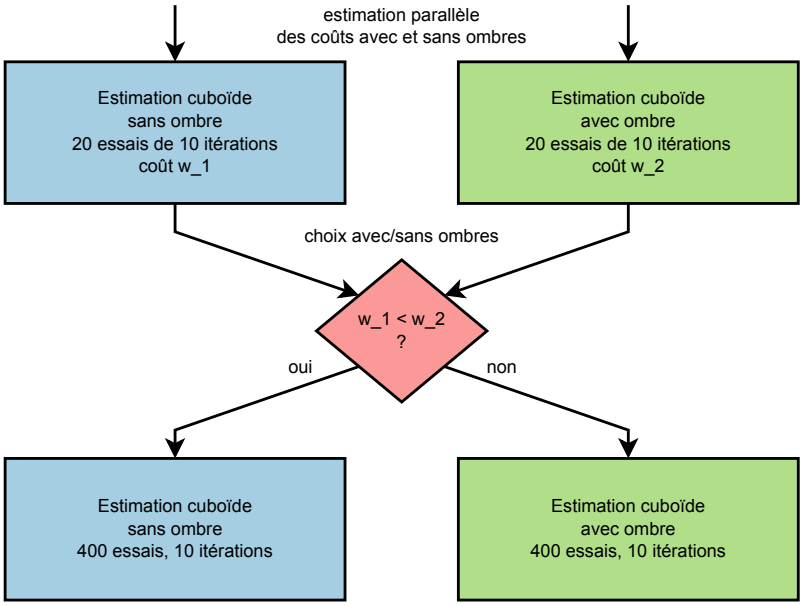


FIGURE 6.3 Procédure d'inférence des cuboïdes des objets suivis tenant compte de la présence ou non d'ombres portées.

L'idée est de tester parallèlement les deux hypothèses (avec ou sans ombres) et de retenir celle qui donne l'énergie la plus basse. De plus, pour éviter de calculer inutilement la totalité des itérations pour chacune des deux hypothèses, le calcul parallèle des deux solutions n'est effectué que sur un nombre restreint d'itérations. À l'issue de ces itérations préliminaires, le choix d'une hypothèse de travail est effectué – il y a une ombre ou il n'y en a pas – et le reste des itérations à effectuer est réalisé en imposant cette hypothèse.

Ainsi, cette démarche permet d'étendre l'estimation des cuboïdes en prenant en compte la présence éventuelle d'une ombre portée. Par ailleurs, à l'inverse de la première méthode proposée (section 4.6) qui faisait l'hypothèse de cylindres de taille fixe, cette démarche s'adapte automatiquement à la taille des objets suivis.

Maintenant que nous avons vu comment estimer les ombres portées éventuelles des objets suivis, voyons comment nous pouvons les supprimer du masque de mouvement.

(a) image courante (b) carte d'ombres

FIGURE 6.4 Exemple de carte d'ombres

6.3.2 Suppression des ombres portées des objets suivis

L'estimation des ombres portées des objets suivis réalisée par l'algorithme de suivi est utilisée afin de générer la carte des ombres définie ainsi :

$$\mathcal{O}_p(x,y) = \begin{cases} 1 & \text{si le pixel } (x,y) \text{ appartient à l'ombre prédite} \\ 0 & \text{sinon} \end{cases} \tag{6.5}$$

Une illustration d'une telle carte est donnée à la figure 6.4.

Nous réinjectons alors cette carte des ombres portées dans le pipeline des traitements (voir figure 6.5) de manière à ce que le module de suppression des ombres, placé en sortie du modèle de fond, puisse réaliser le filtrage des ombres dans les zones définies par le masque d'ombres (*i.e.* $\mathcal{O}_p(x,y) = 1$) ;

Avant de supprimer les pixels d'ombre, nous vérifions que les pixels considérés ont bien des propriétés colorimétriques compatibles avec la présence d'ombre, c'est-à-dire qu'ils présentent une atténuation d'intensité, tout en conservant la couleur qu'ils auraient s'ils n'étaient pas ombrés. Ce test colorimétrique proposé par Cucchiara et al. (2003) se formule dans l'espace HSV de la manière suivante :

$$\mathcal{O}_c(x,y) = \begin{cases} 1 & \text{si } \alpha_V \leq \frac{I(x,y)^V}{B(x,y)^V} \leq \beta_V \wedge \\ & |I(x,y)^S - B(x,y)^S| \leq \tau_S \wedge \\ & min(|I(x,y)^H - B(x,y)^H|, 360 - |I(x,y)^H - B(x,y)^H|) \leq \tau_H \\ 0 & \text{sinon} \end{cases} \tag{6.6}$$

avec $I(x,y)^C$ la valeur du canal C de l'image courante I au pixel (x,y) ; $B(x,y)^C$ la valeur du canal C de l'image de fond B au pixel (x,y) ; α_V et β_V des seuils (compris

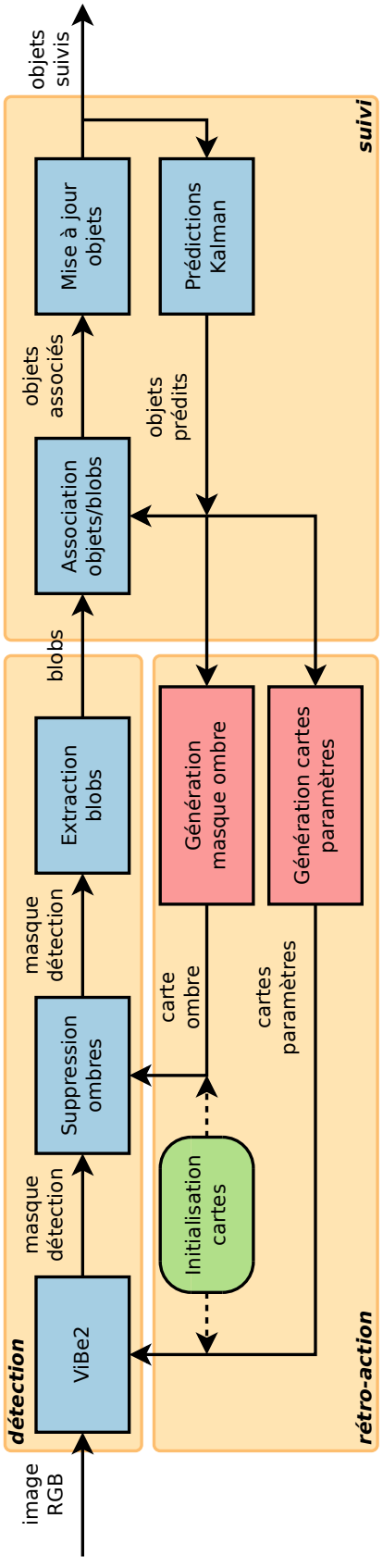


FIGURE 6.5 Architecture du pipeline avec rétro-action réalisant la suppression des ombres portées des objets suivis. La chaîne principale des traitements est montrée en bleu et les boucles de rétro-action intégrant la connaissance des objets suivis sont montrées en rouge. Une étape d'initialisation des cartes de paramètres (en vert) est également nécessaire.

entre 0 et 1) limitant l'atténuation relative de valeur entre l'image courante et celle de fond ; τ_S un seuil limitant la différence entre les saturations de l'image courante ; et τ_H un seuil limitant la différence circulaire entre les valeurs de teinte de l'image courante et de l'image de fond. Si ce test colorimétrique est vérifié ($\mathcal{O}_c = 1$) l'ombre est supprimée du masque de mouvement, sinon la suppression n'est pas effectuée.

