Vers un système de capture du mouvement humain en 3D pour un robot mobile évoluant dans un environnement encombré

Abdallah Dib

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Vers un système de capture du mouvement humain en 3D pour un robot mobile évoluant dans un environnement encombré

THÈSE

présentée et soutenue publiquement le 24 Mai 2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Abdallah DIB

Composition du jury

Rapporteurs :	Roland Chapuis	Professeur à l'Université Blaise Pascal, Clermont-Ferrand
	Christian Wolf	Maitre de conférence, HDR à l'Université de Lyon
Examinateurs :	Jean-Philippe Blanchard	Responsable du Pôle Innovation Credit Agricole, France
	David Daney	Chargé de recherche Inria, Bordeaux
	Isabelle Debled-Rennesson	Professeur à l'Université de Lorraine, Nancy
	Vincent Dupourque	PDG société Robosoft, France
	Mohamad Khalil	Professeur à l'Université Libanaise, Liban
Encadrant :	François Charpillet	Directeur de recherche INRIA, Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria Laboratoire lorrain de recherche en informatique et ses applications

Inria INVENTEURS DU MONDE NUMÉRIQUE

Mis en page avec la classe thesul.

Remerciements

Je tiens à remercier mon directeur de thèse, François Charpillet, avec qui j'ai eu le plaisir de travailler avec lui durant les 3 années de la thèse et aussi les 5 années qui ont précédé cette thèse. Je le remercie principalement pour ses conseils aux niveaux scientifiques et personnels, et aussi pour les discussions régulières que nous avons eu qui m'ont permis de réaliser cette thèse.

Je remercie également les membres de mon jury : Isabelle Debled-Rennesson, professeur à l'université de Lorraine, qui m'a fait l'honneur de présider ce jury, Roland Chapuis, Professeur à l'université Blaise Pascal, et Christian Wolf, maître de conférence, HDR à l'université de Lyon, pour avoir accepté de rapporter cette thèse, et surtout pour leurs remarques et critiques qui ont contribué à l'amélioration de ce mémoire. Je remercie également : David Daney, Chargé de recherche Inria, Vincent Dupourque, le PDG de la société Robosoft, et Mohamad Khalil, Professeur à l'université Libanaise, pour avoir accepté de faire partie du jury et pour leurs remarques et questions constructives.

J'exprime mes sincères remerciements à Jean-Philippe Blanchard, Responsable du Pôle innovation à Credit Agricole, pour son enthousiasme, sa collaboration, ses conseils et ses corrections qui ont contribué à l'amélioration de ce mémoire, je le remercie vivement pour sa participation au jury.

Je remercie également les membres de l'ancienne équipe, Maia, et la nouvelle, Larsen, pour les conseils et corrections qui ont contribué à la bonne rédaction du manuscrit : Olivier Buffet, Vincent Thomas, Francis Colas, Inaki Fernandez, Nassim Kalde.

J'exprime ma profonde reconnaissance à mes amis Cédric Rose et Amandine Dubois, pour leur contribution scientifique dans ce travail de thèse et leur participation directe dans la correction et l'amélioration de ce document. Je les remercie principalement pour leur amitié, leur écoute et leur soutien tout au long de cette thèse. Sans eux, ces années n'auraient pas été aussi agréables.

Je remercie tous les membres de l'ancienne et la nouvelle équipe qui m'ont permis de passer huit ans au LORIA dans une ambiance chaleureuse, de bonne humeur et de plaisir partagé. Je remercie particulièrement, Jamal Saboune, Gabriel Corona, Nicolas Pepinhermann, Nicolas Beaufort, Mihai Andries, Thomas Moinel, Melanie Lelaure, Adrian Bourgaud, Alain Filbois, Serena Ivaldi et Jean-Baptiste Mouret. J'en oublie sûrement, qu'ils veulent bien m'en excuser.

Je remercie mes amis à Nancy, Ahmad Ahmad, Racha El Zein, Dina et Jamil Chawki, Bilal et Khaled Missayke, Ezzedine et Bilal Cherif, abdelrahman hamed et Jamil Sarhal pour leur amitié et leur soutien.

Je veux exprimer toute ma gratitude à ma mère et mes frères, Rabih, Mohamad et Sami et ma sœur, Soline qui m'ont aidé et soutenu depuis toujours, et sans eux, je ne serais pas parvenu à accomplir cette thèse.

Asma, je te remercie pour tout, surtout pour ton soutien permanent et ta présence à coté de moi pendant les moments très difficiles de cette thèse. Sans toi, je ne serais pas parvenu à accomplir ce travail. Je voudrais remercier finalement mes enfants, Leen et Mohamad-Adam qui sont ma source d'inspiration et mon plus grand soutien.

Je dédie cette thèse
à mon père décédé,
j'espère que tu es fier de moi.

Sommaire

Liste des tableaux **xiii**

Table des figures **xv**

Partie I Ouverture **1**

1 Contexte 3

2 Le projet LAR 4

2.1 Robot Kompai 4

2.2 Camera RGB-D 5

3 Problématique et contributions 6

4 Organisation du mémoire 7

Partie II Extraction de la silhouette et suivi 3D du mouvement humain **9**

Introduction **11**

Chapitre 1
Méthodes existantes pour la capture du mouvement humain

1.1 Capture de mouvement par les méthodes à base de marqueurs ou capteurs fixés sur le corps 14

1.2 Capture de mouvement à partir d'une ou plusieurs caméras fixes 14

1.2.1 Panorama des méthodes existantes 15

1.2.2 Discussion 17

1.2.3 Méthodes basées sur l'apprentissage (Méthode de Microsoft) 17

1.2.4 Méthodes stochastiques utilisant le filtrage particulaire 19

1.3 Capture de mouvement à partir d'une caméra mobile 20

1.4 Conclusion 20

Chapitre 2
Filtrage particulaire : Théorie et application pour le suivi du mouvement humain

2.1 Problème de filtrage 23
2.2 Rappel sur la méthode de Monte Carlo 24
2.2.1 Echantillonnage préférentiel 25
2.3 Echantillonnage préférentiel séquentiel (SIS) 25
2.4 Filtre particulaire 28
2.5 Application du filtrage particulaire pour le suivi du mouvement humain . . . 29
2.5.1 Filtrage particulaire avec recuit-simulé 30
2.5.2 Filtrage particulaire factorisé 32
2.5.3 Approche combinée 34
2.6 Conclusion 35

Chapitre 3
Extraction de la silhouette dans un environnement dynamique

3.1 Méthodes existantes 38
3.2 Rappel sur le Modèle de Markov Caché 42
3.2.1 Modèle de Markov 42
3.2.2 Modèle de Markov caché MMC ou HMM 42
3.3 Modèle HMM pour identifier les objets fixes et mobiles dans la scène 44
3.3.1 Evidence virtuelle 47
3.3.2 Inférence 47
3.3.3 Modélisation du fond dynamique 49
3.4 Extraction de la silhouette 50
3.4.1 Étiquetage 50
3.4.2 Reconstruction de la silhouette à partir d'une étiquette 52
3.5 Conclusion 53

Chapitre 4
Suivi 3D du mouvement humain dans un environnement encombré

4.1 Modèle 3D virtuel 56
4.2 Fonction de vraisemblance 57
4.2.1 Choix de la fonction de vraisemblance 57
4.2.2 Construction de la fonction de vraisemblance 58
4.3 Choix de l'estimateur bayésien 61

4.4 Limites des méthodes existantes pour le suivi dans un environnement avec occlusions . . . 62
4.5 Algorithme de suivi du mouvement humain dans un environnement avec occlusions . . . 62
4.5.1 Nouveau diagramme de condensation . . . 67
4.5.2 Prise en compte de l'état de visibilité du parent d'une partie du corps 69
4.5.3 Identification de l'état de visibilité des parties du corps . . . 71
4.5.4 Recuit-simulé . . . 74
4.5.5 Ré-échantillonnage . . . 75
4.6 Conclusion . . . 75

Chapitre 5
Evaluation

5.1 Construction d'une base de données de capture de mouvement avec vérité terrain . . . 80
5.1.1 Principe de fonctionnement du système Qualisys . . . 80
5.1.2 Calibration de la Kinect par rapport au système Qualisys . . . 80
5.2 Comparaison avec les algorithmes APF et PS . . . 82
5.2.1 Choix des paramètres . . . 83
5.2.2 Initialisation . . . 83
5.2.3 Critères d'évaluation . . . 84
5.2.4 Erreur moyenne et écart type . . . 85
5.2.5 Pourcentage d'erreur . . . 86
5.2.6 Comparaison Qualitative . . . 88
5.2.7 Identification des parties cachées du corps . . . 89
5.3 Evaluation de la méthode d'extraction de la silhouette . . . 90
5.3.1 Simulation . . . 90
5.3.2 Etat initial de la grille d'occupation . . . 93
5.3.3 Influence de la résolution de la grille d'occupation sur la silhouette . . 93
5.3.4 Adaptation aux changements da la scène . . . 93
5.4 Vitesse d'exécution et implémentation sur carte graphique . . . 94
5.5 Conclusion . . . 95

Conclusion générale **105**

Partie III Localisation d'une caméra mobile dans un environnement dynamique 107

Introduction 109

Chapitre 6
Etat de l'art

6.1 Méthodes par extraction des points d'intérêt 111
6.2 Méthodes directes 115
6.2.1 Estimation du mouvement de la caméra par minimisation de l'erreur géométrique 115
6.2.2 Estimation du mouvement de la caméra par minimisation de l'erreur photométrique 116
6.3 Conclusion 118

Chapitre 7
Contexte méthodologique

7.1 Représentation minimaliste par l'algèbre de Lie du mouvement d'un objet rigide dans l'espace. 121
7.2 Modèle de la caméra 122
7.3 Méthode de moindres carrés 123
7.4 Ransac 125
7.5 Conclusion 127

Chapitre 8
Estimation robuste du mouvement de la caméra dans un environnement dynamique.

8.1 Formulation mathématique 130
8.1.1 Calcul de la Jacobienne 132
8.2 Pyramide multi-résolution 133
8.3 Estimation robuste du mouvement de la caméra 134
8.3.1 Principe de fonctionnement 135
8.3.2 Application de l'algorithme RANSAC 136
8.3.3 Nettoyage de la matrice jacobienne 137
8.3.4 Choix de nombre des pixels 139
8.3.5 Algorithme d'estimation robuste du mouvement de la caméra 141
8.4 Conclusion 143

Chapitre 9
Evaluation

9.1 Evaluation de la capacité de notre méthode à éliminer les pixels aberrants . . 145
9.1.1 Caméra fixe dans un environnement dynamique 145
9.1.2 Caméra montée sur un motor pan-tilt 148
9.1.3 Résultats qualitatifs 148
9.2 Évaluation avec les données du Benchmark 149
9.3 Influence des paramètres sur les performances 151
9.3.1 Seuil pour le nettoyage de la jacobienne 151
9.3.2 Choix du seuil 151
9.3.3 Nombre de pixels 152
9.4 Vitesse d'exécution 152
9.5 Conclusion 152

Conclusion générale **161**

Partie IV Vers une plate-forme robotique d'analyse de mouvement de la personne dans un environnement intérieur et encombré **163**

Chapitre 10 Système de perception visuelle de la posture humaine **165**
10.1 Architecture 165
10.2 Résultats préliminaires 166
10.2.1 Limites constatées 168
10.2.2 Pistes de travail 170
10.3 Discussion 171

Conclusion et perspective **173**

Annexes **177**

Annexe A Ré-échantillonnage multinomial **179**

Annexe B Calcul des composants de la jacobienne **181**

Publications **183**

Références **185**

Liste des tableaux

5.1 Performances de notre méthode (OUR), APF et PS sur le *benchmark* que nous avons construit. 91
5.2 Tableau de contingence 92
5.3 Temps d'exécution de la méthode d'extraction de la silhouette pour une grille d'occupation de taille $5 \times 5 \times 2.5$ mètres avec différentes résolutions. 94

9.1 Erreur moyenne de la dérive en mètre par seconde (RMSE) pour les différentes méthodes dans différentes scènes dynamiques du *benchmark*. 150

Table des figures

1 Indicateurs démographiques pour la France entre les années 1950 et 2013. 3
2 Evolution de la population Française pour les années 1914, 1974 et 2014 4
3 Robot Kompai développé par la société Robosoft. 5
4 L'interface utilisateur du robot Kompai qui permet de contrôler le robot. 5

1.1 Principe de la capture de mouvement à partir d'une caméra 16
1.2 Un extrait de la base d'apprentissage utilisée par Microsoft pour leur classificateur 18
1.3 Le processus de la méthode de capture de mouvement développée par Microsoft . 18
1.4 L'estimation de pose par la méthode de Microsoft dans des situations d'occlusions 19

2.1 Attraction des particules par un maximum local dans un filtre particulaire 30
2.2 Principe de fonctionnement du recuit-simulé 32
2.3 Diagramme de condensation du filtre particulaire factorisé (PS). 34
2.4 Exemples de deux stratégies d'exploration dans un filtrage particulaire factorisé . 36

3.1 Les différentes étapes de traitement de la méthode développée 38
3.2 Sortie typique d'un système de détection des personnes 39
3.3 La méthode d'extraction de la silhouette utilisée par Zhang *et al.* 40
3.4 La méthode d'extraction de la silhouette utilisée par Jafari *et al.* 42
3.5 Un modèle de Markov à deux états. 43
3.6 Représentation graphique d'un HMM. 43
3.7 HMM à trois états utilisé pour estimer l'état de chaque cellule de la grille. 45
3.8 Une cellule peut être invisible, occupée ou visible 46
3.9 Nouvelle représentation graphique avec l'introduction de l'évidence virtuelle $e_{Y_t^i}$. 48
3.10 La fonction $f(\varepsilon_i)$ représentant la vraisemblance de l'état occupé $P(e_{Y_t^i} \mid Y_t^i = hit)$. 48
3.11 Détection des objets mobiles (en bleu) et fixes (en vert) avec le HMM. 49
3.12 HMM modifié avec une nouvelle probabilité de transition de l'état **M** à **F** 50
3.13 Capacité de notre méthode à apprendre en ligne le fond 51
3.14 Exemple de fonctionnement de l'algorithme d'étiquetage en 2D 53

4.1 Le modèle 3D utilisé pour le suivi 57
4.2 Chaîne cinématique représentant le modèle 3D virtuel 58
4.3 Principe du rendu 3D 59
4.4 Calcul de la fonction de distance 60
4.5 Valeurs de la fonction de distance suite à une variation d'un seul degré de liberté 61
4.6 Limitations des méthodes de filtrage particulaire dans des situations d'occlusions 63
4.7 Diagramme de condensation de l'algorithme PS 65
4.8 Exemple de fonctionnement de l'algorithme PS 68

4.9 Changement de la stratégie d'exploration 69
4.10 Nouveau diagramme de condensation 70
4.11 Nouveau diagramme de condensation avec mémoire. 71
4.12 Un exemple d'une personne en-train de sortir d'une situation d'occlusion 72
4.13 L'approximation des cylindres par un ensemble de sphères. 73

5.1 L'emplacement des 8 caméras du système Qualisys et de la Kinect 81
5.2 Quelques captures d'écran du *Benchmark* que nous avons construit. 82
5.3 Capture d'écran du programme fourni avec la base de données 83
5.4 Initialisation des filtres particulaires 84
5.5 Comparaison de l'erreur moyenne et l'écart type des trois méthodes 85
5.6 Pourcentage de temps où l'erreur ne dépasse pas 20 cm 86
5.7 Pourcentage de temps où l'erreur ne dépasse pas 15 cm 87
5.8 Pourcentage de temps où l'erreur ne dépasse pas 10 cm 88
5.9 Courbe d'erreur obtenue par notre méthode, APF et PS 89
5.10 Décalage entre la position des articulations et les marqueurs 90
5.11 Courbe d'erreur obtenue par notre méthode et par APF 92
5.12 Comparaison de notre méthode avec APF 97
5.13 Résultats de suivi obtenus par notre méthode dans des situations d'occlusions . . 98
5.14 Résultats de suivi obtenus par notre méthode dans des situations d'occlusions . . 99
5.15 Résultats de classification par simulation 100
5.16 Impact de la résolution de la grille d'occupation sur la qualité de la silhouette . . 101
5.17 Résultats visuels de la sortie de la grille d'occupation et la silhouette extraite . . 102
5.18 Images montrant la capacité de notre méthode à apprendre le fond de la scène . . 103