Par cette démarche, nous combinons à la fois un critère géométrique issu de la prédiction des ombres des objets suivis et un critère colorimétrique vérifiant l'atténuation d'intensité à chromaticité constante. Cela permet d'affiner la prédiction purement géométrique mais grossière des ombres. Pour le test colorimétrique nous utilisons volontairement des seuils élevés afin de n'empêcher que les suppressions où les pixels ont des couleurs vraiment différentes de ce qui est attendu pour une ombre. En d'autres termes nous privilégions la prédiction géométrique des ombres, tout en l'affinant grâce au critère colorimétrique.

6.4 Évaluation

Dans cette section, nous souhaitons montrer l'incrément obtenu grâce aux rétro-actions présentées dans ce chapitre. Nous évaluons en particulier l'amélioration de la segmentation, la qualité de la suppression des ombres et l'apport pour le suivi des objets.

6.4.1 Choix des paramètres

Pour choisir les paramètres à utiliser lors de la génération des cartes de paramètres pour modèle de fond ViBe, nous avons procédé ainsi : pour le paramètre N définissant le nombre d'échantillons à mémoriser par pixel, nous avons conservé la valeur originale. Dans l'article original Barnich and Van Droogenbroeck (2011), cette valeur est justifiée par le fait que les performances de segmentation augmentent avec N et stagnent au-delà de 20 échantillons. Il est donc préférable, afin de réduire l'encombrement mémoire et le temps de calcul, de prendre la valeur marquant le début du plateau de performance, c'est-à-dire 20 échantillons.

Pour les autres paramètres, nous avons réduit l'espace de recherche des paramètres optimaux en fixant les valeurs de $(R^{bg}, \sharp^{bg}_{min}, \phi^{obj})$ aux valeurs de l'algorithme original c'est-à-dire (20,2,16). Nous avons ensuite cherché les valeurs optimales de R^{obj}, $\sharp^{obj}_{min}$ et ϕ^{bg} en les faisant varier séparément (les autres valeurs étant fixées à leur valeur par défaut). Nous obtenons ainsi les courbes de performances données à la figure (6.6). Ces premiers résultats nous ont poussés à tester également les cas R^{obj} et ϕ^{bg} fixés respectivement

à 35 et 2 avec $\sharp_{min}^{obj}$ variable. Les meilleures performances sont obtenues pour la valeur $\sharp_{min}^{obj} = 4$.

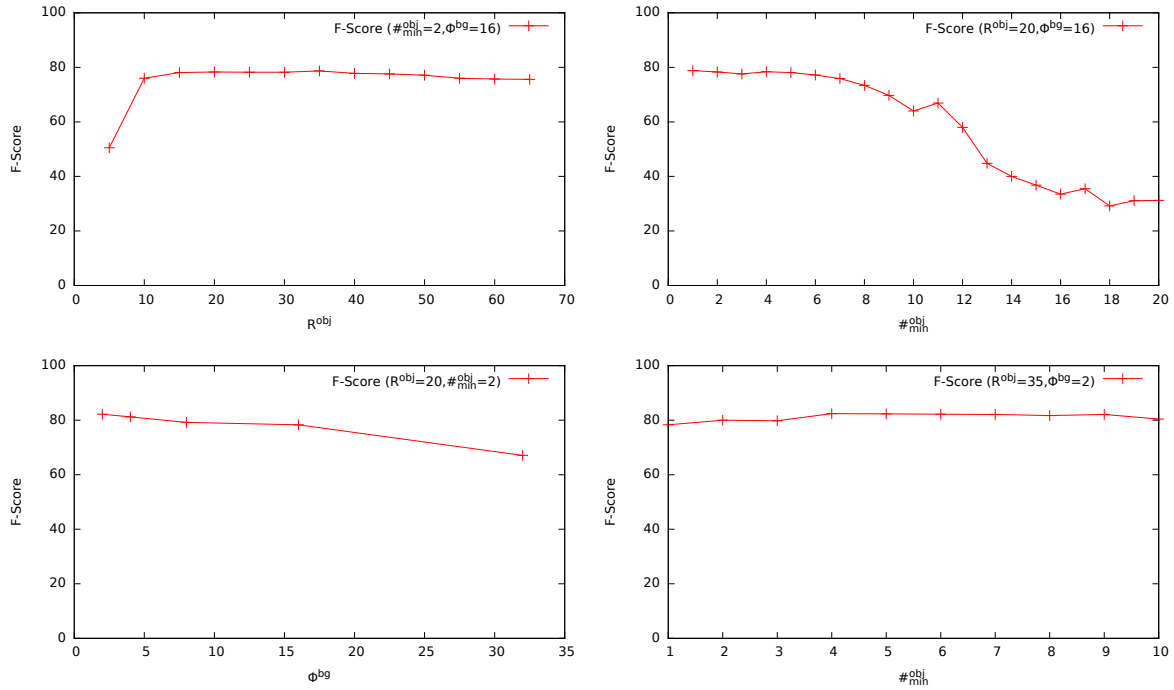


FIGURE 6.6 Courbes de performance de segmentation obtenues pour différentes valeurs de paramètres.

C'est pourquoi nous utilisons dorénavant pour l'ensemble des tests les valeurs suivantes : $(R^{obj}, \sharp_{min}^{obj}, \phi^{obj}) = (35, 4, 16)$, $(R^{bg}, \sharp_{min}^{bg}, \phi^{bg}) = (20, 2, 2)$.

Concernant les autres paramètres intervenant dans la génération des cartes de paramètres, nous avons fixé le temps d'immobilité en zone de parking à $T_s = 30$ et nous utilisons les valeurs suivantes pour le test colorimétrique de confirmation de suppression des ombres : $\alpha_V = 0.3, \beta_V = 0.9, \tau_S = 40, \tau_H = 68$.

6.4.2 Amélioration de la segmentation

Afin de montrer qualitativement et quantitativement l'incrément obtenu sur la qualité de la segmentation nous avons annoté quelques images (une quinzaine) issues de la séquence PETS01. Pour cette annotation, nous avons considéré qu'un véhicule une fois garé dans une zone de parking doit être inclus dans le fond puisqu'il n'est plus en mouvement.

Nous avons utilisé les métriques d'évaluation de segmentation telles que la "Précision", le "Rappel" et le "F-Score" (dont les équations ont déjà été données en 4.14). Nous avons

choisi cette séquence car celle-ci met en jeu à la fois des piétons et des véhicules (voitures, camionnettes) dont le comportement permet d'illustrer l'apport de la rétro-action.

En particulier, dans cette scène (voir figure 6.7), outre les quelques groupes de piétons qui traversent la scène, nous avons une voiture (cercle rouge) qui vient se garer dans la zone de parking (pointillés jaunes) et une camionnette (cercle vert) qui vient s'arrêter longuement dans une zone hors parking puis repartir.