6.1 Principe de l'odométrie visuelle 112
6.2 Les points de Harris extraits de l'image. 113
6.3 Mise en correspondance des pixels entre deux images 113
6.4 Géométrie épipolaire 114
6.5 Principe de fonctionnement de l'algorithme ICP 115
6.6 Principe de l'odométrie visuelle dense par minimisation de l'erreur photométrique 117

7.1 Modèle simplifié de la caméra 123
7.2 Sensibilité de la méthode des moindres carrés aux points aberrants 126
7.3 Un exemple de fonctionnement de RANSAC sur une régression linéaire 128

8.1 Hypothèse Lambertienne 130
8.2 La fonction de *warp* permettant de projeter chaque pixel de l'image I_t dans I_{t+1} 132
8.3 Processus itérative d'alignement des images avec une pyramide multi-résolution . 134
8.4 Courbe d'erreur sur la position de la caméra. 135
8.5 Suite à une sélection d'un ensemble de pixels, trois cas de figures se présentent . . 136
8.6 Les endroits uniformes dans la scène possède un faible gradient proche de zéros . 138
8.7 Un exemple illustrant l'importance de l'information géométrique représentée par J_G 139
8.8 Un exemple montrant les pixels éloignés ne décrivent pas la translation 140
8.9 Variation de nombre des itérations en fonction de n dans RANSAC 140
8.10 Processus de minimisation robuste par RANSAC. 142

9.1 Détection des pixels pertinents et aberrants par notre méthode 146
9.2 Courbe d'erreur sur la position et l'orientation pour une caméra fixe 147

9.3 Courbe d'erreur sur la position et l'orientation d'une caméra sur un pan-tilt . . . 148
9.4 Images montrant la capacité de notre méthode à éliminer les pixels aberrants . . 154
9.5 Carte 3D obtenue obtenue par notre méthode dans une scène statique. 155
9.6 Carte 3D obtenue par notre méthode dans une scène dynamique 155
9.7 Images prises du TUM *benchmark* avec un nombre élevé de pixels invalides . . . 156
9.8 Image résultante pour différente valeur du seuil s 157
9.9 Erreur moyenne sur la position de la caméra pour différentes valeurs de t 158
9.10 Variation de la vitesse d'exécution en fonction de nombre des pixels choisis 158
9.11 Erreur moyenne sur la position de la caméra pour différentes valeurs de n 159

10.1 Architecture du système de perception visuelle de la posture humaine 166
10.2 La personne filmée par une caméra montée sur un robot mobile. 168
10.3 Résultats de suivi obtenus par notre système avec une caméra mobile 169
10.4 Impact de la dérive de l'odométrie sur le fonctionnement du système 170

Première partie

Ouverture

1 Contexte

Le vieillissement de la population est un enjeu majeur auquel les sociétés modernes vont devoir faire face dans les années à venir. La population française a augmenté de 21 millions de personnes entre 1950 et 2013. Pendant ces 64 ans, le pourcentage des jeunes de moins de 20 ans a baissé. En 1950 les jeunes représentaient 29.9%, alors qu'en 2013, ils ne représentent plus que 24.4% de la population française. Cette baisse de la proportion des jeunes a été accompagnée d'une augmentation du nombre de seniors (65 ans ou plus) qui est passé de 11.3% à 18.2%. Cette augmentation est liée à l'espérance de vie qui est passée de 60 ans en 1950 à 80 ans en 2013. Le taux de naissance a chuté de 20.5 à 12.2 pour 1000 habitants. Ces chiffres sont extraits des statistiques de l'INSEE illustrées dans le tableau de la figure 1.

Tableau. Indicateurs démographiques 1950 à 2013, France métropolitaine

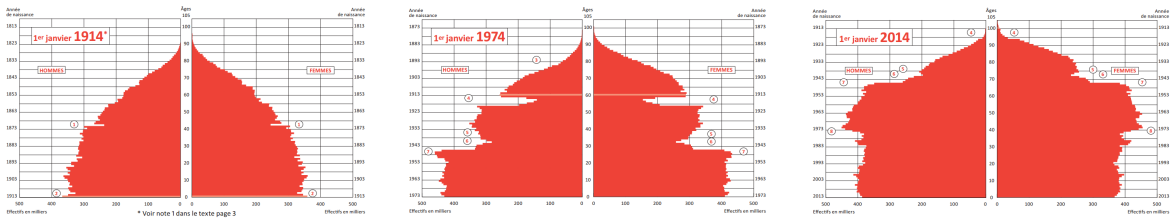
	1950	1960	1970	1980	1990	2000	2005	2006	2007	2008	2009	2010	2011	2012(p)	2013(p)
Naissances (m)	858	816	848	800	762	775	774	797	786	796	793	802	793	790	780
Décès (m)	530	517	540	547	526	531	528	516	521	532	538	540	535	559	561
Excédent naturel (m)	328	299	308	253	236	244	247	280	265	264	255	262	258	232	219
Solde migratoire (m)	35	140	180	44	80	70	95	115	75	67	44	43	50	50	50
Variation totale (m)	363	439	488	297	316	314	342	395	340	331	299	305	308	282	269
Ajustement (1) (m)	-	-	-	-	-	94	94		-	-	-	-	-	-	-
Taux de natalité (t)	20,5	17,9	16,7	14,9	13,4	13,1	12,7	12,9	12,7	12,8	12,7	12,7	12,5	12,4	12,2
Taux de mortalité (t)	12,7	11,3	10,6	10,2	9,3	9,0	8,6	8,4	8,4	8,5	8,6	8,6	8,5	8,8	8,8
Taux de mort. infantile (r)	51,9	27,4	18,2	10,0	7,3	4,4	3,6	3,6	3,6	3,6	3,7	3,5	3,3	3,3	3,5
Indice de fécondité (e)	2,93	2,73	2,47	1,94	1,78	1,87	1,92	1,98	1,96	1,99	1,99	2,02	2,00	1,99	1,97
Espérance de vie :															
hommes (a)	63,4	67,0	68,4	70,2	72,7	75,3	76,8	77,2	77,4	77,6	77,8	78,0	78,4	78,5	78,7
femmes (a)	69,2	73,6	75,9	78,4	80,9	82,8	83,8	84,2	84,4	84,4	84,5	84,7	85,0	84,9	85,0
Mariages (m)	331	320	394	334	287	298	276	267	267	259	245	245	231	240	225
Taux de nuptialité (t)	7,9	7,0	7,8	6,2	5,1	5,0	4,5	4,3	4,3	4,2	3,9	3,9	3,7	3,8	3,5
Population (2) (m)	42 010	45 904	51 016	54 029	56 893	59 267	61 400	61 795	62 135	62 466	62 765	63 070	63 379	63 660	63 929
Moins de 20 ans (2) (m)	12 556	14 665	16 748	16 419	15 632	15 068	15 280	15 315	15 338	15 369	15 407	15 440	15 485	15 534	15 606
65 ans ou plus (2) (m)	4 727	5 288	6 174	7 541	8 036	9 561	10 163	10 208	10 301	10 421	10 540	10 667	10 978	11 295	11 619
Moins de 20 ans (2) %	29,9	31,9	32,8	30,4	27,5	25,4	24,9	24,8	24,7	24,6	24,5	24,4	24,4	24,4	24,4
65 ans ou plus (2) %	11,3	11,5	12,1	14,0	14,1	16,1	16,6	16,5	16,6	16,7	16,8	16,9	17,3	17,7	18,2

(a) années – (e) nombre d'enfants par femme – (m) milliers – (p) provisoire – (r) pour 1 000 naissances vivantes – (t) pour 1 000 habitants.
(1) Les estimations de population pour la période 1990-2005 tiennent compte d'un ajustement destiné à rétablir la cohérence comptable entre les recensements de 1990, 1999 et 2006 (voir Vanessa Bellamy et Catherine Beaumel, 2014 [4]).
(2) En fin d'année.
Source : Insee, Division des enquêtes et études démographiques (http://www.insee.fr).

FIGURE 1 – Indicateurs démographiques pour la France entre les années 1950 et 2013.

Pour ces raisons et pour d'autres raisons sociales et économiques, le pourcentage des jeunes diminue en faveur de celui des personnes âgées. Ainsi la pyramide des âges dans des pays comme l'Espagne, l'Italie et le Japon, s'est transformée en une "toupie". En France, la structure de la population évolue aussi vers une "toupie". La figure 2 illustre nettement cette évolution.

Face au vieillissement de la population, la mise en œuvre de solutions pour la prise en charge des personnes âgées s'avère importante pour la société. L'un des enjeux, par sa dimension sociale et économique porte sur le maintien à domicile ([Thomesse *et al.*, 2001], [Hewson *et al.*, 2007]), alternative à la maison de retraite et/ou à l'hôpital. Pour les personnes ayant gardé leur autonomie, la problématique est de conserver cet état le plus longtemps possible, tout en limitant les risques, notamment les accidents domestiques ou les malaises, qui, non pris en compte rapidement, peuvent dégénérer en situation extrêmement dangereuse. Mais il faut aussi renforcer le lien social afin d'éviter l'isolement pouvant conduire à des dépressions graves. Pour les personnes en perte d'autonomie, les solutions existantes peuvent mobiliser parfois à plein temps des personnes



(a) (b) (c)

FIGURE 2 – Evolution de la Structure de la population Française pour les années 1914, 1974 et 2014. a) Du pyramide (1914) c) Vers la toupie (2014).

du cercle familial (aidants) ou du monde de la santé. Le coût psychologique pour les aidants est d'autant plus fort qu'ils sont souvent sur le front 7 jours sur 7 et 24 heures sur 24. Quant aux seconds, leur coût économique pour les personnes et la collectivité est très élevé, même s'il reste parfois inférieur à l'accueil en institution spécialisée.

2 Le projet LAR

Cette thèse s'est effectuée dans le cadre du projet LAR (Living Assistant Robot) financé en réponse à l'appel « Technologies de l'e-santé- Santé et autonomie sur le lieu de vie grâce au numérique » (programme des Investissements d'Avenir).

Le projet LAR s'est construit autour d'un consortium constitué de quatre partenaires : Le Credit Agricole, la société Robosoft, la société Diatelic et INRIA et il vise, notamment à développer un assistant robotique :

- permettant d'une part l'accès à des techniques de communication avancées pour faciliter le maintien du lien social (famille, amis, soignants, ...)
- et d'autre part offrant un service de télé-vigilance de la personne âgée afin de détecter des situations à risques (chutes) ou des anomalies relevant d'une dégradation de l'état physique ou de santé de la personne suivie.

Dans ce cadre, le programme de recherche plus spécifique de la thèse, cherche à répondre au besoin de disposer d'un dispositif autonome, temps réel, peu coûteux, pouvant être embarqué sur un robot mobile (robot Kompai) et capable de localiser, suivre la personne en mouvement et d'en estimer la pose (posture) y compris lorsque la personne suivie est partiellement occultée par des objets. Le dispositif développé comprend une caméra (du type RGB-D), un calculateur (NUC Intel) et les algorithmes de traitement.

La thèse s'articule ainsi autour de deux contributions complémentaires :

1. La proposition d'un algorithme d'estimation de la posture humaine dans un environnement encombré.
2. La proposition d'un algorithme pour l'estimation de la pose d'une caméra RGB-D en mouvement.

2.1 Robot Kompai

La plate-forme robotique utilisée dans nos travaux est basée sur le robot Kompai développé par la société française Robosoft. Ce robot comme le montre la figure 3 se déplace sur un socle à roulettes capable d'obéir à des ordres vocaux simples en utilisant un module de reconnaissance

vocale développé par Microsoft. Ce robot est équipé de différents capteurs : télémètre Laser, web cam, capteurs infra-rouges, ultra-son et une caméra RGB-D. Le robot est équipé aussi d'un écran tactile avec une interface graphique (figure 4) permettant d'accéder à une variété de services développés tant par Robosoft que par les partenaires du projet LAR : météo, web, vidéos, jeux, appels vocaux, vidéo conférence.

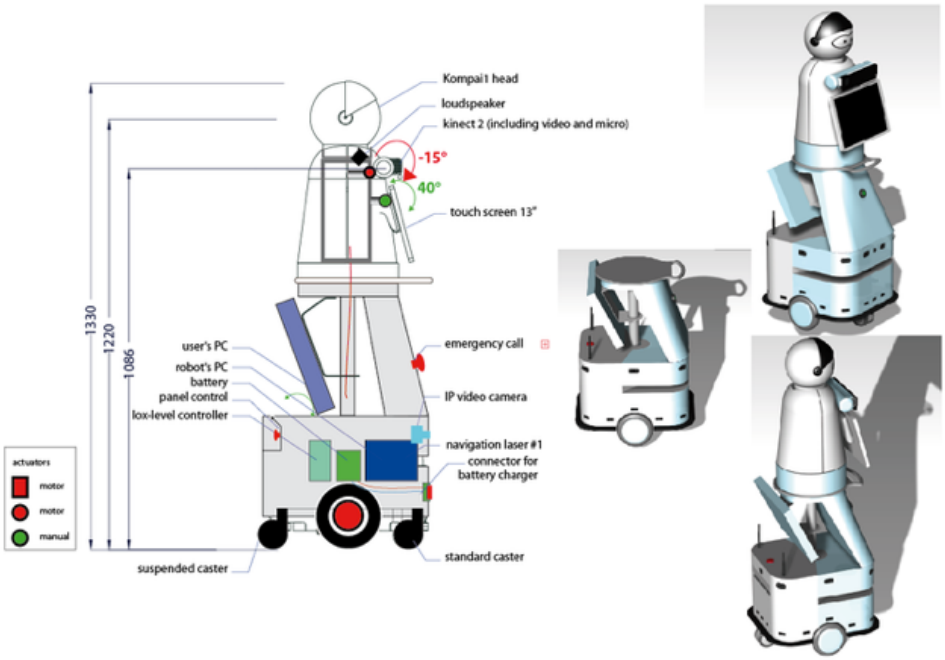


FIGURE 3 – Robot Kompai développé par la société Robosoft.

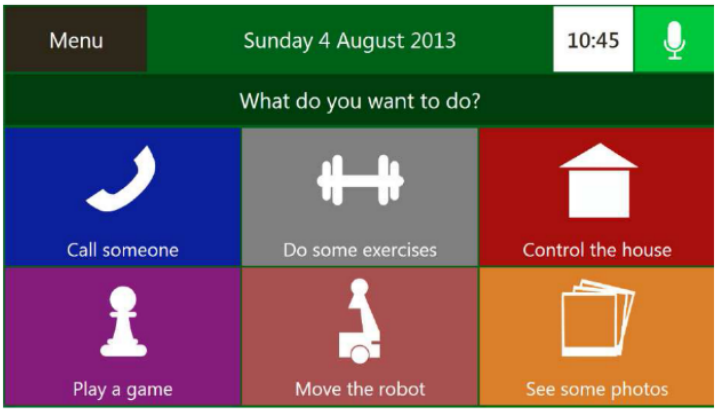


FIGURE 4 – L'interface utilisateur du robot Kompai qui permet de contrôler le robot.