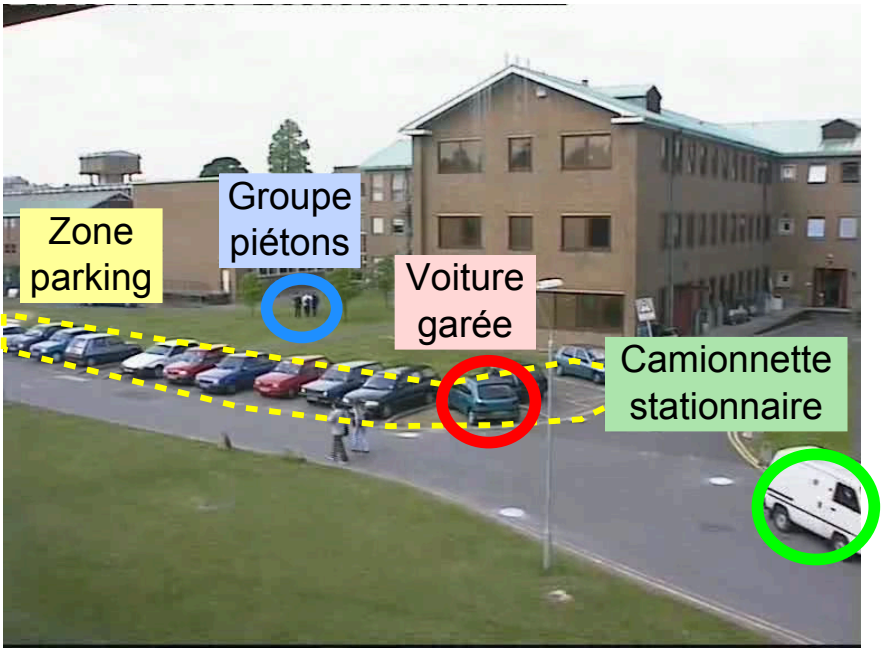


FIGURE 6.7 Description de quelques éléments de la scène PETS01. Dans cette scène, la voiture (cercle rouge) vient se garer dans la zone de parking (pointillés jaunes). Puis la camionnette blanche (cercle vert) vient s'arrêter un long moment sur le bord de l'image. Le groupe de piétons (cercle bleu) traverse la scène.

Le tableau 6.2 montre les scores moyens obtenus sur cette séquence.

TABLE 6.2 Résultats de segmentation obtenus sans et avec rétro-actions

Méthode	Précision	Rappel	F-Score
ViBe	0.653	0.876	0.747
ViBe + rétro-actions	**0.773**	**0.888**	**0.824**

Nous donnons également quelques illustrations correspondantes à la figure 6.8.

Les mesures quantitatives indiquent une amélioration de la qualité de la segmentation (+ 0.08 pour le F-Score par exemple). Ces améliorations de la segmentation proviennent principalement du basculement immédiat de la voiture se garant (cercle rouge de la

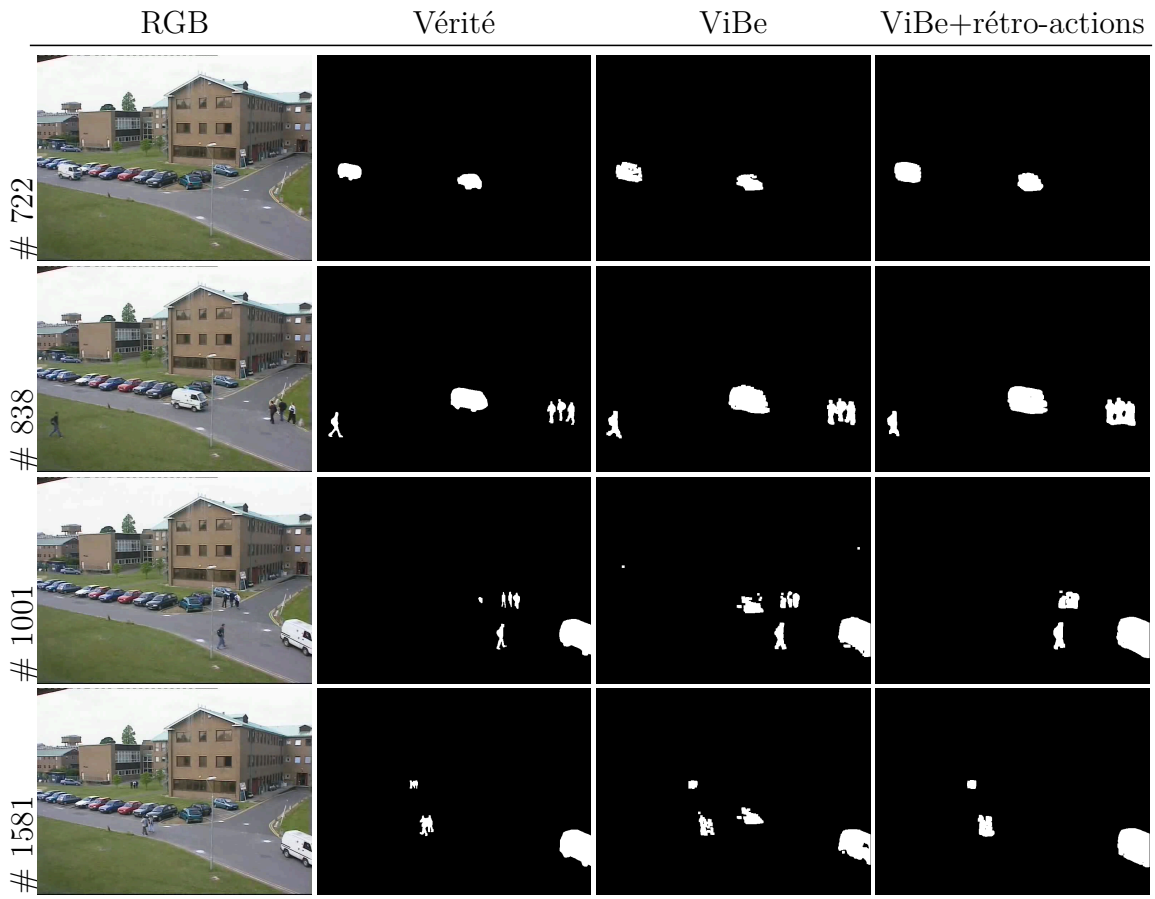


FIGURE 6.8 Exemples de masques de mouvement obtenus avec et sans rétro-actions.

figure 6.7). Dans la version sans retro-action, cette voiture est érodée progressivement et donc persiste jusqu'à la fin de la séquence (voir 3^{e} colonne de la figure 6.8). En revanche, dans la version avec rétro-action, une fois garée, cette voiture est intégrée dans le fond et n'est donc plus présente pour le reste de la séquence (voir 4^{e} colonne).

Notons également que dans le cas où un objet est stationnaire dans une zone hors parking, comme c'est le cas pour la camionnette blanche (cercle vert de la figure 6.7), celui-ci persiste au premier-plan et, grâce au mécanisme de blocage de la propagation spatiale du modèle de fond ViBe dans les zones comportant des objets, n'est pas érodé comme cela se produit sans rétro-action (frame 1581).