2.2 Camera RGB-D

Le robot Kompai est équipé d'un capteur actif du type RGB-D qui fournit à la fois une image de profondeur et une image RGB. La commercialisation de ce type de capteur pour les jeux vidéo est une opportunité pour développer de nouvelles fonctions en robotique d'assistance

comme le montre le nombre de projets de recherche allant dans cette voie. Ce type de capteur est particulièrement bien adapté à la problématique de la surveillance à domicile, car il est à la fois peu coûteux et très performant. Ces capteurs dédiés initialement à l'interaction par le geste dans les jeux vidéo, trouveront naturellement leur place dans notre contexte applicatif.

Dans cette thèse, nous avons choisi de nous concentrer sur ce capteur pour le développement des différents algorithmes de traitement.

3 Problématique et contributions

Le développement d'un système de perception visuelle de la posture humaine pour un robot se déplaçant dans un environnement dynamique et encombré, est une tâche difficile et complexe. L'objectif de notre travail est de répondre essentiellement les trois questions suivantes :

1. Extraire de la scène l'information pertinente pour reconstruire la posture.
2. Suivre la posture de la personne dans le temps et l'espace.
3. Localiser la caméra en mouvement dans l'environnement.

Pour la première question, la difficulté est de séparer de l'image, les pixels qui correspondent à la silhouette de la personne de ceux qui appartiennent au décor. Le fait que la caméra n'est pas fixe d'une part, et d'autre part, la dynamique de la scène qui évolue au cours du temps, rendent cette tâche difficile. Les méthodes classiques de segmentation, qui supposent que le fond est fixe et procèdent à une soustraction de l'arrière-plan pour extraire la silhouette de la personne, ne sont pas adaptées. **La première contribution** de ce travail est la proposition d'une méthode capable de modéliser en 3D un environnement dynamique et d'identifier les parties fixes et mobiles de la scène pour ensuite extraire la silhouette de la personne. Cette contribution a été présentée dans une conférence internationale [Dubois *et al.*, 2011].

Pour la seconde question, la difficulté est de reconstruire le mouvement de la personne en 3D à partir de sa silhouette en sachant que certaines parties du corps peuvent être occultées par des objets ou obstacles présents dans la scène. Ce problème de reconstruction de posture dans un environnement encombré avec occlusions est très peu traité dans la littérature et reste un défi scientifique ouvert. La majorité des méthodes existantes supposent que la personne se déplaçant dans la scène est toujours visible en entier à chaque instant. Certaines méthodes traitent indirectement le problème des occlusions en utilisant un dispositif à plusieurs caméras. Mais même avec plusieurs caméras, les occlusions sont inévitables. **La deuxième contribution** de cette thèse, porte sur le développement d'une méthode d'extraction et de suivi de la posture humaine dans un environnement encombré. L'originalité de cette méthode est sa capacité à s'adapter aux occlusions générées par les objets de la scène masquant certaines parties du corps. **Une troisième contribution** liée à cette partie est le développement d'un *benchmark* composé d'un ensemble de séquences vidéo enregistrées avec une caméra RGB-D filmant une personne qui se déplace dans différents types de scènes encombrées, contenant des obstacles masquant certaines parties du corps. La vérité terrain est obtenue par un système externe de capture de mouvement utilisant des marqueurs réfléchissants fixés sur la personne. Ce *benchmark* est disponible en ligne sur le lien suivant [motion capture dataset, 2015], et il est mis à disposition de la communauté. Cette contribution a été présentée dans une conférence internationale [Dib and Charpillet, 2015a].

Pour la dernière question, la difficulté principale ici est liée à la dynamique de l'environnement qui rend la localisation difficile. En effet, dans un milieu dynamique, le déplacement des objets présents dans la scène induit une difficulté supplémentaire pouvant nuire à l'estimation du mouvement de la caméra. Le problème de localisation d'une caméra dans un environnement dynamique reste un défi scientifique ouvert. Dans la littérature, peu de travaux traitent l'aspect dynamique de la scène, et la majorité des approches existantes suppose que la scène est statique. **La dernière contribution** de cette thèse, est la proposition d'une nouvelle méthode de localisation d'une caméra mobile dans un environnement dynamique. L'originalité de cette méthode est sa capacité à éliminer les pixels de l'image qui appartiennent à des objets mobiles, dans le but de fournir une estimation précise du mouvement de la caméra. Cette contribution a été présentée dans une conférence internationale [Dib and Charpillet, 2015b].

Notons que dans cette thèse, nous ne traitons pas le problème de la navigation du robot.

4 Organisation du mémoire

Ce mémoire est organisé en 4 parties. Après cette première partie (ouverture), nous développons les parties suivantes :

La partie II traite les deux premiers problèmes déjà ciblés auparavant (voir paragraphe "Problèmatique et contributions"), à savoir, l'extraction de la silhouette et le suivi 3D du mouvement humain dans un environnement en présence d'occlusions. Elle est divisée en 5 chapitres :

1. Chapitre 1 : nous exposons les différentes méthodes existantes de capture de mouvement et nous expliquons comment elles traitent les problèmes cités ci-dessus. Nous expliquons notre choix des approches stochastiques par filtrage particulaire pour le développement de notre méthode de suivi du mouvement humain.
2. Chapitre 2 : nous faisons un rappel sur le filtrage particulaire en général et de son application pour le suivi du mouvement humain.
3. Chapitre 3 : nous décrivons la méthode que nous avons développée pour modéliser un environnement dynamique et extraire la silhouette de la personne.
4. Chapitre 4 : nous décrivons la méthode de suivi du mouvement humain que nous avons développée dans cette thèse. L'originalité de cette méthode est sa capacité à s'adapter aux occlusions générées par les objets de la scène masquant certaines parties du corps.
5. Chapitre 5 : nous décrivons d'abord la base de données étiquetées (*benchmark*) que nous avons développée et mis en ligne sur le lien suivant [motion capture dataset, 2015]. Cette base va servir à évaluer d'une façon quantitative et qualitative le système de suivi du mouvement humain développé.

La partie III traite le troisième problème, déjà identifié dans le paragraphe "Problèmatique et contributions", qui est la localisation d'une caméra mobile dans un environnement dynamique. Elle est divisée en 4 chapitres :

1. Chapitre 6 : nous faisons un état de l'art sur les différentes approches d'odométrie visuelle existantes.
2. Chapitre 7 : nous décrivons les outils théoriques dont nous avons besoin pour le développement de notre méthode d'odométrie visuelle.
3. Chapitre 8 : nous décrivons la méthode de localisation visuelle que nous avons développée dans cette thèse, capable d'estimer correctement le mouvement de la caméra dans un environnement dynamique.

4. Chapitre 9 : nous évaluons la performance de notre méthode avec une série d'expériences que nous avons réalisée et avec une base de données étiquetées, disponible en ligne.

La partie IV décrit le système complet de perception de la posture humaine à partir d'une caméra montée sur un robot mobile. Ce système intègre les différentes méthodes développées dans cette thèse, à savoir, l'extraction de la silhouette, le suivi du mouvement humain et l'odométrie visuelle. Nous discutons les premiers résultats obtenus suite à une expérience que nous avons réalisée consistant à filmer avec une caméra montée sur un robot mobile, une personne se déplaçant dans un environnement avec occlusions. Cette expérience nous a permis d'identifier les limites du système proposé ainsi que les difficultés rencontrées.

En conclusion, nous proposons certains perspectives d'améliorations futures.

Deuxième partie

Extraction de la silhouette et suivi 3D du mouvement humain

Introduction

Notre objectif est de doter un robot assistant mobile, des capacités de perception visuelle de la posture humaine. La reconstruction de la posture est une étape fondamentale pour toute application d'interaction et d'assistance. Elle permet de mieux maîtriser certaines situations, par exemple, détecter une situation à risque comme une chute ou analyser les capacités motrices de la personne.

Dans des conditions réelles, et à cause de la dynamique de la scène, plusieurs difficultés existent et notamment les occlusions et l'évolution de la scène au court du temps. La majorité des méthodes existantes ignorent ces problèmes et traitent le suivi dans des situations contrôlées, en supposant que la scène est statique, et que la personne qui se déplace est toujours visible en entier à n'importe quel instant par la caméra. Ces situations parfaites sont loin de répondre aux besoins d'une analyse à domicile. Certains travaux de la littérature traitent indirectement les occlusions présentes dans la scène en utilisant un système multi-caméra. Cependant, même avec un nombre élevé de caméras, les occlusions restent inévitables.

Notre motivation principale est de développer un système capable de suivre le mouvement de la personne dans un environnement dynamique et encombré. La contribution majeure de cette partie, est la proposition d'une méthode de capture de mouvement robuste dans des conditions réelles. Le système de capture de mouvement proposé aborde les problèmes suivants :

1. Dynamique de la scène : la difficulté principale dans un environnement dynamique est de pouvoir extraire la silhouette de la personne de l'image. Nous supposons qu'un modèle de fond n'est pas disponible, ainsi les méthodes classiques de soustraction de fond ne sont pas applicables. Nous proposons alors une nouvelle méthode capable d'apprendre en ligne le fond de la scène et d'extraire la silhouette de la personne dans un environnement dynamique.
2. Occlusions : les objets présents dans la scène masquent certaines parties du corps de la personne rendant ainsi l'extraction de la posture difficile. Nous proposons une nouvelle méthode d'extraction de posture capable de gérer les situations où la silhouette de la personne est partiellement observable.

Par ailleurs, nous avons construit une base de référence qui contient des données étiquetées d'une personne se déplaçant dans un environnement avec occlusion. La vérité terrain est obtenue par un système externe de capture de mouvement basé sur des marqueurs lumino-réfléchissants Qualisys. Cette base est disponible en ligne sur le lien suivant [motion capture dataset, 2015] et elle est mise à disposition de la communauté.

Dans la suite de cette partie, le chapitre 1, fait le panorama des méthodes de capture de mouvement. Nous distinguons deux types de méthodes, les premières basées sur la fixation des marqueurs ou capteurs sur les différentes parties du corps et les deuxièmes utilisant uniquement une ou plusieurs caméras fixes ou mobiles. Ensuite, le chapitre 2 fait un rappel sur le filtrage particulaire et son application pour le suivi du mouvement humain. Dans le chapitre 3, nous

décrivons la méthode que nous avons développée pour extraire la silhouette de la personne dans un environnement dynamique. Dans le chapitre 4, la méthode que nous avons développée pour reconstruire la posture à partir d'une silhouette partiellement observable dans un environnement avec occlusion sera décrite. Finalement, dans le chapitre 5, nous évaluons le système de suivi du mouvement humain sur notre base de données.

Chapitre 1

Méthodes existantes pour la capture du mouvement humain

Sommaire

1.1 Capture de mouvement par les méthodes à base de marqueurs ou capteurs fixés sur le corps 14

1.2 Capture de mouvement à partir d'une ou plusieurs caméras fixes 14

1.2.1 Panorama des méthodes existantes 15

1.2.2 Discussion 17

1.2.3 Méthodes basées sur l'apprentissage (Méthode de Microsoft) 17

1.2.4 Méthodes stochastiques utilisant le filtrage particulaire 19

1.3 Capture de mouvement à partir d'une caméra mobile 20

1.4 Conclusion 20

La capture du mouvement humain (en anglais Human Motion Capture HMC) consiste à reconstruire, à partir d'un certain nombre de capteurs, les positions et les orientations des différents membres du corps pour obtenir un modèle de la posture de type fils de fer ou squelette. Le squelette reconstruit est utilisé dans une variété d'application notamment les jeux vidéos, cinéma et dans le domaine médical pour analyser la qualité de la marche des personnes. Dans le domaine robotique, l'extraction du squelette est principalement utilisée pour l'interaction homme machine pour permettre au robot de mieux comprendre l'intention de la personne, d'interagir avec elle et aussi de surveiller son activité.

Dans ce chapitre, nous exposons, les différentes méthodes existantes pour extraire le squelette. Ces méthodes sont classées en deux catégories : la première utilise des capteurs ou des marqueurs fixés sur la personne, et la deuxième n'a pas besoin de capteurs ni de marqueurs portables, elle utilise uniquement un flux vidéo correspondant au mouvement de la personne. Cette dernière catégorie quand à elle, peut être divisée en deux sous catégories, la première utilise une ou plusieurs caméras fixes, et la seconde utilise une caméra mobile. Dans la conclusion de ce chapitre, nous expliquons notre choix des méthodes stochastiques par filtrage particulaire pour le développement de la méthode de suivi du mouvement humain.

1.1 Capture de mouvement par les méthodes à base de marqueurs ou capteurs fixés sur le corps

Ces méthodes sont basées sur le positionnement des capteurs ou marqueurs sur les différentes parties du corps de la personne. Plusieurs systèmes utilisant cette technique existent et nous distinguons deux méthodes. La première utilise des marqueurs lumino-réfléchissants fixés à des endroits clés de la personne. Ces marqueurs passifs sont observés et suivis par des caméras infrarouges. Plusieurs systèmes commerciales existent actuellement et utilisent cette technique notamment les systèmes Vicon, Motion Analysis, Optitrack et Qualisys (Nous disposons au LORIA, dans la plate-forme habitat intelligent d'un système Qualisys). La deuxième technique existante consiste à utiliser des marqueurs actifs qui comportent une source de lumière à base de LED. Cette technique est utilisée par plusieurs systèmes telles que Phoenix et Phasespace. Le fonctionnement de ces méthodes repose sur l'utilisation de plusieurs caméras infrarouges. Une phase de calibration des différentes caméras est nécessaire dans un premier temps. Ensuite, chaque marqueur réfléchit le signal émis (pour le cas des système passifs) ou émet un signal infrarouge (pour les systèmes actifs). Ce signal n'est capté que par les caméras ayant une ligne de vue directe sur ce marqueur. C'est pour cela, qu'un tel système a besoin d'un nombre de caméras élevé pour garantir un suivi non interrompu. Ce nombre augmente avec la dimension de l'espace à couvrir. La position 3D de chaque marqueur est obtenue par une méthode de triangulation. Cette triangulation n'est possible que si le marqueur est visible par au moins deux caméras à un instant donné.

La deuxième catégorie de système de capture de mouvement et d'extraction de la posture repose sur l'utilisation d'accéléromètres et/ou de gyroscopes 3 axes qui sont fixés sur les différentes parties du corps. Ces capteurs renvoient la position et l'orientation en sortie. Quelques solutions commerciales existent actuellement, notamment celle de la société française Tea, les sociétés Metamotion et XSens. La mise en place d'un tel système nécessite une phase d'étalonnage au repos. Une étape de filtrage et de traitement de signal est réalisé pour éliminer le bruit du signal brut.

1.2 Capture de mouvement à partir d'une ou plusieurs caméras fixes

Le principe de fonctionnement de ces méthodes consiste à estimer le mouvement 3D de la personne à partir d'un flux d'images provenant d'une ou plusieurs caméras. La structure générale qu'utilise ces méthodes consiste d'abord à segmenter l'image d'entrée pour extraire la personne de la scène. L'image segmentée obtenue est transformée en une autre représentation plus compacte et mieux adaptée pour chaque algorithme : silhouette, contours, ou un nuage de points 3D par exemple. Finalement, le suivi est appliqué pour en extraire les différentes parties du corps en utilisant la représentation de l'image d'entrée comme une fonction de vraisemblance. La sortie typique d'un système de capture de mouvement est un squelette comme le montre la figure 1.1.

Le capteur utilisé par ces méthodes est une caméra couleur classique ou une caméra RGB-D. En effet, le suivi du mouvement humain à partir des caméras restent un sujet de recherche actif et très peu de solutions commerciales existent actuellement. Nous avons identifié deux *startup* qui proposent des systèmes de capture de mouvement à base de plusieurs caméras RGB : la société Organic Motion et The Captury. Ces systèmes nécessitent une phase de calibration des différentes caméras en utilisant des techniques classiques de vision. Un nombre élevé de caméras est nécessaire pour couvrir tout l'espace de travail. Un autre système proposé par Microsoft pour

la console de jeu Xbox utilise une caméra RGB-D pour reconstruire le squelette de la personne. Ce système est orienté pour les jeux vidéo et nécessite que la personne reste toujours en face de la caméra.

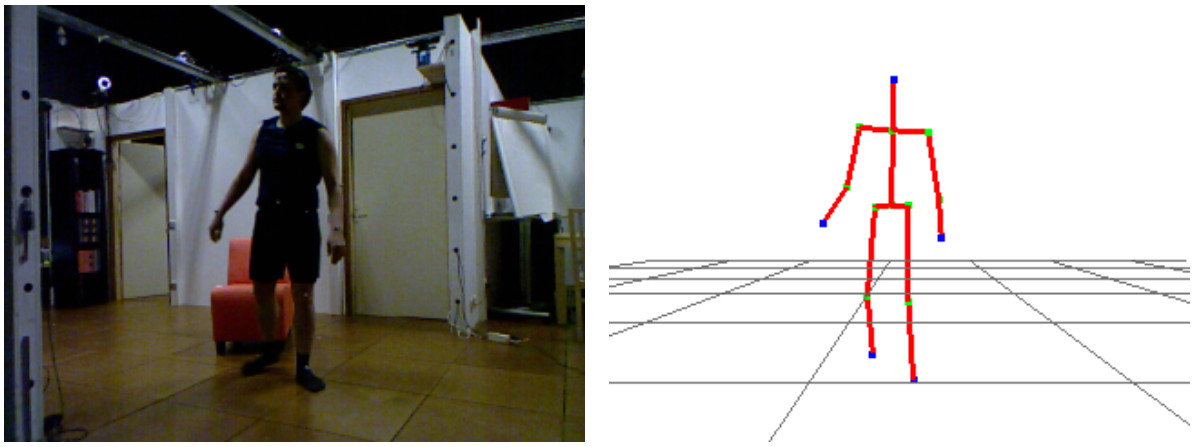
D'après la littérature, nous pouvons citer, deux grandes études faites par Moeslund *et al.* [Moeslund *et al.*, 2006] et Poppe [Poppe, 2007] qui résument les principaux travaux et avancés sur ce sujet. Deux approches sont actuellement prédominantes pour traiter le problème de la capture de mouvement. La première approche dite **sans modèle** (en anglais *Model Free*) n'a pas besoin d'un modèle à priori de l'être humain, et nous citons principalement dans cette famille les méthode de type *Machine Learning* ou d'apprentissage. Ces méthodes consistent à apprendre le mouvement humain hors ligne à partir d'un dictionnaire construit avec des images synthétiques ou réelles. La deuxième approche **basée modèle** (en anglais *Model Based*) qui, quand à elle utilise un modèle *a priori* de l'être humain. Dans cette famille, nous citons principalement les méthodes probabilistes consistant à prédire en ligne la pose et plus précisément, les méthodes stochastiques qui utilise le filtrage particulaire.

Alors que le domaine de la capture de mouvement humain est un sujet bien établi et étudié dans la littérature, très peu de travaux traitent le problème de suivi dans des conditions réelles. En effet, la majorité des méthodes existantes ont été testées et validées dans des environnements contrôlés où la scène est supposée statique et la personne se déplaçant devant la caméra est toujours visible en entier. De telles conditions ne sont pas réalisables dans le cadre de notre projet qui consiste à développer un assistant robotique dans un environnement quotidien. Dans la vie quotidienne, l'environnement est évolutif : les meubles bougent, la lumière change, l'emplacement des capteurs n'est pas forcement optimal,... Ainsi la première difficulté dans ces conditions est de pouvoir segmenter la silhouette de la personne de l'image. Une deuxième difficulté est liée aux occlusions causées par la présence des objets dans la scène pouvant ainsi masquer certaines parties du corps. Ces occlusions sont inévitables et il faut donc trouver une méthode capable de détecter et gérer les situations où la personne n'est pas visible entièrement. Dans l'étude faite par Poppe [Poppe, 2007], l'auteur a mentionné que le suivi 3D dans un environnement avec occlusion reste un *challenge* et peu de travaux traitent ce problème. Dans la section suivante, nous faisons un panorama des méthodes de captures de mouvement existantes et comment elles traitent les problèmes cités ci-dessus.

1.2.1 Panorama des méthodes existantes

Parmi les premiers travaux traitant des occlusions, nous pouvons citer les travaux de Lee *et al.* [Lee *et al.*, 2002] qui utilisent un filtre particulaire ([Arulampalam *et al.*, 2002]) avec une inférence analytique pour réduire le nombre de degrés de liberté et pour pouvoir restaurer le suivi après une occlusion. Pour fonctionner, leur méthode doit utiliser une détection analytique très précise, ce qui est très difficile à obtenir. Une autre approche traitant des occlusions est celle de Peursum *et al.* [Peursum *et al.*, 2007] en 2007, qui utilise un module de reconnaissance d'actions pour assister et guider leur algorithme de suivi pendant les occlusions. Les résultats obtenues sont très grossiers et peu précis, comme indiqués par les auteurs eux mêmes.

Lee et Nevatia [Lee and Nevatia, 2009] proposent en 2009, un système complexe à trois étages pour gérer les occlusions qui se produisent suite à l'interaction entre plusieurs personnes dans l'image. Pour cela, ils extraient pour chaque personne un *blob*, puis un histogramme est appris pour chaque *blob* dans le but de détecter les occlusions en observant un changement dans la forme du *blob* appris. L'inconvénient de cette méthode est le temps de traitement qui peut atteindre 5 minutes par image comme indiqué par les auteurs. Plus récemment, les approches de Stoll *et al.* [Stoll *et al.*, 2011] et Liu *et al.* [Liu *et al.*, 2013] utilisent plusieurs caméras pour gérer



(a) (b)

FIGURE 1.1 – Principe de la capture de mouvement à partir d'une caméra (image prise de notre système) : a) Une personne filmée par une caméra. b) La reconstruction du mouvement de la personne représentée ici par un squelette en fil de fer.

les occlusions. Un ensemble de 12 caméras est utilisé. La difficulté de ce type d'approche est la nécessité d'un nombre élevé de caméras difficile à mettre en place.

Avec l'apparition des caméras RGB-D en 2011, les approches basées sur les algorithmes d'apprentissage de type forêts aléatoires (en anglais *Random Forests* [Ho, 1995]) produisent des résultats intéressants. Nous citons principalement les travaux de Shotton *et al.* [Shotton *et al.*, 2011] et Lallemand2013 *et al.* [Lallemand *et al.*, 2013]. La robustesse de ces méthodes n'a pas été évaluée dans des situations avec occlusion. En 2015, Rafi *et al.* [Rafi *et al.*, 2015] montrent que les méthodes basées sur les forêts aléatoires donnent de mauvais résultats dans des situations d'occlusion, et pour les améliorer, les auteurs intègrent un modèle sémantique d'occlusion. D'autres types d'approches par apprentissage existent, notamment, celles qui utilisent du *Deep Learning* ([Lecun *et al.*, 1998]). En effet, le *Deep Learning* est actuellement un sujet très actif, mais qui nécessite une puissance de calcul importante ainsi que des gros volumes de données. Appliqué au domaine de la capture du mouvement humain, il produit des résultats intéressants (par exemple, voir Jiu *et al.* [Jiu *et al.*, 2014]). Cependant, dans l'article précédemment cité, les auteurs ne traitent pas le problème lié aux occlusions.

D'autres méthodes qui utilisent l'information provenant d'une caméra 3D, comme les travaux de Ganapathi *et al.* [Ganapathi *et al.*, 2012] et Wei *et al.* [Wei *et al.*, 2012] produisent des résultats précis et en temps réel. Par contre, les auteurs ne traitent pas les occlusions et supposent que le fond est fixe et procèdent à une soustraction de fond classique. En 2012, Zhang *et al.* [Zhang *et al.*, 2012] fusionnent les images de profondeurs de plusieurs caméras 3D en un seul nuage de points commun et utilisent une fonction de distance tronquée comme observation dans leur filtre particulaire. Les auteurs affirment que l'utilisation d'une seule caméra 3D est très sensible aux occlusions et en utilisant plusieurs caméras, ils arrivent à réduire l'ambiguïté. En 2014, Ghiasi *et al.* [Ghiasi *et al.*, 2014] proposent une méthode d'apprentissage supervisé pour apprendre les situations d'occlusion ce qui leur permet d'apprendre l'apparence des différents types d'occlusions. Cela nécessite une grande base de données pour apprendre tous les types d'occlusions qui peuvent se produire dans des environnement très variés.

1.2.2 Discussion

La recherche bibliographique que nous avons effectuée montre qu'il existe actuellement de nombreux méthodes pour l'extraction de la posture humaine, mais la majorité de ces méthodes sont testées dans des environnement contrôlés. Certaines méthodes traitent indirectement le problème des occlusions en utilisant plusieurs caméras. Mais l'utilisation de plusieurs caméras sur un robot n'est pas envisageable pour nous et même avec plusieurs caméras les occlusions sont toujours présents. Dans cette thèse, nous proposons une adaptation d'une méthode existante pour la rendre robuste aux problèmes rencontrés dans des conditions réelles, à savoir, l'extraction de la silhouette et la gestion des occlusions. Pour le choix de la méthode que nous souhaitons adapter, les méthodes basées sur l'apprentissage hors-ligne sont intéressantes et précisément celle proposée par Microsoft. Les approches stochastiques en ligne basée sur le filtrage particulaire sont à considérer. Dans la suite nous décrivons brièvement le fonctionnement de la méthode de Microsoft et des méthodes basées sur le filtrage particulaire. Le choix de la méthode sera fait dans la conclusion de ce chapitre.

1.2.3 Méthodes basées sur l'apprentissage (Méthode de Microsoft)

La méthode d'extraction de la posture développée par Microsoft ([Shotton *et al.*, 2011]) appartient à la famille des approches sans modèle. Cette méthode consiste à apprendre les mouvements humains hors-ligne à partir d'un dictionnaire construit avec des images synthétiques et réelles. Cette méthode est inspirée des modèles appelés *Pictorial Structure* introduits en 2005 par Felzenszwalb et Huttenlocher [Felzenszwalb and Huttenlocher, 2005] qui consistent à estimer la position des différentes parties d'un objet dans une image en respectant les relations entre ces parties. Cette solution est bien adaptée pour l'estimation de la position du corps humain composé d'un ensemble de segments reliés entre eux. Ainsi, Microsoft re-formalise le problème d'estimation de pose en un problème de classification qui consiste à identifier pour chaque pixel de l'image la partie du corps à laquelle il appartient. Le classificateur utilisé est de type forêt aléatoire ([Breiman, 2001]) qui est un algorithme d'apprentissage automatique, rapide et qui peut être facilement parallélisable sur les architectures modernes comme les GPU (*Graphical Processor Unit*) d'aujourd'hui. Pour obtenir des résultats satisfaisants, Microsoft a généré une grande base d'apprentissage constituée de million d'images synthétiques générées par ordinateur. Les pixels de chaque image sont étiquetés pour savoir à quelle partie du corps ils appartiennent. Le corps humain est divisé en 31 parties comme le montre la figure 1.2. La base doit couvrir toutes les configurations et les mouvements possibles que la personne peut faire, y compris les changements de textures et de couleurs. Grâce à l'image de profondeur retournée par la caméra RGB-D, cette tâche est simplifiée car l'image de profondeur retournée par la caméra est invariante aux changements de couleur et texture.

Les caractéristiques (en anglais *features*) utilisées pour classifier les pixels sont basées sur une soustraction de profondeur entre des pixels de l'image. Soit $\theta_i = (u_i, v_i)$ une caractéristique i parmi N. u_i et v_i définissent un offset dans l'image. Pour un pixel donné x, la caractéristique est calculée comme suit :

$$f_{\theta_i}(I, x) = d_I(x + \frac{u_i}{d_I(x)}) - d_I(x + \frac{v_i}{d_I(x)}), \tag{1.1}$$

avec $d_I(x)$ la valeur de profondeur pour le pixel x dans l'image I. La forêt aléatoire est composée de T arbres. Chacun de ces arbres est composé d'une division et de feuilles. Chaque division est composée d'une caractéristique f_{θ_i} et d'un seuil σ_i. Pour classifier un pixel, l'algorithme commence par la racine d'un arbre, l'équation 1.1 est évalué à chaque division, pour savoir

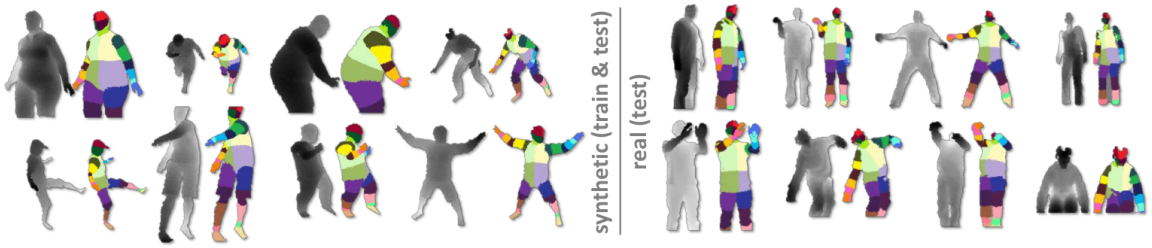


FIGURE 1.2 – Un extrait de la base d'apprentissage utilisée par Microsoft pour apprendre leur classificateur. A gauche les images synthétiques et à droite les images réelles. Pour chaque image de profondeur (en niveau de gris), un étiquetage est fait pour chaque pixel pour identifier la partie du corps à laquelle ce pixel appartient.

s'il faut brancher à gauche ou à droite de l'arbre suivant le résultat de la comparaison de la caractéristique (*feature*) au niveau du pixel par rapport au seuil σ_i. Une fois que le parcours d'un arbre t est terminé, la valeur de probabilité de distribution $P_t(c \mid I, x)$ est apprise, définissant la probabilité que ce pixel appartienne à une partie du corps notée c. La classification finale d'un pixel sur l'ensemble des arbres est égale à la valeur moyenne des probabilités de distribution pour tous les arbres :

$$P(c \mid I, x) = \frac{1}{T} \sum_{t=1}^{T} P_t(c \mid I, x).$$

La dernière étape consiste à extraire la position des différentes articulations du squelette à partir des pixels classifiés en utilisant une méthode de regroupement basée sur l'algorithme de *Mean Shift* ([Comaniciu *et al.*, 2002]). La figure 1.3 montre le processus final qui résume les différentes étapes de la méthode.

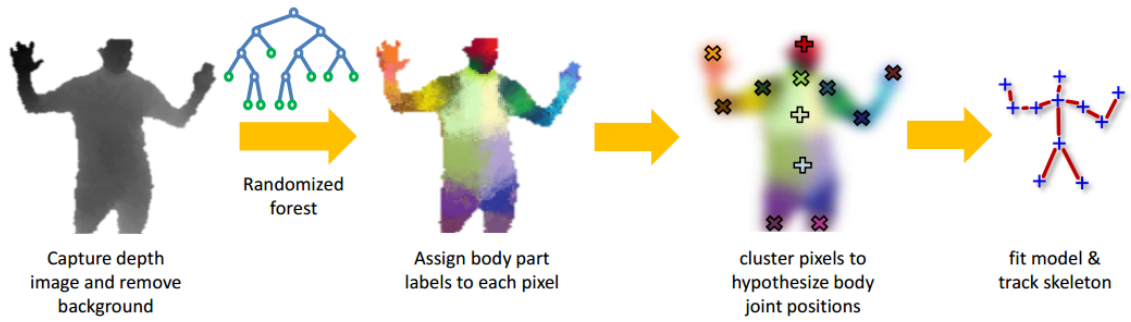


FIGURE 1.3 – Le processus de la méthode de capture de mouvement développée par Microsoft. Après soustraction du fond, une classification en utilisant les forêt aléatoires est appliquée pour classifier tous les pixels de l'image. Finalement, une extraction de la position des différentes articulations est faite en utilisant une méthode de regroupement.