De plus grâce à l'augmentation de la sensibilité dans les zones où il y a des objets, les blobs présentent moins de "trous" (camionnette au frame 1581). Toutefois, cette augmentation de la sensibilité de détection dans les zones contenant des objets suivis favorise également la fusion de blobs "proches" comme on peut le voir pour le groupe de piétons (cercle bleu de la figure 6.7) par exemple au frame 838. Ce groupe de piétons

est en effet entré ensemble dans la scène et le module de suivi n'a pas su détecter individuellement chacun : il considère donc qu'il n'y a qu'un seul objet et augmente donc la sensibilité du modèle de fond dans cette zone, d'où la fusion des blobs observée. Nous reviendrons sur ce problème dans la section dédiée à l'évaluation du suivi (section 6.4.4).

6.4.3 Suppression des ombres

Nous souhaitons également montrer l'amélioration de la segmentation dûe à la prise en compte et à la suppression des ombres des objets suivis. Pour ce faire, nous avons repris la séquence PETS01.D3 que nous avions utilisé dans le chapitre 4.

Nous donnons, dans la table 6.3, les scores de "Précision", "Rappel" et "F-Score" moyens obtenus sur la séquence dans diverses configurations afin d'évaluer la contribution de chacun des éléments du système. Ainsi nous rappelons les scores obtenus sur cette séquence par le modèle de fond ViBe seul (a) et avec la méthode de suppression des ombres spécifique aux piétons du chapitre 4 (b). Ces deux cas ne tirent parti d'aucune des rétro-actions présentées dans ce chapitre. Nous considérons également différents cas tirant parti de la rétro-action pilotant le modèle de fond ViBe : sans suppression d'ombres (c), avec la suppression des ombres du chapitre 4 (d), avec la suppression des ombres de ce chapitre sans le test colorimétrique (e) et avec le test colorimétrique (f).

Nous illustrons également ces scores à la figure 6.9.

TABLE 6.3 Résultats de segmentation dans diverses configurations de suppression des ombres

Méthode	Précision	Rappel	F-Score
(a) ViBe seul	0.343	**0.769**	0.425
(b) ViBe + Suppression ombres chap. 4	0.352	0.762	0.432
(c) ViBe + Rétroaction - Pas de suppression d'ombres	0.500	0.722	0.584
(d) ViBe + Rétroaction + Suppression ombres chap. 4	**0.520**	0.722	**0.600**
(e) ViBe + Rétroaction + Suppression ombres sans test colorimétrique	**0.520**	0.726	**0.601**
(f) ViBe + Rétroaction + Suppression ombres avec test colorimétrique	**0.520**	0.720	**0.600**

Nous observons dans ces résultats un gain important de performance grâce à l'ajout de la rétro-action pilotant le modèle de fond. En effet, dans cette vidéo, les conditions d'éclairement sont difficiles et changent brutalement à cause du passage de nuages. Sans

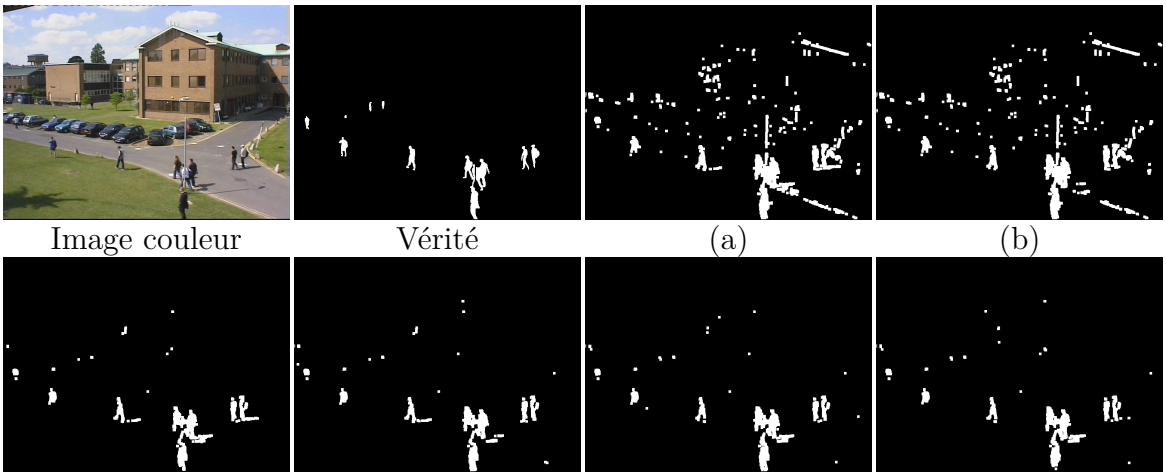


FIGURE 6.9 Exemples de masques de mouvement obtenus avec et sans rétro-actions de suppression des ombres.

rétro-action, le modèle de fond n'est pas capable de s'adapter suffisamment rapidement d'où les faibles performances. En revanche, avec la rétro-action, le modèle de fond s'adapte mieux aux passages des nuages.

Notons également l'incrément obtenu (+0.02) sur le F-Score suite à la suppression des ombres quelque soit la méthode de suppression choisie (lignes 4 à 6). Ceci est remarquable car, alors que la méthode de suppression des ombres du chapitre 4 est spécifique aux piétons, la méthode de suppression des ombres de ce chapitre s'adapte à tout type d'objets (cuboïdes).

Toutefois, du fait de l'assujétissement au suivi des objets, nous n'avons pas pu appliquer notre méthode aux autres vidéos qui avaient été utilisées lors des tests de suppression des ombres du chapitre 4 (BMC.V1, SudEst, PETS09.S2L2). En effet, ces vidéos ne sont pas adaptées à nos conditions à cause du framerate bas et variable (BMC.V1), de la qualité d'acquisition faible (SudEst) ou du fait qu'il y ait une foule de personnes (PETS09.S2L2).

6.4.4 Amélioration du suivi

Pour mettre en avant l'amélioration du suivi, nous proposons de reprendre les expériences de suivi réalisées au chapitre précédent et d'ajouter les rétro-actions présentées dans ce chapitre.

Avec les mêmes notations et conditions qu'au chapitre précédent nous obtenons les résultats présentés dans le tableau 6.4.

TABLE 6.4 Résultats de suivi obtenus sans et avec les rétro-actions

Vidéo	Méthode	MOTA	MOTP	GT	FP	FN	IDs	MT	PT	ML
PETS.S2L1	Rogez	**0.774**	0.617	4644	**114**	**737**	199	**14**	5	0
	Rogez+retro	0.726	**0.650**	4644	157	933	**181**	13	**6**	0
parking1	Rogez	0.729	0.516	15439	200	3754	228	12	**15**	0
	Rogez+retro	**0.753**	**0.616**	15439	**90**	**3525**	**196**	**14**	13	0

Ces résultats appellent à quelques commentaires : sur la vidéo PETS.S2L1, les indicateurs MOTA, FP, FN, et MT ont été dégradés par l'ajout des rétro-actions, alors que les indicateurs MOTP, IDs et PT ont été améliorés. Nous avons souhaité mieux comprendre les causes de cette dégradation inattendue des performances et nous avons identifié une des sources de cette dégradation. Plusieurs personnes entrent dans la scène en étant très proches (collées), notre algorithme les considère alors comme un seul objet. Ainsi pour les métriques d'évaluations, cela compte une omission alors que les deux piétons sont suivis collectivement. Tout au long de son parcours le groupe reste proche, ce qui explique que cette erreur soit visible quantitativement (augmentation du nombre de FP et donc diminution du MOTA). Alors que dans la version sans rétro-action, le modèle de fond arrive, au bout d'un certain temps, à séparer ce groupe de personnes et à les suivre individuellement, dans la version avec rétro-action, cette séparation n'est pas observée car le modèle de fond est encouragé à détecter des objets dans cette zone.