Limitations dans des situations d'occlusion

Rafi *et al.* [Rafi *et al.*, 2015], ont montré que la méthode décrite ci-dessus est très sensible aux occlusions et produit des mauvais résultats dans ces situations comme le montre la figure

1.4. Ce résultat n'est pas surprenant puisque la base construite par Microsoft ne contient pas de personnes en situation d'occlusion.

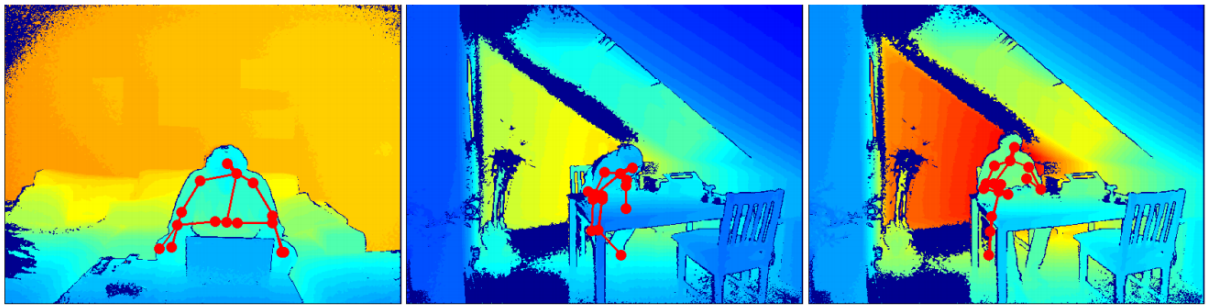


FIGURE 1.4 – L'estimation de pose par la méthode de Microsoft [Shotton *et al.*, 2011] dans des situations avec occlusions. Les résultats montrent que la méthode ne réussit pas à estimer correctement la pose de la personne dans ces situations.

1.2.4 Méthodes stochastiques utilisant le filtrage particulaire

Les approches stochastiques à base de filtrage particulaire ([Arulampalam *et al.*, 2002]) qui appartiennent à la famille des approches basées modèle, consistent à approximer l'espace d'état par un ensemble de particules. Ces approches sont prometteuses et donnent des résultats intéressants pour le suivi du mouvement humain. Les filtres particulaire sont bien adaptés pour modéliser l'incertitude dans ce contexte. Le fait de garder plusieurs hypothèses au cours du temps permet au système d'améliorer l'estimation de pose. Le filtre particulaire approxime l'espace d'état complexe du mouvement humain par un ensemble de particules dont chacune représente une hypothèse sur la configuration de la personne. Ces particules sont évaluées par rapport à l'image observée de la caméra en utilisant une fonction de vraisemblance qui affecte un poids ou un score à chaque particule. Finalement, les particules qui obtiennent des scores élevés sont gardées pour la suite. Par contre, cette méthode nécessite un nombre très élevées de particules pour le suivi ce qui nécessite une grande puissance de calcul pour évaluer toutes les particules, rendant l'implémentation loin du temps réels.

Dans le but d'optimiser l'algorithme de filtrage particulaire original, Deutscher *et al.* [Deutscher *et al.*, 2000] proposent une version qui réduit le nombre de particules en utilisant le principe du recuit-simulé. Peursum *et al.* [Peursum *et al.*, 2007] montrent que le suivi du mouvement humain avec cette méthode donne des mauvaises résultats dans les situations d'occlusion. Une autre approche pour réduire le nombre de particules proposée par MacCormick et Isard [MacCormick and Isard, 2000], similaire à celle proposée par Gavrila et Davis [Gavrila and Davis, 1996], qui consiste à représenter la structure cinématique du corps humain sous forme factorisée. Cette factorisation d'espace d'état permet de réduire considérablement le nombre de particules requis. Rose *et al.* [Rose *et al.*, 2008] formalisent le problème du suivi du mouvement humain dans un réseau bayésien dynamique ([Murphy, 2002]) et utilisent un filtre particulaire pour l'inférence dans ce réseau. Bandouch et Beetz [Bandouch and Beetz, 2009] proposent une méthode de suivi du mouvement humain qui combine la méthode basée sur le recuit-simulé et celle qui factorise l'espace d'état. Les auteurs de cette méthode traitent les occlusions en utilisant un système à 4 caméras et en étiquetant manuellement les zones de l'image susceptibles de générer des occlusions avant de lancer leur système de suivi. Cette méthode suppose que la scène est fixe et que les endroits pouvant générer des occlusions ne vont pas changer lors du suivi. Saboune et Charpillet

([Saboune and Charpillet, 2005a], [Saboune and Charpillet, 2005b]) proposent un algorithme de filtrage particulaire avec une exploration déterministe de l'espace d'état. Les auteurs montrent que cet algorithme permet de réduire le nombre de particules nécessaire pour le suivi.

1.3 Capture de mouvement à partir d'une caméra mobile

Alors que le problème de la capture de mouvement à partir d'un ensemble de caméra fixes est bien étudié dans la littérature, peu de travaux traitent le problème d'un point de vue robotique, c-à-d à partir d'une caméra montée sur un robot mobile. Dans cette section, nous décrivons les principales méthodes que nous avons identifiées dans la littérature qui s'adressent à ce problème, plus précisément nous décrivons comment ces méthodes traitent les problèmes que nous avons identifiés, à savoir l'extraction de la silhouette et la gestion des occlusions.

Les premiers travaux que nous avons identifiés datent de 2007, avec Azard *et al.* [Azad *et al.*, 2007] qui proposent un système de stéréo-vision passif pour capturer le mouvement de la partie supérieure de la personne situant en face d'une tête robotique motorisée. Pour extraire la silhouette de la personne, les auteurs proposent une alternative à la méthode de soustraction de fond classique qui consiste à supposer que la personne est toujours habillé d'un T-shirt de couleur uniforme et avec une méthode de segmentation, ils arrivent à extraire la partie supérieure de la personne. Les auteurs ne s'adressent pas au problème des occlusions puisque leur système est orienté pour l'interaction directe entre le robot et l'homme et ils ne s'intéressent qu'à la partie supérieure du corps qui est supposée toujours visible.

Les travaux de Hasler *et al.* [Hasler *et al.*, 2009] sont intéressants, bien que le domaine d'application de ces travaux ne soit pas la robotique. Les auteurs proposent une méthode de capture de mouvement à partir d'un ensemble de caméras mobiles et non synchronisées. L'approche consiste à procéder d'abord à une étape de synchronisation entre les caméras en utilisant le son audio enregistré par chacune des caméras. Ensuite, la trajectoire de chaque caméra est calculée par une méthode de localisation à base d'extraction de points d'intérêts, et la géométrie 3D de la scène observée par chaque caméra est reconstruite par des méthodes de stéréo-vision classique (*Structure From Motion* [Hartley and Zisserman, 2004]). La transformation rigide entre chaque caméra est calculée pour obtenir la géométrie globale sous forme d'un nuage de point 3D de la scène. Pour l'extraction de la silhouette, l'algorithme repose sur une étape préliminaire d'extraction d'un modèle 3D de la personne en utilisant un scanner 3D, ce modèle est utilisé pour la segmentation de l'image par un processus itératif qui calcule à la fois, la position des différentes parties du corps et l'image segmentée. Les auteurs ne traitent pas dans leur travaux le problème lié aux occlusions.

1.4 Conclusion

Dans ce chapitre nous avons détaillé les principales méthodes de capture de mouvement. Ces méthodes sont séparées en deux catégories : La première utilise un ensemble de marqueurs ou capteurs fixés sur les différentes parties de corps. Ce type d'approche n'est pas envisageable dans notre situations à cause de la complexité de la mise en place d'un tel système, son prix très élevé, et d'autre part, un tel système oblige la personne à porter ces capteurs/marqueurs ce qui n'est pas applicable dans notre cas. La deuxième catégorie utilise uniquement des caméras et n'utilise ni marqueurs ni capteurs portables est intéressante à étudier. Nous avons vu d'après l'état de l'art, que la majorité de ces méthodes traitent le problème de suivi dans un environnement contrôlé. En effet, ces méthodes supposent que la scène est statique et utilisent une méthode

d'extraction de fond classique qui consiste à supposer qu'à l'initialisation le fond est fixe et la personne est en dehors de la scène. Une fois que la personne entre dans la scène, une soustraction de fond est appliquée pour extraire la silhouette. En plus, ces méthodes ne traitent pas les occlusions générées par des objets de la scène. Cette situation arrive très fréquemment dans un environnement du quotidien souvent encombré où la personne peut être dernière son bureau, assise sur une chaise etc. Certains travaux comme nous l'avons montré traitent le problème des occlusions indirectement en utilisant plusieurs caméras. Mais même avec plusieurs caméras, les occlusions sont inévitables.

Notre objectif est de fournir un suivi dans des conditions de vie quotidienne où la scène évolue en permanence, d'où la nécessité d'avoir une méthode capable d'extraire la silhouette de la personne se déplaçant dans une scène dynamique et de suivre son mouvement en présence des occlusions. Notre objectif est de reprendre une méthode existante et de l'adapter pour la rendre robuste aux problèmes liés aux occlusions et à la dynamique de la scène.

Pour le choix de la méthode, nous avons étudié les méthodes basées sur l'apprentissage hors-ligne utilisant les forêts aléatoires et le *Deep learning* et nous avons présenté comme exemple, l'approche proposée par Microsoft. Cette méthode est orientée pour les jeux vidéos où la personne est devant la console et elle n'est pas adaptée pour suivre l'activité de la personne dans sa vie quotidienne avec notamment la gestion d'occlusion. Microsoft a utilisée une base de millions d'images pour l'apprentissage. Nous avons vu que Rafi *et al.* [Rafi *et al.*, 2015] ont montré que cette méthode donne des mauvais résultats en présence des occlusions. Pour permettre à cette méthode de gérer les occlusions, il est nécessaire d'intégrer dans la phase d'apprentissage des situations d'occlusion, ce qui augmentera la taille de la base pour garantir un suivi fiable, ce qui rend cette méthode difficile à mettre en place pour nous. Une deuxième limitation de cette approche est qu'elle se focalise uniquement sur la détection des parties du corps sans exploiter les contraintes temporelles et mécaniques puisque chaque image est traitée séparément. Par conséquent, elle peut générer des postures non consistantes dans certains cas. Par exemple, elle peut détecter la même partie du corps à plusieurs endroits voir même ne pas détecter certaines parties. Ainsi, dans des situations réelles avec occlusions, le taux de réussite de leur classifieur sera faible et produira un suivi interrompu.

En conclusion, à cause de la complexité du problème que nous cherchons à résoudre, la méthode retenue est basée sur les modèles stochastiques par filtrage particulaire qui consiste à maintenir plusieurs hypothèses dans le temps. Ainsi, la méthode ne rejettera pas d'hypothèses à priori, ce qui permet de reprendre le suivi suite à une perte occasionnelle. Un autre avantage du filtrage particulaire est qu'il ne nécessite pas de connaître une forme analytique de l'espace d'état, cet espace sera approximé par un ensemble de particules. En revanche, un inconvénient de cette méthode est qu'elle nécessite un nombre élevé de particules pour bien approximer l'espace d'état ce qui nécessite une grande puissance de calcul rendant l'implémentation loin du temps réels. Cependant, il existe des variantes de cette méthode que nous avons décrit dans ce chapitre permettant de réduire considérablement le nombre de particules tout en gardant une bonne qualité de suivi.

Dans le chapitre suivant, nous faisons un rappel sur la méthode de filtrage particulaire, et nous décrivons les principales variantes de cette méthode permettant de réduire le nombre des particules nécessaire pour le suivi.

Chapitre 2

Filtrage particulaire : Théorie et application pour le suivi du mouvement humain

Sommaire

2.1 Problème de filtrage 23
2.2 Rappel sur la méthode de Monte Carlo 24
2.2.1 Echantillonnage préférentiel 25
2.3 Echantillonnage préférentiel séquentiel (SIS) 25
2.4 Filtre particulaire 28
2.5 Application du filtrage particulaire pour le suivi du mouvement humain 29
2.5.1 Filtrage particulaire avec recuit-simulé 30
2.5.2 Filtrage particulaire factorisé 32
2.5.3 Approche combinée 34
2.6 Conclusion 35

Dans ce chapitre, nous présentons un bref rappel des techniques d'estimation d'état et de filtrage bayésien. Nous verrons que le filtre particulaire est une approximation du filtrage bayésien adaptée aux problèmes non linéaires, non gaussiens, caractéristiques commune à de nombreux problèmes dont la reconnaissance de la posture humaine. Nous présentons ensuite les deux principales extensions existantes du filtrage particulaire qui améliore considérablement les performances de l'algorithme pour le suivi du mouvement humain. D'autres variantes existent que nous n'abordons pas dans cette thèse ; elles sont décrites dans les thèses de Saboune [Saboune, 2008] et Nguyen [Nguyen, 2013].

2.1 Problème de filtrage

Soit un processus stochastique Markovien de premier ordre décrivant l'évolution de l'état d'un système dynamique suivant l'équation :

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \pi_t), \tag{2.1}$$

où $\mathbf{x}_t$ est un vecteur représentant l'état du système à un instant t, π_t est le bruit sur l'état et f_t est la fonction de transition qui peut être non-linéaire décrivant l'évolution du vecteur d'état.

Le vecteur d'état $\mathbf{x}_t$ est supposé inconnu ou non observable directement, l'information sur $\mathbf{x}_t$ est obtenue à partir des mesures bruitées notées $\mathbf{y}_t$ défini par l'équation :

$$\mathbf{y}_t = h_t(\mathbf{x}_t, \nu_t), \tag{2.2}$$

où h_t est la fonction d'observation qui peut être non-linéaire et ν_t le bruit sur la mesure.

Nous rappelons dans la suite, le problème du filtrage à temps discret d'un processus Markovien de premier ordre. Le filtrage consiste à estimer le vecteur d'état à chaque instant t, connaissant toutes les observations jusqu'à l'instant t notées $\mathbf{y}_{1:t}$. Dans un cadre bayésien, cela revient à estimer la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ d'une façon récursive en deux étapes :

1. **Prédiction** : Dans cette étape, la densité de probabilité $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ est calculée à partir de $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ à $t-1$ supposée connue par récursion :

$$\mathbf{p(x_t \mid y_{1:t-1})} = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \cdot p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) \mathrm{d}\mathbf{x}_{t-1}, \tag{2.3}$$

   où $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ est donnée par l'équation 2.1.

2. **Correction** : Dans cette étape, $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ est corrigée avec la nouvelle mesure $\mathbf{y}_t$ en utilisant la règle de Bayes pour obtenir la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ sur $\mathbf{x}_t$:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t \mid \mathbf{x}_t) \cdot \mathbf{p(x_t \mid y_{1:t-1})}, \tag{2.4}$$

   où $p(\mathbf{y}_t \mid \mathbf{x}_t)$ est donnée par l'équation 2.2.

Les deux équations récurrentes de prédiction 2.3 et de correction 2.4 forment la base d'une solution bayésienne optimale. La propagation récursive de la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ est une solution théorique et ne peut pas être déterminée d'une façon analytique sauf dans le cas particulier où la fonction de transition 2.1 et d'observation 2.2 sont linéaires et le bruit est gaussien. Dans ce cas là, des solutions existent en utilisant un filtre de Kalman ([Kalman, 1960]) ou un maillage de l'espace d'état ([Arulampalam *et al.*, 2002]). Dans le cas général, les filtres particulaires basés sur des méthodes de Monte Carlo sont souvent utilisés pour approximer la solution bayésienne optimale. En effet, si le modèle d'évolution $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ est non linéaire, il n'existe pas de solution optimale pour trouver la probabilité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_t)$ car il n'est pas possible d'évaluer analytiquement la probabilité $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. Dans ce cas, il est possible de calculer une approximation convergente par un filtre particulaire. Dans la suite, nous faisons un rappel sur la méthode de Monte Carlo dont nous avons besoin pour introduire le filtrage particulaire.