Cette erreur de suivi n'est cependant pas gênante vis-à-vis de l'utilisateur car la trajectoire du groupe suivi reste pertinente.

Pour la vidéo parking1, tous les indicateurs affichent une amélioration. En particuliers, on notera l'amélioration du MOTP et la division par 2 du nombre de faux positifs.

6.4.5 Temps de calculs

Nous donnons le détail des temps de calculs obtenus pour chacune des séquences de ce chapitre dans la table 6.5. Dans cette table nous rappelons également les résolutions de chacune des vidéos. La machine de test utilisée est un ordinateur portable doté d'un Core i7 @ 2,4 GHz et de 3 Go RAM.

Les résultats obtenus montrent que l'étape la plus chronophage est celle du suivi. Ceci s'explique principalement par le processus relativement long d'inférence des cuboïdes à partir du masque de mouvement.

En l'état, les travaux que nous avons proposés ne sont pas strictement utilisables dans un contexte temps réel lorsque la résolution d'analyse ou que le nombre d'ob-

(a) Sans Rétro-action (b) Avec Rétro-action

FIGURE 6.10 Cas défavorable où deux piétons sont suivis collectivement avec rétro-action, et individuellement sans rétro-action.

TABLE 6.5 Temps de calculs moyens et écarts types des différentes étapes pour chacune des vidéos. Les temps sont exprimés en millisecondes.

Séquence (Résolution)	Vibe	Post-Traitements et suppression ombres	Extraction de blobs	Suivi	Générations cartes	Global
PETS01.D3 (768 × 576)	20 ± 1	6 ± 1	2 ± 1	65 ± 40	0 ± 0	**95 ± 40**
PETS01 (768 × 576)	21 ± 3	6 ± 1	2 ± 1	100 ± 66	1 ± 0	**129 ± 66**
PETS09.S2L1 (768 × 576)	19 ± 1	5 ± 1	2 ± 0	201 ± 85	1 ± 0	**228 ± 85**
parking1 (1280 × 960)	53 ± 3	15 ± 1	5 ± 1	133 ± 67	2 ± 0	**208 ± 67**

jets dans la scène sont trop importants. C'est pourquoi, nous préconisons de limiter la résolution d'analyse à une résolution CIF (352 × 288). Cette résolution est en effet couramment employée par les caméras de vidéosurveillance et est compatible avec la contrainte d'exécution en temps réel.

6.5 Conclusion

Au cours de ce chapitre nous avons présenté un ensemble de modifications permettant de réinjecter la connaissance des objets suivis dans les étapes de segmentation. En particulier, nous avons proposé 3 types de rétro-actions du module de suivi sur le module de détection : amélioration de la segmentation, possibilité de basculer instantanément dans le fond un objet statique (voiture garée) et suppression des ombres des objets suivis. Ces rétro-actions s'appuient sur des cartes de paramètres permettant d'adapter localement le comportement du modèle de fond.

La méthode de suppression d'ombre proposée dans ce chapitre permet de lever les contraintes (taille fixe, pas de cohérence temporelle) qui étaient présentes dans la première méthode présentée en section 4.6.

Finalement, nous montrons que ces rétro-actions améliorent à la fois la qualité de la segmentation des objets et la qualité de leur suivi.

Ce chapitre conclut ainsi la présentation des différentes contributions de cette thèse.

Chapitre 7

Conclusion

7.1 Travail réalisé

La détection et le suivi d'objets en vidéosurveillance sont des thématiques récurrentes et difficiles. Nous avons souhaité aborder le sujet en intégrant des informations complémentaires aux images acquises par les caméras. Nous avons en particulier proposé de prendre en compte les données issues des études de prédéploiement des caméras qui, dans le contexte de la vidéosurveillance, sont disponibles et offrent ainsi une opportunité d'améliorer la connaissance de la scène observée. Nous avons proposé une modélisation géométrique 3d géolocalisée de la scène et une modélisation de la caméra pouvant être directement extraites de ces études de prédéploiement. La construction manuelle du modèle de scène pouvant être fastidieuse, nous avons également suggéré l'utilisation de la base de donnée OpenStreetMap afin d'en automatiser la construction.

Profitant de cette modélisation géométrique géolocalisée, nous avons intégré la possibilité de prédire la position du Soleil (source de lumière principale en extérieur) et des ombres portées résultant de son action en fonction de la date et de l'heure courante (Rogez et al., 2013). Cette possibilité de prédire les ombres portées a été mise à profit afin d'améliorer la détection des piétons. Nous avons en effet développé une méthode filtrant les ombres portées des piétons du masque de mouvement. Cependant la méthode proposée reste limitée au cas des piétons, certes largement dominant en vidéosurveillance, et n'exploite aucun mécanisme de cohérence temporelle.

C'est pourquoi nous avons développé, dans un deuxième temps, un algorithme de suivi à partir des travaux réalisés par Di Lascio et al. (2013) pour le suivi de piétons (Rogez et al., 2014). L'originalité de l'algorithme est de formuler l'évolution d'un objet suivi sous la forme d'automate fini, qui rend ainsi possible des traitements adaptés à chaque objet selon son état. De plus, en intégrant un mécanisme de suivi collectif des

objets sous la forme de groupe, celui-ci se montre capable de gérer de nombreux cas d'occultations. Dans le cadre de cette thèse, nous avons généralisé cet algorithme afin qu'il puisse suivre n'importe quel type d'objet se déplaçant au sol. Nous avons également amélioré le mécanisme de création/destruction des groupes et avons intégré le contexte spatial 3d de la scène.

Dans un dernier temps, nous avons revisité les étapes de détection en tirant parti de la connaissance des objets suivis. Cette rétro-action, via des cartes de paramètres, nous a permis notamment de mieux traiter le cas des objets stationnaires qui peuvent être problématiques pour les algorithmes de soustraction de fond. Nous avons également revisité la méthode de suppression des ombres portées grâce à la connaissance des objets suivis.

7.2 Perspectives

Le travail présenté a exploré divers moyens de tirer parti du contexte disponible en vidéosurveillance, cependant nous pensons que certains éléments mériteraient d'être revisités et étendus.

Les études de prédéploiement contiennent généralement plusieurs caméras. Nous avons jusqu'à présent considéré chacune de manière indépendante en travaillant dans le cas mono-caméra. Il serait profitable de proposer une extension multi-caméra. Par exemple, quand un objet sort du champ d'une caméra (état SORTANT), il serait possible non pas de le basculer à l'état SUPPRIMÉ, mais de le mettre en "réserve" (éventuellement dans l'état PERDU ou équivalent) quelque temps pour voir s'il n'apparaitrait pas dans le champ de vision d'une autre caméra, ou s'il n'entrerait pas à nouveau dans le même champ de vision. Ceci pourrait s'appliquer dans le cas de caméras à champ disjoint. Pour les caméras à champ recouvrant, il serait intéressant d'utiliser l'approche des *Probabilistic Occupancy Map* (Fleuret et al., 2008) ou celle de Carr et al. (2012) afin de fiabiliser l'estimation de position 3d qui reste potentiellement ambiguë avec une seule caméra.

En ce qui concerne le paramétrage du système, nous tirons parti des études de prédéploiement. Cependant, il se peut qu'il existe une différence entre les spécifications préconisées dans ces études et l'installation réelle. Pour l'instant une correction manuelle est requise pour corriger ces différences. Il serait judicieux d'intégrer un mécanisme de recalibration permettant de corriger les faibles différences automatiquement. Nous pourrions par exemple formuler cette recalibration comme une optimisation des paramètres de la caméra en utilisant les structures dominantes de la scène (arêtes de bâtiments, marque au sol) comme guide. Il y a en effet un réel enjeu de simplicité de déploiement

qu'il est nécessaire d'aborder avant d'industrialiser une telle solution. Une variante de cette optimisation pourrait permettre également d'estimer la hauteur des bâtiments environnants afin de compléter les données manquantes d'OpenStreetMap. Cette variante pourrait s'appuyer, par exemple, sur les contours des ombres projetées par les bâtiments si ceux-ci ne sont pas entièrement visibles. On pourra également réinvestir la connaissance de la position du soleil dans le logiciel utilisé dans le prédéploiement des caméras. En effet, ceci permettrait de prévenir les risques d'éblouissement directs ou indirects (reflets) d'une caméra par le soleil en notifiant l'utilisateur que sa caméra peut être éblouie par le soleil dans certaines conditions. Cette démarche pourrait également être étendue à d'autres sources de lumières (projecteurs, néons) pour peu qu'elles soient renseignées dans les études de prédéploiement.

Nous avons travaillé principalement avec des hypothèses de caméra fixe et avons utilisé en conséquence la soustraction de fond afin de réaliser les détections. Ce choix était motivé par des containtes d'exécution en temps réel et par le contexte de détection d'intrusion où il ne fallait pas faire trop d'hypothèses sur le type de détection afin d'éviter les omissions dues à la spécificité du détecteur (cas d'une intrusion en rampant par exemple). Ce choix pourrait être remis en cause au profit d'un ou plusieurs détecteurs spécifiques pour les piétons ou les véhicules quand cela est adapté à la situation (relecture *a posteriori*, surveillance de foules). Dans ce cadre il serait intéressant de tester les approches de détections par parties structurées à l'aide d'un modèle géométrique proposées par Xiao et al. (2012) ou Fidler et al. (2012). Ceci permettrait à la fois d'inférer la géométrie des objets détectés et de s'affranchir des problèmes liés aux ombres ou de détections fusionnées. Notons que le passage à des détecteurs spécifiques permettrait également de pouvoir gérer les foules denses de personnes mieux qu'un algorithme de soustraction de fond.

Nous pensons également qu'il serait bénéfique de présenter une vue reconstruite de la scène 3d augmentée des détections et des trajectoires en temps réel. Ceci apporterait un confort d'utilisation appréciable pour les opérateurs. En effet, pour l'instant le modèle 3d ne montre que des polygones non texturés qui peuvent limiter la compréhension de la scène. En reprojetant la (ou les) image(s) acquise(s) par les caméras, nous pourrions proposer une vue texturée bien plus intuitive pour les utilisateurs.

Bibliographie

Andriyenko, A., Roth, S., and Schindler, K. (2011). An analytical formulation of global occlusion reasoning for multi-target tracking. In *Proceedings of the 2011 International Conference on Computer Vision Workshops (ICCVW)*, pages 1839–1846. IEEE.

Andriyenko, A. and Schindler, K. (2010). Globally optimal multi-target tracking on a hexagonal lattice. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, volume 6311 of *Lecture Notes in Computer Science*, pages 466–479. Springer Berlin Heidelberg.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., and Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV)*, 92(1) :1–31.

Bardet, F., Chateau, T., and Ramadasan, D. (2009). Illumination aware mcmc particle filter for long-term outdoor multi-object simultaneous tracking and classification. In *Proceedings of the 12th International Conference on Computer Vision (ICCV)*, pages 1623–1630. IEEE.

Barnich, O. and Van Droogenbroeck, M. (2011). Vibe : a universal background subtraction algorithm for video sequences. *Transactions on Image Processing (TIP)*, 20(6) :1709–1724.

Barrow, H. G. and Tenenbaum, J. M. (1978). Recovering intrinsic scene characteristics from images. Technical report, Artificial Intelligence Center, SRI International.

Benenson, R., Mathias, M., Timofte, R., and Van Gool, L. (2012). Pedestrian detection at 100 frames per second. In *Proceedings of the 2012 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2903–2910. IEEE.

Benenson, R., Mathias, M., Tuytelaars, T., and Van Gool, L. (2013). Seeking the strongest rigid detector. In *Proceedings of the 2013 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3666–3673. IEEE.

Benenson, R., Timofte, R., and Van Gool, L. (2011). Stixels estimation without depth map computation. In *Proceedings of the 2011 International Conference on Computer Vision Workshops (ICCVW)*, pages 2010–2017. IEEE.

Berclaz, J., Fleuret, F., and Fua, P. (2006). Robust people tracking with global trajectory optimization. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 744–750. IEEE.

Berclaz, J., Fleuret, F., Turetken, E., and Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(9) :1806–1819.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance : The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008.

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35 :99–109.

Black, M. J. and Jepson, A. D. (1996). Estimating optical flow in segmented images using variable-order parametric models with local deformations. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(10) :972–986.

Blanco-Muriel, M., Alarcón-Padilla, D. C., López-Moratalla, T., and Lara-Coira, M. (2001). Computing the solar vector. *Solar Energy*, 70(5) :431–441.

Blinn, J. F. (1988). Me and my (fake) shadow. *Computer Graphics and Applications*, 8(1) :82–86.

Bose, B., Wang, X., and Grimson, W. E. L. (2007). Multi-class object tracking algorithm that handles fragmentation and grouping. In *Proceedings of the 2007 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE.

Bouwmans, T. (2014). Traditional and recent approaches in background modeling for foreground detection : an overview. *Computer Science Review*, 11–12 :31–66.

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

Brown, D. C. (1971). Close-range camera calibration. *Photometric Engineering*, 37(8) :855–866.

Brutzer, S., Höferlin, B., and Heidemann, G. (2011). Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the 2011 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1937–1944. IEEE.

Carr, P., Sheikh, Y., and Matthews, I. (2012). Monocular object detection using 3d geometric primitives. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, volume 7572 of *Lecture Notes in Computer Science*, pages 864–878. Springer Berlin Heidelberg.

Chen, C.-T., Su, C.-Y., and Kao, W.-C. (2010). An enhanced segmentation on vision-based shadow removal for vehicle detection. In *Proceedings of the International Conference on Green Circuits and Systems (ICGCS)*, pages 679–682.

Chen, Y.-T., Chen, C.-S., Huang, C.-R., and Hung, Y.-P. (2007). Efficient hierarchical method for background subtraction. *Pattern Recognition*, 40(10) :2706–2715.

Crow, F. C. (1977). Shadow algorithms for computer graphics. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, SIGGRAPH '77, pages 242–248. ACM.

Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2003). Detecting moving objects, ghosts and shadows in video streams. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(10) :1337–1342.

Cucchiara, R., Grana, C., Piccardi, M., Prati, A., and Sirotti, S. (2001). Improving shadow suppression in moving object detection with HSV color information. In *Intelligent Transportation Systems*, pages 334–339.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE.