2.2 Rappel sur la méthode de Monte Carlo

Soit la fonction f d'une variable aléatoire de densité de probabilité $p(\mathbf{x})$. Par définition, l'espérance $E_p(f)$ de f est égale à :

$$E_p(f) = \int f(\mathbf{x}) \cdot p(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2.5}$$

Cette espérance, représentée sous forme d'une intégrale, peut être approchée par la méthode de Monte Carlo en tirant aléatoirement un grand nombre d'échantillons $\{\mathbf{x}^{(i)}\}^{i:1\cdots N}$ de f selon la loi $p(\mathbf{x})$:

$$E_p(f) \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}^{(i)}). \tag{2.6}$$

A partir de ce résultat, il est possible d'approximer $p(\mathbf{x})$ par un estimateur non biaisé en tirant N échantillons de la loi $p(\mathbf{x})$:

$$p(\mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}^{(i)}}(\mathbf{x}),$$

avec $\delta_{\mathbf{x}^{(i)}}(\mathbf{x})$ une mesure de Dirac sur $\mathbf{x}^{(i)}$.

2.2.1 Echantillonnage préférentiel

L'échantillonnage préférentiel (en anglais Importance Sampling) consiste à approximer la densité de probabilité $p(\mathbf{x})$ à partir des échantillons tirés d'une autre fonction $q(\mathbf{x})$ appelée fonction de proposition. En effet, l'équation 2.5 peut s'écrire de la façon suivante :

$$E_p(f) = \int \left[f(\mathbf{x}) \cdot \frac{p(\mathbf{x})}{q(\mathbf{x})} \right] \cdot q(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2.7}$$

En appliquant l'approximation de Monte Carlo sur l'équation 2.7, nous obtenons :

$$E_p(f) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})} \cdot f(\mathbf{x}^{(i)}),$$

avec $\{x^{(i)}\}^{i:\cdots N}$ cette fois-ci, échantillonnés de $q(\mathbf{x})$. Ce résultat est intéressant puisqu'il permet d'approximer $E_p(f)$ à partir des échantillons tirés d'une autre fonction $q(\mathbf{x})$ connue. Ce résultat peut être généralisé pour approximer une densité de probabilité $p(\mathbf{x})$ à partir des échantillons tirés d'une fonction de proposition $q(\mathbf{x})$:

$$p(\mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} w^{(i)} \delta_{\mathbf{x}^{(i)}}(\mathbf{x}),$$

avec $\mathbf{x}_t^{(i)} \sim q(\mathbf{x})$, et $w^{(i)} \propto \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}$ est le poids normalisé de l'échantillon d'indice i.

2.3 Echantillonnage préférentiel séquentiel (SIS)

L'échantillonnage préférentiel séquentiel (en anglais *Sequential Importance Sampling* SIS) est une technique pour implémenter un filtre bayésien récursif en utilisant l'approximation de Monte Carlo. Reprenons le problème de départ qui consiste à estimer la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ (voir section 2.1). Cette densité peut être exprimée sous la forme suivante (pour plus de détails voir le papier de Arulampalam *et al.* [Arulampalam *et al.*, 2002]) :

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) \mathrm{d}\mathbf{x}_{t-1}, \tag{2.8}$$

où $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ est proportionnelle (en raison de l'hypothèse markovienne de premier ordre) à :

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t \mid \mathbf{x}_t) \cdot p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \cdot p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}). \tag{2.9}$$

Dans le cas général, une solution analytique de l'intégrale de l'équation 2.8 n'est pas possible, et l'algorithme SIS permet d'obtenir une solution approximative en se basant sur l'échantillonnage de Monte-Carlo. Ainsi, l'intégrale de l'équation 2.8 peut être approximé par :

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \cdot \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t), \tag{2.10}$$

avec :

$$w_t^{(i)} \propto \frac{p(\mathbf{x}^{(i)}{}_{0:t} \mid \mathbf{y}_{1:t})}{q(\mathbf{x}^{(i)}{}_{0:t} \mid \mathbf{y}_{1:t})}. \tag{2.11}$$

q étant une fonction de proposition. Dans l'échantillonnage séquentiel par importance, q est choisie sous la forme factorisée suivante :

$$q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = q(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) \cdot q(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}). \tag{2.12}$$

et en supposant que :

$$q(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = q(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_t). \tag{2.13}$$

Ainsi, en remplaçant d'abord 2.13 dans 2.12 et ensuite 2.12 et 2.9 dans 2.11, nous obtenons :

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) \cdot p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)}. \tag{2.14}$$

Ainsi, l'algorithme d'échantillonnage séquentiel par importance approxime la densité *a posteriori* par un ensemble de N échantillons (appelées aussi particules) tirés d'une fonction de proposition q et effectue une propagation récursive des poids w_t^i des particules avec l'équation 2.14 à chaque réception d'une observation $\mathbf{y}_t$. Les différentes étapes de SIS sont présentées dans l'algorithme 1. Cet algorithme prend en entrée un ensemble de particules $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}^{i:1\cdots N}$ représentant $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, et retourne un nouvel ensemble de particules $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N}$ estimant la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ en tirant des nouvelles particules $\mathbf{x}_t^{(i)}$ de la fonction de proposition $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$, et ensuite en corrigeant les poids avec l'équation 2.14. Finalement, la densité *a posteriori* est estimée en utilisant l'équation 2.10.

Algorithm 1: Algorithme SIS

1 $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N} = \text{SIS}(\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}^{i:1\cdots N})$
2 **pour** $i : 1 \cdots N$ **faire**
3 - Prédiction : $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$
4 - Correction : $w_t^{(i)} \propto w_{t-1}^{(i)} \cdot \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \cdot p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)}$
5 - Normalisation : $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^{N} w_t^{(i)}}$
6 - Estimation de la densité *a posteriori* : $p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \cdot \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t)$

Dégénérescence

Un phénomène inévitable avec l'algorithme d'échantillonnage préférentiel séquentiel (SIS) est la dégénérescence des poids des particules au cours du temps (voir les travaux de Doucet et Johansen [Doucet and Johansen, 2011]). En effet, après plusieurs itérations, très peu de particules possèdent des poids élevés et contribuent à l'approximation de la densité *a posteriori* alors que les autres auront des poids très faibles. Une solution naïve consiste à augmenter le nombre de particules au cours du temps, mais cette solution n'est pas réalisable en pratique parce que le temps de traitement continue à croître jusqu'à atteindre les limites des ressources disponible pour le calcul. Liu et Chen [Liu and Chen, 1998] ont introduit un critère qui permet d'évaluer la dégénérescence d'un filtre particulaire égale à :

$$N_{eff} = \frac{N}{1 + Var(w_t^{*(i)})}, \tag{2.15}$$

avec $w_t^{*(i)} = \frac{p(\mathbf{x}_t^i|\mathbf{y}_{1:t})}{q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i,\mathbf{y}_t)}$ représente le "vrai poids" et $Var(w_t^{*(i)})$ est la variance de $w_t^{*(i)}$. Ce critère mesure le nombre de particules qui contribuent à l'approximation de la densité de probabilité. Si N_{eff} est inférieure à un certain seuil, l'algorithme SIS est alors dégénéré. En réalité, il est impossible d'évaluer N_{eff} , par contre une bonne approximation de N_{eff} (voir les travaux de Doucet *et al.* [Doucet *et al.*, 2000] et MacCormik et Isard [MacCormick and Isard, 2000]) est facile à obtenir, égale à :

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2}. \tag{2.16}$$

$\hat{N}_{eff}$ est appelé le diagnostic de survie (en anglais *Survival Diagnostic*). Pour comprendre comment N_{eff} mesure la dégénérescence de l'algorithme SIS, prenons deux exemples extrêmes, le premier où seule une particule possède un poids égale à 1 et les autres à zéro. Dans cette situation $\hat{N}_{eff} = 1$ indiquant qu'une seule particule contribue à l'approximation de la densité *a posteriori*. L'autre cas est lorsque toutes les particules possèdent un poids égale à $\frac{1}{N}$, ainsi $\hat{N}_{eff} = N$ indiquant que toutes les particules contribuent à l'approximation.

Ré-échantillonnage

Le ré-échantillonnage est utilisée pour réduire l'impact de la dégénérescence à chaque fois que $\hat{N}_{eff}$ devient inférieure à un seuil N_{min} prédéfini. L'idée du ré-échantillonnage est d'éliminer les particules de faibles poids et de dupliquer ceux qui ont des poids élevés. Le ré-échantillonnage consiste à générer un nouvel ensemble de particules $S'_t = \{\mathbf{x}_t^{(i)}, \frac{1}{N}\}$ de poids égal à partir d'un l'ensemble $S_t = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ de particules pondérées en éliminant les particules possédant des poids très faibles et en dupliquant ceux qui possèdent des poids élevés. La première technique de ré-échantillonnage utilisée en filtrage particulaire est l'échantillonnage multinomial (en anglais *Multinomial Resampling* [Efron and Tibshirani., 1993]) dont l'implémentation est décrite dans l'annexe A. Plusieurs variantes de cette technique existent, et les plus utilisées sont : Résiduel ([Liu and Chen, 1998]), Stratifié ([Kitagawa, 1996]) et systématique ([Kitagawa, 1996]). Dans les travaux de Douc et Cappe [Douc and Cappe, 2005], les auteurs comparent les techniques de ré-échantillonnage citées ci-dessus et montrent que les résultats de ces méthodes sont comparables et que l'échantillonnage systématique est souvent utilisé parce qu'il est facile à mettre en place. Les

travaux de Doucet *et al.* [Doucet *et al.*, 2000] listent les différentes techniques d'échantillonnage les plus utilisées.

L'échantillonnage préférentiel séquentiel que nous avons décrit forme la base des algorithmes de filtrage particulaire. Dans la section suivante Nous décrivons le fonctionnement général de ces algorithmes.

2.4 Filtre particulaire

L'algorithme de filtrage particulaire est dérivé de SIS en choisissant une fonction de proposition égale à la fonction de transition :

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}), \tag{2.17}$$

comme elle est plus aisée de calculer, et cela simplifie également le calcul des poids de l'équation 2.14 qui deviennent proportionnelles à la fonction d'observation :

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}). \tag{2.18}$$

$p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ est aussi appelée fonction de vraisemblance (en anglais *likelihood*) qui mesure le degré de similitude entre une particule et l'observation émise par le système. $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$ représente la fonction de transition qui décrit comment une particule évolue de l'instant $t-1$ à t. En filtrage particulaire l'étape de ré-échantillonnage est effectuée à chaque instant. Ainsi les poids des particules deviennent égales à :

$$w_t^{(i)} \propto p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}), \tag{2.19}$$

puisque $w_{t-1}^{(i)} = \frac{1}{N}$ suite à l'étape de ré-échantillonnage.

En résumé, les étapes d'un algorithme de filtrage particulaire dont le but est d'estimer la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ par un ensemble de N particules sont les suivantes :

— **Prédiction (propagation ou exploration)** : A l'instant t, les particules sont échantillonnées de la fonction de transition :

$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}).$$

— **Correction (ou pondération)** : Pour chaque particule, un poids $w_t^{(i)}$ est calculé par rapport à l'observation courante :

$$w_t^{(i)} \propto p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}).$$

— **Ré-échantillonnage** : Un nouvel ensemble de particules $S'_t = \{\mathbf{x}_t{}^{(i)}, \frac{1}{N}\}$ de poids égales est généré à partir de l'ensemble $S_t = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ de particules pondérées en éliminant les particules possédant des poids très faibles et en dupliquant les particules proportionnellement à leur poids.

Finalement, la densité de probabilité *a posteriori* est estimée de la façon suivante :

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \cdot \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t)$$

Un filtre particulaire générique est représenté dans l'algorithme 2. Cet algorithme est aussi appelé algorithme de Condensation qui a été introduit par Isard et Blake [Isard and Blake, 1998] pour le suivi d'un objet dans une image.

Algorithm 2: Algorithme d'un filtre particulaire (PF)

$\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N} = \text{PF}(\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}^{i:1\cdots N})$
pour $i : 1 \cdots N$ **faire**
    - Prédiction (propagation ou exploration) : $\mathbf{x}_t^i \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$.
    - Correction : $w_t^{(i)} \propto p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$.
- Calculer la somme totale des poids : $S = \sum_{i=1}^{N} w_t^{(i)}$
pour $i : 1 \cdots N$ **faire**
    - Normalisation des poids : $w_t^{(i)} = \frac{w_t^{(i)}}{S}$
- Estimation de la densité *a posteriori* : $p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \cdot \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t)$
- Ré-échantillonnage : $\{\mathbf{x}_t^{(i)}, w_t^{(i)} = \frac{1}{N}\} = Echantillonner(\{\mathbf{x}_t^{(i)}, w_t^{(i)}\})$

2.5 Application du filtrage particulaire pour le suivi du mouvement humain

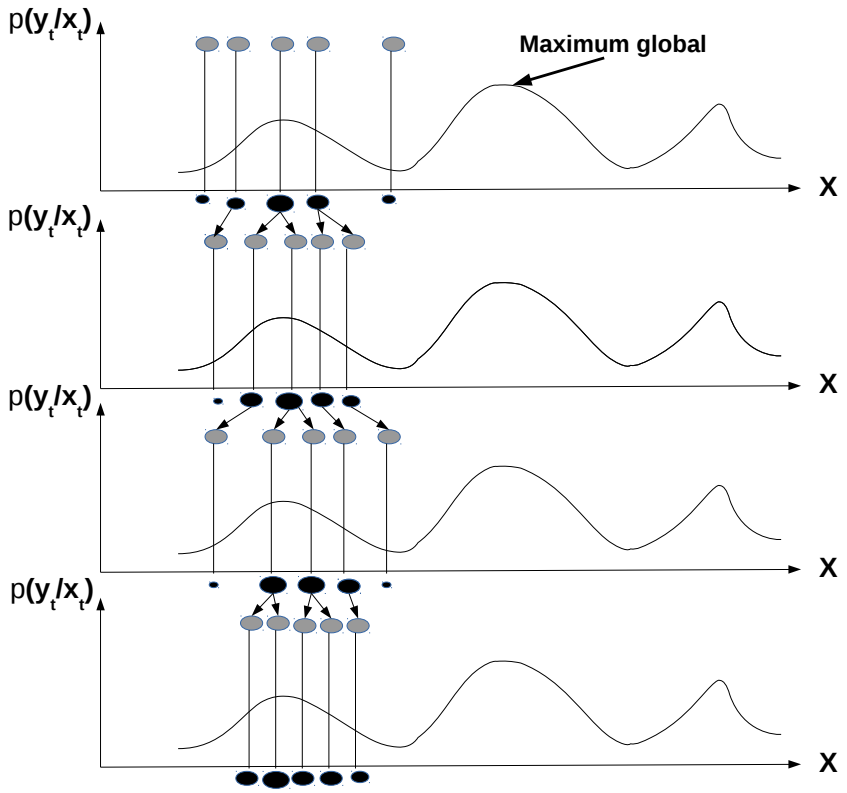
Le suivi du mouvement humain à partir d'une séquence d'images peut être formalisé comme un problème de filtrage. En effet, supposons que nous disposons d'un modèle humanoïde 3D d'un être humain. Cet humanoïde est composé d'un ensemble de segments reliés entre eux. Chaque configuration de ses segments représente un état ou une posture (par exemple, assis, debout ...). L'objectif est alors de trouver l'état $\mathbf{x_t}$ le plus probable de l'humanoïde connaissant l'observation $\mathbf{y_t}$ provenant de la caméra à l'instant t et les connaissances *a priori* sur les états précédents. Ainsi le problème se ramène à calculer la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. En général, pour le problème de suivi, il n'est pas possible de calculer cette densité d'une façon analytique à cause de la grande taille de l'espace d'état et la non linéarité du modèle de transition et d'observation. C'est pour ça que l'utilisation du filtrage particulaire pour résoudre ce problème est intéressante. En effet, le filtrage particulaire permet d'approximer la densité *a posteriori* par un ensemble de particules sans connaître sa forme ou son expression analytique exacte et peut converger vers un estimateur bayésien optimal. Un autre avantage du filtrage particulaire, est qu'il modélise bien l'incertitude. En effet, si à un instant donné une particule n'a pas une bonne vraisemblance, elle ne sera pas immédiatement rejetée, au contraire elle aura encore une chance de se prouver, ce qui permet au filtre de reprendre le suivi suite à une perte occasionnelle. En revanche, les techniques de filtrage particulaires ne se mettent pas facilement à l'échelle, et la version de base ce cet algorithme nécessite un nombre N de particules qui croit d'une façon exponentielle avec la dimension de l'espace d'état notée d ([MacCormick and Isard, 2000]) :