Dallmeyer, J., Lattner, A. D., and Timm, I. J. (2014). Gis-based traffic simulation using osm. In Cervone, G., Lin, J., and Waters, N., editors, *Data Mining for Geoinformatics*, pages 65–82. Springer New York.

Di Lascio, R., Foggia, P., Percannella, G., Saggese, A., and Vento, M. (2013). A real time algorithm for people tracking using contextual reasoning. *Computer Vision and Image Understanding (CVIU)*, 117(8) :892–908.

Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 68.1–68.11. BMVA Press.

Dollár, P., Tu, Z., Perona, P., and Belongie, S. (2009). Integral channel features. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 91.1–91.11. BMVA Press.

Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. In Vernon, D., editor, *Proceedings of the 6th European Conference on Computer Vision (ECCV)*, volume 1843 of *Lecture Notes in Computer Science*, pages 751–767.

Ferryman, J. and Shahrokni, A. (2009). Pets2009 : Dataset and challenge. In *Proceedings of the 2009 12th International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, pages 1–6.

Fidler, S., Dickinson, S., and Urtasun, R. (2012). 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In Pereira, F., Burges, C. J., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 611–619. Curran Associates, Inc.

Finlayson, G. D., Drew, M. S., and Lu, C. (2009). Entropy minimization for shadow removal. *International Journal of Computer Vision (IJCV)*, 85(1) :35–57.

Finlayson, G. D., Hordley, S. D., Lu, C., and Drew, M. S. (2002). Removing shadows from images. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, volume 2353 of *Lecture Notes in Computer Science*, pages 823–836. Springer Berlin Heidelberg.

Fleuret, F., Berclaz, J., Lengagne, R., and Fua, P. (2008). Multi-camera people tracking with a probabilistic occupancy map. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(2) :267–282.

Fortmann, T. E., Bar-Shalom, Y., and Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3) :173–184.

Gelencsér-Horváth, A., Tornai, G., Horváth, A., and Cserey, G. (2013). Fast, parallel implementation of particle filtering on the gpu architecture. *EURASIP Journal on Advances in Signal Processing*, 2013(1).

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2) :107–113.

Goyat, Y., Chateau, T., Malaterre, L., and Trassoudaine, L. (2006). Vehicle trajectories evaluation by static video sensors. In *Intelligent Transportation Systems Conference (ITSC)*, pages 864–869.

Goyat, Y., Chateau, T., and Trassoudaine, L. (2010). Tracking of vehicle trajectory by combining a camera and a laser rangefinder. *Machine Vision and Applications*, 21(3) :275–286.

Green, M. W., National Institute of Justice (U.S.), Safe and Drug-Free Schools Program (U.S.), and Sandia National Laboratories (1999). The appropriate and effective use of security technologies in u.s. schools : a guide for schools and law enforcement agencies. Research report, U.S. Dept. of Justice, Office of Justice Programs, National Institute of Justice.

Greenhill, D., Renno, J., Orwell, J., and Jones, G. A. (2008). Occlusion analysis : Learning and utilising depth maps in object tracking. *Image and Vision Computing*, 26(3) :430–441. 15th Annual British Machine Vision Conference.

Grena, R. (2008). An algorithm for the computation of the solar position. *Solar Energy*, 82(5) :462–470.

Grena, R. (2012). Five new algorithms for the computation of sun position from 2010 to 2110. *Solar Energy*, 86(5) :1323–1337.

Gu, L. and Robles-Kelly, A. (2012). Shadow detection via rayleigh scattering and mie theory. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 2165–2168.

Hartley, R. and Zisserman, A. (2004). *Multiple view geometry in computer vision*. Cambridge University Press, 2 edition.

Henriques, J. F., Caseiro, R., and Batista, J. (2011). Globally optimal solution to multi-object tracking with merged measurements. In *Proceedings of the 2011 International Conference on Computer Vision (ICCV)*, pages 2470–2477. IEEE.

Henschel, R., Leal-Taixé, L., and Rosenhahn, B. (2014). Efficient multiple people tracking using minimum cost arborescences. In Jiang, X., Hornegger, J., and Koch, R., editors, *Pattern Recognition*, volume 8753 of *Lecture Notes in Computer Science*, pages 265–276. Springer International Publishing.

Hofmann, M., Tiefenbacher, P., and Rigoll, G. (2012). Background segmentation with feedback : the pixel-based adaptive segmenter. In *Proceedings of the 2012 Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 38–43. IEEE.

Horn, B. K. P. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1–3) :185–203.

Hsieh, J.-W., Hu, W.-F., Chang, C.-J., and Chen, Y.-S. (2003). Shadow elimination for effective moving object detection by gaussian shadow modeling. *Image and Vision Computing*, 21(6) :505–516.

Huang, J.-B. and Chen, C.-S. (2009). Moving cast shadow detection using physics-based features. In *Proceedings of the 2009 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2310–2317. IEEE.

Isard, M. and Blake, A. (1998). Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1) :5–28.

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. In Kadar, I., editor, *Proceedings of the Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 182–193.

Kaewtrakulpong, P. and Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. In Remagnino, P., Jones, G. A., Paragios, N., and Regazzoni, C. S., editors, *Video-Based Surveillance Systems*, pages 135–144. Springer US.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1) :35–45.

Kaucic, R., Perera, A. G. A., Brooksby, G., Kaufhold, J., and Hoogs, A. (2005). A unified framework for tracking through occlusions and across sensor gaps. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 990–997. IEEE.

Khan, Z., Balch, T., and Dellaert, F. (2004). An mcmc-based particle filter for tracking multiple interacting targets. In Pajdla, T. and Matas, J., editors, *Proceedings of the 8th European Conference on Computer Vision (ECCV)*, volume 3024 of *Lecture Notes in Computer Science*, pages 279–290. Springer Berlin Heidelberg.

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.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2) :83–97.

Lalonde, J.-F., Efros, A. A., and Narasimhan, S. G. (2010). Detecting ground shadows in outdoor consumer photographs. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, volume 6312 of *Lecture Notes in Computer Science*, pages 322–335. Springer Berlin Heidelberg.

Leone, A. and Distante, C. (2007). Shadow detection for moving objects based on texture analysis. *Pattern Recognition*, 40(4) :1222–1233.

Leotta, M. J. and Mundy, J. L. (2011). Vehicle surveillance with a generic, adaptive, 3d vehicle model. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(7) :1457–1469.

Li, Y., Huang, C., and Nevatia, R. (2009). Learning to associate : Hybridboosted multi-target tracker for crowded scene. In *Proceedings of the 2009 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2953–2960. IEEE.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the DARPA Imaging Understanding Workshop*, pages 121–130.

Maddalena, L. and Petrosino, A. (2008). A self-organizing approach to background subtraction for visual surveillance applications. *Transactions on Image Processing (TIP)*, 17(7) :1168–1177.

Mathias, M., Benenson, R., Pedersoli, M., and Van Gool, L. (2014). Face detection without bells and whistles. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, volume 8692 of *Lecture Notes in Computer Science*, pages 720–735. Springer International Publishing.

McFarlane, N. and Schofield, C. (1995). Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3) :187–193.

Meeus, J. H. (1991). *Astronomical algorithms*. Willmann-Bell, Incorporated, 1 edition.

Michalsky, J. J. (1988). The astronomical almanac's algorithm for approximate solar position (1950 - 2050). *Solar Energy*, 40(3) :227–235.

Montemayor, A. S., Pantrigo, J. J., Sánchez, . A., and Fernández, F. (2004). Particle filter on gpus for real-time tracking. In Barzel, R., editor, *SIGGRAPH 2004 Posters*, SIGGRAPH '04, page 94. ACM.

Munoz-Salinas, R., Aguirre, E., and García-Silvente, M. (2007). People detection and tracking using stereo vision and color. *Image and Vision Computing*, 25(6) :995–1007.

Nadimi, S. and Bhanu, B. (2004). Physical models for moving shadow and object detection in video. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(8) :1079–1087.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4) :308–313.

Nicolas, H. and Pinel, J.-M. (2006). Joint moving cast shadows segmentation and light source detection in video sequences. *Signal Processing : Image Communication*, 21(1) :22–43.

Okuma, K., Taleghani, A., De Freitas, N., Little, J. J., and Lowe, D. G. (2004). A boosted particle filter : Multitarget detection and tracking. In Pajdla, T. and Matas, J., editors, *Proceedings of the 8th European Conference on Computer Vision (ECCV)*, volume 3021 of *Lecture Notes in Computer Science*, pages 28–39. Springer Berlin Heidelberg.

Oliver, N., Rosario, B., and Pentland, A. (2000). A bayesian computer vision system for modeling human interactions. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8) :831–843.

Panicker, J. and Wilscy, M. (2010). Detection of moving cast shadows using edge information. In *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE)*, volume 5, pages 817–821.

Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision (IJCV)*, 38(1) :15–33.

Prati, A., Mikic, I., Trivedi, M. M., and Cucchiara, R. (2003). Detecting moving shadows : algorithms and evaluation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(7) :918–923.

Prisacariu, V. and Reid, I. (2009). fasthog - a real-time gpu implementation of hog. Technical Report 2310/09, Department of Engineering Science, Oxford University.

Qin, R., Liao, S., Lei, Z., and Li, S. (2010). Moving cast shadow removal based on local descriptors. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 1377–1380.

Reda, I. and Andreas, A. (2008). Solar position algorithm for solar radiation applications. Technical report, National Renewable Energy Laboratory.

Reid, D. B. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6) :843–854.

Rogez, M., Robinault, L., and Tougne, L. (2014). A 3d tracker for ground-moving objects. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., McMahan, R., Jerald, J., Zhang, H., Drucker, S., Kambhamettu, C., El Choubassi, M., Deng, Z., and Carlson, M., editors, *Advances in Visual Computing*, volume 8888 of *Lecture Notes in Computer Science*, pages 695–705. Springer International Publishing.

Rogez, M., Tougne, L., and Robinault, L. (2013). A prior-knowledge based casted shadows prediction model featuring openstreetmap data. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, volume 1, pages 602–607.

Sabzmeydani, P. and Mori, G. (2007). Detecting pedestrians by learning shapelet features. In *Proceedings of the 2007 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE.

Salvador, E., Cavallaro, A., and Ebrahimi, T. (2004). Cast shadow segmentation using invariant color features. *Computer Vision and Image Understanding (CVIU)*, 95(2) :238–259.

Sanin, A., Sanderson, C., and Lovell, B. C. (2010). Improved shadow removal for robust person tracking in surveillance scenarios. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 141–144.

Sanin, A., Sanderson, C., and Lovell, B. C. (2012). Shadow detection : a survey and comparative evaluation of recent methods. *Pattern Recognition*, 45(4) :1684–1695.

Smeulders, A. W. M., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., and Shah, M. (2014). Visual tracking : an experimental survey. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(7) :1442–1468.

Stauffer, C. and Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings of the 1999 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2246–2252. IEEE.

Sun, B. and Li, S. (2010). Moving cast shadow detection of vehicle using combined color models. In *Proceedings of the Chinese Conference on Pattern Recognition (CCPR)*, pages 1–5.

Tian, Y. L., Lu, M., and Hampapur, A. (2005). Robust and efficient foreground analysis for real-time video surveillance. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1182–1187. IEEE.

Vacavant, A., Tougne, L., Chateau, T., and Robinault, L. (2013). Background models challenge, workshop of accv 2012. In Park, J.-I. and Kim, J., editors, *Proceedings of the 11th Asian Conference on Computer Vision Workshops (ACCVW)*, volume 7728 of *Lecture Notes in Computer Science*, pages 291–300. Springer Berlin Heidelberg.

Vermaak, J., Doucet, A., and Pérez, P. (2003). Maintaining multimodality through mixture tracking. In *Proceedings of the 9th International Conference on Computer Vision (ICCV)*, volume 2, pages 1110–1116. IEEE.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 57(2) :137–154.

Weiss, Y. (2001). Deriving intrinsic images from image sequences. In *Proceedings of the 8th International Conference on Computer Vision (ICCV)*, volume 2, pages 68–75. IEEE.

Williams, L. (1978). Casting curved shadows on curved surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, SIGGRAPH '78, pages 270–274. ACM.

Wu, B. and Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Proceedings of the 10th International Conference on Computer Vision (ICCV)*, volume 1, pages 90–97. IEEE.

Wu, B. and Nevatia, R. (2006). Tracking of multiple, partially occluded humans based on static body part detection. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 951–958. IEEE.

Xiao, J., Russell, B., and Torralba, A. (2012). Localizing 3d cuboids in single-view images. In Pereira, F., Burges, C. J., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 746–754. Curran Associates, Inc.

Xu, D., Li, X., Liu, Z., and Yuan, Y. (2005). Cast shadow detection in video segmentation. *Pattern Recognition Letters*, 26(1) :91–99.

Yang, B. and Nevatia, R. (2012). An online learned crf model for multi-target tracking. In *Proceedings of the 2012 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2034–2041. IEEE.

Yoneyama, A., Yeh, C. H., and Kuo, C. C. J. (2003). Moving cast shadow elimination for robust vehicle extraction based on 2d joint vehicle/shadow models. In *Proceedings of the Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 229–236. IEEE.

Yoshinaga, S., Shimada, A., Nagahara, H., and Taniguchi, R.-i. (2013). Background model based on statistical local difference pattern. In Park, J.-I. and Kim, J., editors, *Proceedings of the 11th Asian Conference on Computer Vision Workshops (ACCVW)*, volume 7728 of *Lecture Notes in Computer Science*, pages 327–332. Springer Berlin Heidelberg.

Zhang, W., Fang, X. Z., and Xu, Y. (2006). Detection of moving cast shadows using image orthogonal transform. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, volume 1, pages 626–629.

Zhao, T. and Nevatia, R. (2004). Tracking multiple humans in crowded environment. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 406–413. IEEE.

Zhu, Q., Yeh, M.-C., Cheng, K.-T., and Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1491–1498. IEEE.

Zilske, M., Neumann, A., and Nagel, K. (2011). OpenStreetMap for traffic simulation. In Schmidt, M. and Gartner, G., editors, *Proceedings of the 1st European State of the Map – OpenStreetMap conference*, pages 126–134.

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.

Zuniga, M., Bremond, F., and Thonnat, M. (2006). Fast and reliable object classification in video based on a 3d generic model. In *Proceedings of the 3rd International Conference on Visual Information Engineering (VIE)*, pages 433–440. IET.

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.