$$N \geq \frac{N_{min}}{\alpha^d},$$

avec α, le taux de survie, et N_{min} le diagnostic de survie minimal (voir section 2.3). Pour le suivi du mouvement humain, la dimension de l'espace d'état d est très élevée et souvent elle est supérieure à 25 degrés de liberté. Ainsi l'algorithme de filtrage particulaire nécessite un nombre très élevé de particules pour converger ce qui augmente la complexité temporelle de l'algorithme et le rend pratiquement inapplicable. Un autre problème de l'application du filtrage particulaire pour la capture de mouvement est lié à la multi-modalité de la fonction de vraisemblance $p(\mathbf{y}_t \mid \mathbf{x}_t)$ qui mesure le degré de similitude entre une particule et l'observation reçue. Cette fonction

possède souvent plusieurs maximums locaux qui peuvent attirer les particules amenant le filtre à diverger. La solution naïve à ce problème est d'utiliser un nombre très élevé de particules rendant son application inapplicable en pratique. Plusieurs techniques d'optimisations ont été introduites pour pallier ces problèmes. Parmi ces techniques, nous nous intéressons particulièrement à la méthode basée sur le recuit-simulé et une autre basée sur une factorisation de l'espace d'état. Ces deux méthodes ont été utilisées pour le suivi du mouvement humain et produisent des résultats intéressants d'après la littérature. Nous décrivons le principe de fonctionnement de ces deux techniques dans la suite.

2.5.1 Filtrage particulaire avec recuit-simulé



(a)

FIGURE 2.1 – Un exemple illustrant le phénomène d'attraction des particules par un maximum local dans un filtre particulaire classique.

La technique du recuit-simulé a pour but d'aider une faible population de particules à s'échapper des maxima locaux présents dans la fonction de vraisemblance $p(\mathbf{y}_t \mid \mathbf{x}_t)$ qui est souvent multi-modale. Ces maxima locaux attirent les particules amenant le filtre à diverger. Ce phénomène est bien illustré sur la figure 2.1 : En effet sur cette figure, en partant d'un ensemble de 5 particules de poids égaux (cercles en gris sur la courbe en haut), un poids est affecté à chaque particule suite à l'étape de correction de l'algorithme 2 (cercles en noirs). Un nouvel ensemble de particules non pondérées est généré en dupliquant les particules d'une façon proportionnelle à leurs poids (étape de ré-échantillonnage). Ensuite l'étape de prédiction est appliquée sur les

particules résultantes et ainsi de suite. Nous remarquons qu'après plusieurs itérations, toutes les particules sont attirées par le maximum local présent dans la fonction de vraisemblance (courbe en bas), et il est difficile et voir impossible à l'algorithme de sortir de cette situation à moins d'utiliser un nombre très grand de particules.

Pour aider les particules à s'échapper des maxima locaux, Deutscher *et al.* [Deutscher *et al.*, 2000] proposent une méthode basée sur le recuit-simulé originalement introduit par Kirkpatrick *et al.* [Kirkpatrick *et al.*, 1983] qui consiste à appliquer une politique de recherche multi-couches (en anglais *Layered Search*). Pour chaque couche, une passe de filtrage particulaire est appliquée mais cette fois ci en assouplissant fortement la fonction de vraisemblance dans les premières couches. En effet, soit M le nombre de couches, l'algorithme commence de la couche M jusqu'à atteindre la première couche. Pour une couche m, avec $0 \leq m \leq M$, la fonction de vraisemblance est atténuée et lissée avec un facteur β_m. Ainsi la nouvelle fonction de vraisemblance pour un niveau m est égale à :

$$p_m(\mathbf{y}_t \mid \mathbf{x}_t) = p(\mathbf{y}_t \mid \mathbf{x}_t)^{\beta_m}$$

avec $1 = \beta_0 > \beta_1 \cdots > \beta_M$. Ainsi, pour chaque observation reçue à l'instant t, les particules sont filtrées M fois avec chaque fois une fonction de vraisemblance différente en commençant par la couche M jusqu'à atteindre la première couche. Cette heuristique de recherche permet aux particules au début d'explorer mieux l'espace d'état en utilisant la fonction de vraisemblance comme guide mais sans que les particules soient mal orientées. Au fur et à mesure qu'on monte dans les couches, la fonction de vraisemblance devient de plus en plus pointue, guidant les particules à atteindre le maximum global de la fonction. La figure 2.2 illustre ce processus, en utilisant la même fonction d'observation que l'exemple de la figure 2.1, cette fonction est lissée sur 3 niveaux. Ce lissage rend les maximums moins pointus, ce qui assouplit le poids des particules, et donne ainsi la chance à ces particules de mieux explorer l'espace sans qu'elles soient rejetées.

La méthode de recuit-simulé introduit une autre modification qui intervient dans l'étape d'exploration (ou prédiction) de filtrage particulaire (voir algorithme 2). Cette modification permet aux particules dans les premières couches d'explorer un grand intervalle de l'espace et au fur et à mesure qu'on monte dans les couches, l'intervalle d'exploration devient de plus en plus réduit permettant aux particules de se focaliser sur les caractéristiques locales de la fonction d'observation. Ainsi la dynamique du système [1] devient :

$$x_t{}^{(i),(m-1)} = x_t{}^{(i),(m)} + \mathcal{N}(0, P^m), \tag{2.20}$$

avec $\mathcal{N}$ est une loi normale multidimensionnelle avec P^m la matrice de covariance pour une couche m obtenue de la façon suivante :

$$P^m = P^0 \cdot 0.5^{(M-m)},$$

où P^0 est la matrice de covariance initiale dont les éléments diagonaux correspondent à la vitesse maximale des paramètres du modèle sur un pas de temps.

Finalement, à chaque itération, la méthode du recuit-simulé :

— Corrige de plus en plus avec la fonction de vraisemblance.

— Explore de moins en moins en utilisant la fonction de transition.

1. La dynamique du système est représentée par la fonction de transition et elle est utilisée pour faire évoluer les particules dans l'étape d'exploration de l'algorithme 2. Dans le cas où la fonction de transition du modèle est inconnue (comme dans le problème de suivi du mouvement humain), il est commun d'utiliser un déplacement aléatoire, ce qui se traduit par l'équation 2.20.

Un filtre particulaire avec recuit-simulé est représenté par l'algorithme 3. Dans cet algorithme, l'ensemble des particules pour une couche donnée est représenté par :

$$S_t^{(m)} = \{x_t^{(i),(m)}, w_t^{(i),(m)}\}^{i:1\cdots N}$$

.

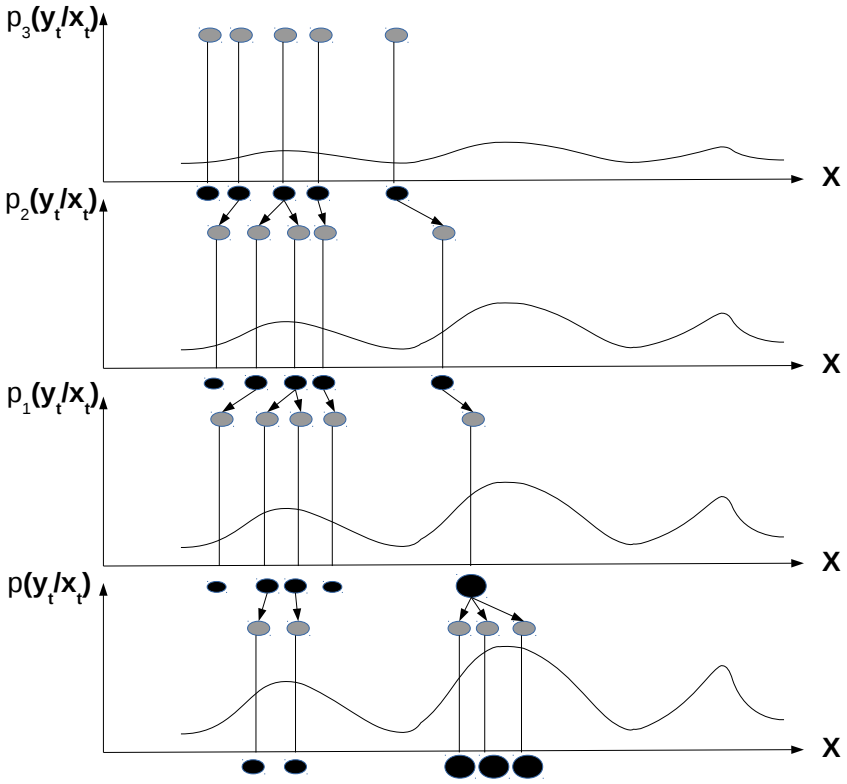


FIGURE 2.2 – La fonction de vraisemblance originale (la même que la figure 2.1)lissée sur trois niveaux. Les maxima sont fortement atténués sur les couches plus profondes ce qui permet aux particules de mieux explorer l'espace sans qu'elles soient attirées par des maximums locaux.

2.5.2 Filtrage particulaire factorisé

Originalement proposée par MacCormick et Isard [MacCormick and Isard, 2000], aussi appelée échantillonnage partitionnée (en anglais *Partitioned Sampling* PS), l'objectif de cette méthode est de réduire le nombre de particules nécessaires pour le suivi d'un objet articulé dans un espace d'état de grande dimension. L'idée derrière cette méthode est simple et consiste à factoriser l'espace d'état en sous-espaces et d'appliquer un filtre particulaire sur chaque sous-espace. Pour le corps humain, cette factorisation est naturelle. En effet, la structure cinématique du corps humain est composée d'un ensemble de segments reliés entre eux par des contraintes bio-mécanique. Cette structure peut être représentée sous forme factorisée en considérant le torse comme la racine de cet arbre. Cette factorisation d'espace d'état permet de réduire considérablement le nombre de particules requis. Par exemple, l'évaluation des hypothèses sur la position du torse permet de réduire l'espace de recherche sur les cuisses et ainsi de suite. Nous décrivons dans la suite le fonctionnement de cette méthode.

Algorithm 3: Algorithme d'un filtre particulaire avec recuit-simulé

entrée: $\{x_{t-1}^{(i),(0)}, w_{t-1}^{(i),(0)}\}^{i:1\cdots N}$, $\mathbf{y}_t$
sortie : $\{x_t^{(i),(0)}, w_t^{(i),(0)}\}^{i:1\cdots N}$

pour $i : 1 \cdots N$ **faire**
  - Exploration : ${x_t}^{(i),(M)} = x_t^{(i),(0)} + \mathcal{N}(0, P^0)$
  - Correction : $w_t^{(i),(M)} \propto p(\mathbf{y}_t/\mathbf{x}_t^{(i)})^{\beta_M}$.

pour $m : M \cdots 1$ **faire**
  - Calculer $P^m = P^0 \cdot 0.5^{(M-m)}$
  - Calculer la somme totale des poids : $S = \sum_{i=1}^{N} w_t^{(i),(m)}$
  pour $i : 1 \cdots N$ **faire**
    - Normalisation des poids : $w_t^{(i),(m)} = \frac{w_t^{(i),(m)}}{S}$
  - Ré-échantillonnage : $\{{x_t}^{(i),(m)}, w_t^{(i),(m)} = \frac{1}{N}\} = Echantillonner(\{{x_t}^{(i),(m)}, w_t^{(i),(m)}\})$
  pour $i : 1 \cdots N$ **faire**
    - Exploration : ${x_t}^{(i),(m-1)} = {x_t}^{(i),(m)} + \mathcal{N}(0, P^m)$
    - Correction : $w_t^{(i),(m-1)} \propto p(\mathbf{y}_t/\mathbf{x}_t^{(i)})^{\beta_{m-1}}$.

Soit X l'espace d'état et Y l'espace des observations. Cette méthode suppose que X et Y peuvent être factorisés telles que $X = X^1 \times \cdots \times X^K$ et $Y = Y^1 \times \cdots \times Y^K$ respectivement (K étant le nombre des sous-espaces, par exemple, pour le suivi du mouvement humain, K est le nombre des différentes parties du corps). La méthode factorisée suppose aussi que la dynamique du système de l'équation 2.1 peut être décomposée de la façon suivante :

$$f_t(\mathbf{x}_{t-1}, \pi_t) = f_t^K \circ f_t^{K-1} \circ \cdots f_t^j \circ \cdots \circ f_t^1(\mathbf{x}_{t-1}), \tag{2.21}$$

avec f_t^j modifie uniquement le sous-espace X^j. Ainsi, l'étape de propagation dans l'algorithme de filtrage particulaire classique (voir algorithme 2) est remplacée par une séquence de propagations f_t^j des sous-espaces. De la même façon, la fonction d'observation de l'équation 2.2 peut être décomposée de la façon suivante :

$$p(\mathbf{y}_t \mid \mathbf{x}_t) = h_t(\mathbf{x}_t, \nu_t) = \prod_{j=1}^{K} p_t^j(\mathbf{y}_t^j \mid \mathbf{x}_t^j), \tag{2.22}$$

avec $\mathbf{x}_t^j$ et $\mathbf{y}_t^j$ sont les projections de $\mathbf{x}_t$ et $\mathbf{y}_t$ dans le sous espace X^j et Y^j respectivement. Ainsi, l'étape de correction dans l'algorithme de filtrage particulaire classique (voir algorithme 2) est remplacée par une séquence de corrections p_t^j. Les hypothèses présentées ci-dessus amènent au diagramme de condensation illustré sur la figure 2.3, où la méthode factorisée (PS) peut être vue comme une cascade de K filtres particulaire classiques. En effet, la méthode PS prend en entrée un ensemble de particules estimant $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{t-1})$, elle propage les particules en utilisant la fonction f_t^1 appliquée sur X^1. Ensuite, elle applique l'étape de correction en utilisant p_t^1 suivi d'une étape de ré-échantillonnage. Ce processus est répété sur les K sous-espaces pour obtenir finalement la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. MacCormick et Isard [MacCormick and Isard, 2000] montrent que ce diagramme est mathématiquement correct.

Un filtre particulaire factorisé est représenté par l'algorithme 4 (dans cet algorithme, chaque particule $x_t^{(i)}$ est représentée par un vecteur de tuples $(x_t^{(i),(j)})^{j:1\cdots K}$.

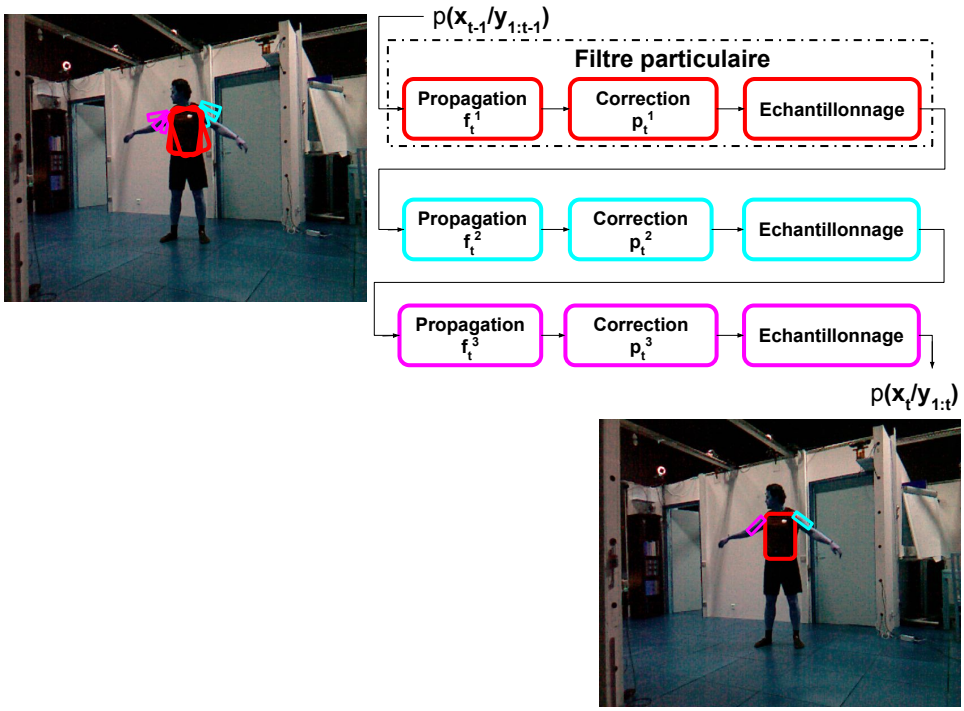


FIGURE 2.3 – Diagramme de condensation du filtre particulaire factorisé (PS).

Une remarque sur la méthode de filtre particulaire factorisé est que la factorisation de l'espace d'état est souvent appelée stratégie d'exploration car elle décrit comment les particules explorent l'espace. Dans la littérature, plusieurs stratégie ont montré leur efficacité. A titre d'exemple, sur la figure 2.4 est affichée deux stratégies d'exploration différentes.

Algorithm 4: Algorithme d'un filtre particulaire factorisé

entrée: $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}^{i:1\cdots N}$

sortie : $\{x_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N}$

pour $j : 1 \cdots K$ **faire**

  pour $i : 1 \cdots N$ **faire**

    - Propagation $x_t^{(i),(j)} \sim f_t^j$.

    - Correction $w_t^{(i),(j)} \propto p_t^j$

  - Ré-échantillonnage : $\{x_t^{(i),(j)}, w_t^{(i),(j)} = \frac{1}{N}\} = Echantillonner(\{x_t{}^{(i),(j)}, w_t^{(i),(j)}\})$

2.5.3 Approche combinée

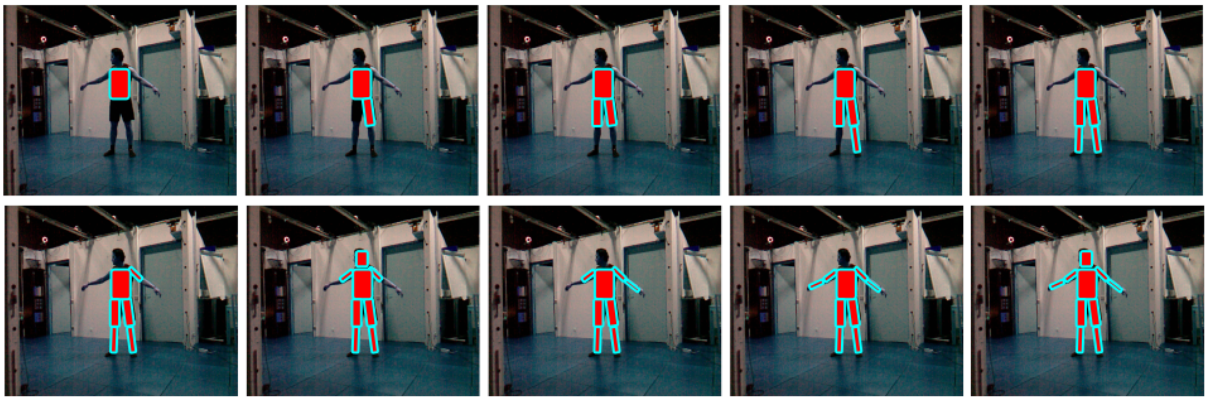
Bandouch *et al.* [Bandouch *et al.*, 2008] et Bandouch et Beetz [Bandouch and Beetz, 2009] ont proposé une méthode qui combine les deux méthodes citées ci-dessus pour le suivi du mouvement humain. Cette combinaison consiste à appliquer une recherche multi-couches sur les différents sous-espaces. Les auteurs proposent aussi une nouvelle stratégie d'exploration qui consiste à représenter le torse, les cuisses et la tête dans un même sous-espace (voir figure 2.4(b)). Les auteurs ont démontré que cette nouvelle méthode donne des meilleurs résultats que celle du

recuit-simulé et de la méthode factorisée et nécessite moins de particules. Cependant, les auteurs traitent les occlusions en utilisant un système à 4 caméras et en étiquetant manuellement les zones de l'images susceptibles de générer des occlusions avant de lancer leur système de suivi. Ces zones sont ensuite retirées de la fonction d'observation. Cette façon de gérer les occlusions suppose que la scène est fixe et que les endroits pouvant générer des occlusions ne vont pas changés lors du suivi.

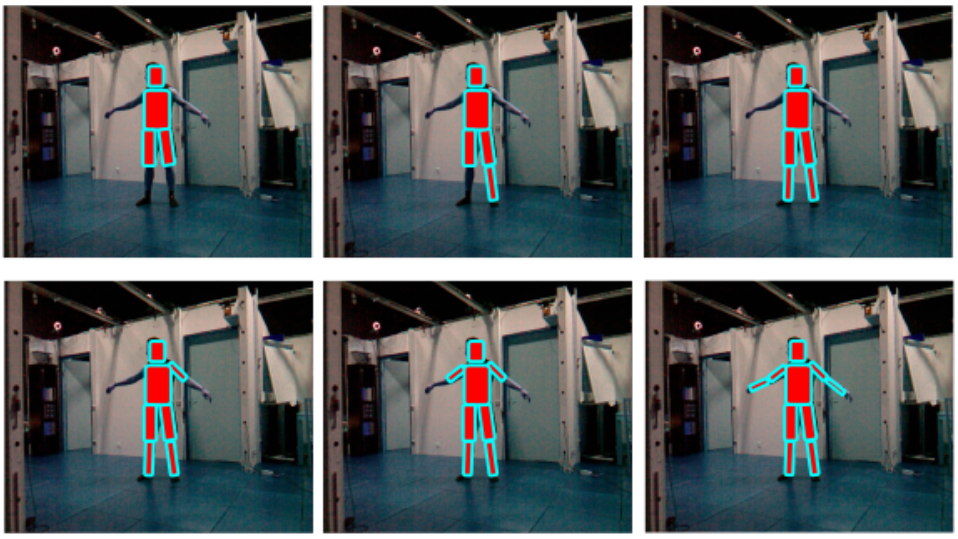
2.6 Conclusion

Dans ce chapitre, nous avons présenté d'abord le problème du filtrage en général qui consiste à estimer l'état du système à chaque instant en calculant la densité *a posteriori* de l'équation 2.4. Nous avons vu qu'il est impossible de calculer d'une façon analytique cette densité sauf dans des cas où les fonctions de transition et d'observation du système sont linéaires (en utilisant le filtre de Kalman par exemple). Dans le cas général, le filtrage particulaire est utilisé pour approximer cette densité. Nous avons ensuite présenté le cadre théorique du filtrage particulaire basé sur l'échantillonnage de Monte Carlo, et nous avons décrit un premier algorithme générique d'un filtre particulaire qui utilise un ensemble d'échantillons (aussi appelés particules) pour approximer la densité *a posteriori*. Ensuite nous avons montré que l'application de cet algorithme pour le suivi du mouvement humain nécessite un nombre très élevé de particules pour converger. Nous avons enfin présenté les deux techniques les plus utilisées qui ont été développées dans la littérature pour améliorer les performances de l'algorithme original. La première technique utilise le principe du recuit-simulé pour aider les particules à s'échapper des maximums locaux présents dans la fonction de vraisemblance, et la deuxième représente l'espace d'état sous forme factorisée pour réduire le nombre de particules requis pour le suivi.

Ces deux techniques vont être utilisées pour développer notre méthode de suivi du mouvement humain (qui sera décrite dans le chapitre 4). Avant de décrire la méthode de suivi du mouvement, nous introduisons dans le chapitre suivant, la méthode que nous avons développée pour extraire la silhouette de la personne dans un environnement dynamique. L'obtention de la silhouette est une étape importante pour l'extraction de la posture. Elle va servir à construire la fonction de vraisemblance pour évaluer le degré de similitude entre chaque particule et l'observation.



(a)



(b)

FIGURE 2.4 – Exemples de deux stratégies d'exploration différentes utilisées dans un filtrage particulaire factorisé. a) Chaque partie du corps est considérée comme un sous-espace. b) Le torse, la tête et les cuisse forment un même sous-espace. Cette dernière stratégie a été proposée par Bandouch *et al.* [Bandouch *et al.*, 2008].

Chapitre 3

Extraction de la silhouette dans un environnement dynamique

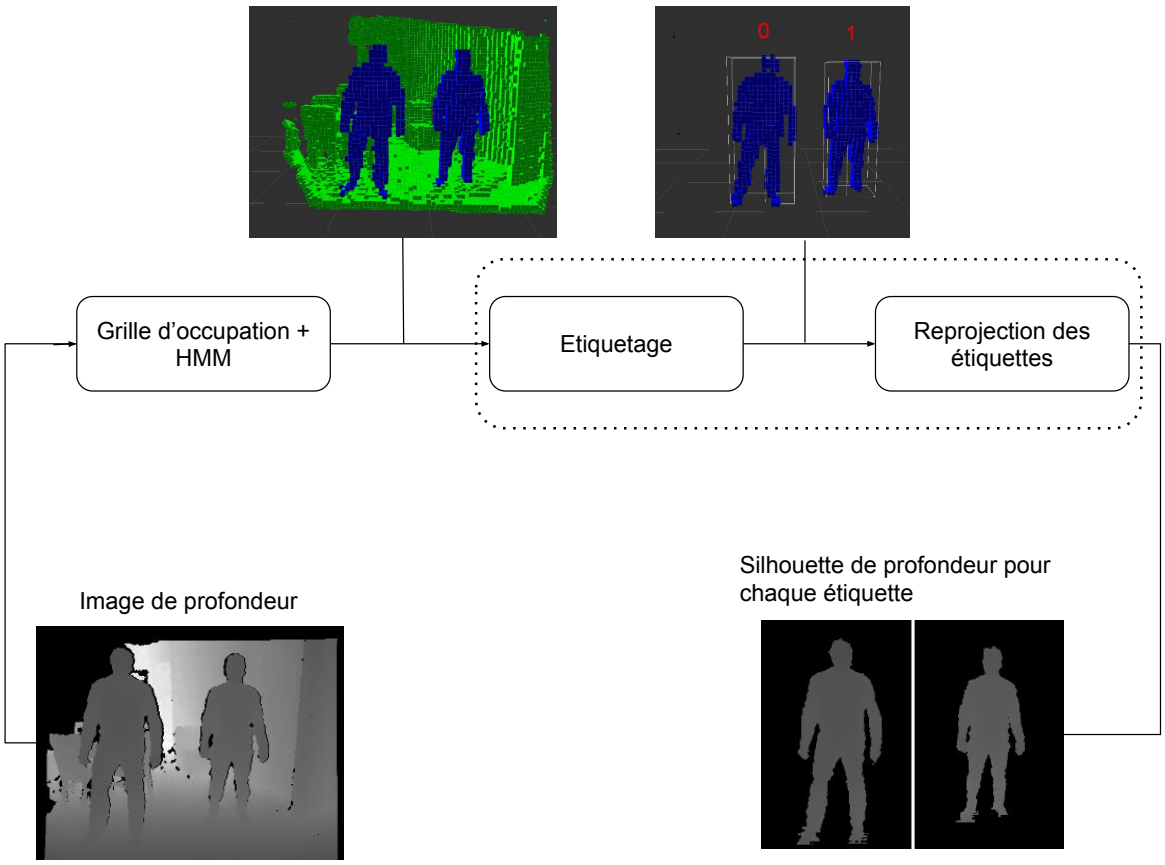
Sommaire

3.1 Méthodes existantes 38
3.2 Rappel sur le Modèle de Markov Caché 42
3.2.1 Modèle de Markov 42
3.2.2 Modèle de Markov caché MMC ou HMM 42
3.3 Modèle HMM pour identifier les objets fixes et mobiles dans la scène 44
3.3.1 Evidence virtuelle 47
3.3.2 Inférence 47
3.3.3 Modélisation du fond dynamique 49
3.4 Extraction de la silhouette 50
3.4.1 Étiquetage 50
3.4.2 Reconstruction de la silhouette à partir d'une étiquette 52
3.5 Conclusion 53

Dans ce chapitre, nous présentons la méthode que nous avons développée pour extraire la silhouette de la personne à partir d'une caméra RGB-D dans un environnement dynamique. Cette silhouette sera utilisée par l'algorithme de suivi du mouvement qui sera détaillé dans le chapitre suivant. Etant donné que la caméra est mobile et que l'environnement est dynamique, les méthodes classiques de soustraction d'arrière plan ne sont pas adaptées. Nous proposons alors dans ce chapitre une nouvelle méthode d'extraction de la silhouette capable d'apprendre en continu l'arrière plan de la scène et d'extraire la silhouette de la personne. La méthode développée est composée de deux parties (voir figure 3.1) :

— La première partie segmente l'espace 3D en un ensemble fini de cellules de taille fixe formant une grille d'occupation ([Elfes, 1989a]). Ensuite, un Modèle de Markov Caché (HMM pour *Hidden Markov Model* [Rabiner, 1989]) est utilisé pour déterminer si chaque cellule de la grille est occupée par un objet fixe (cellules en vert sur la figure 3.1) ou mobile (cellules en bleu sur la figure 3.1).

— La deuxième partie utilise un algorithme d'étiquetage (en anglais *Component Labeling*) pour regrouper en étiquettes les cellules occupées par des objets mobiles. Cette partie assure le suivi du déplacement de ces étiquettes dans le temps. Chaque étiquette est ensuite projetée sur l'image de profondeur de la caméra pour extraire sa silhouette de



(a)

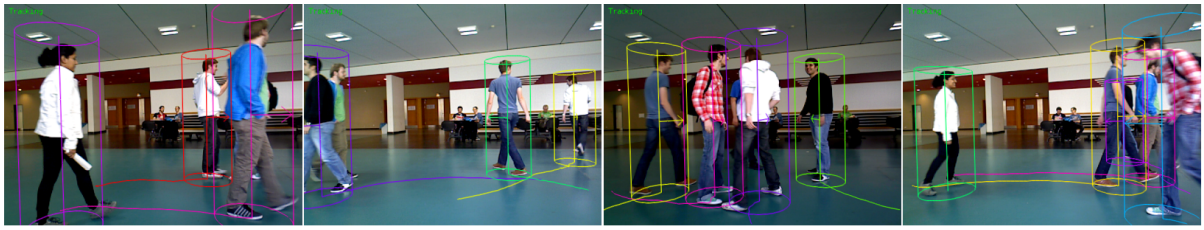
FIGURE 3.1 – Les différentes étapes de traitement de la méthode développée pour extraire la silhouette de la personne dans un environnement dynamique. Le HMM utilise l'image de profondeur reçue en entrée pour identifier si chaque cellule de la grille d'occupation est occupée par un objet du décor (en vert) ou un objet mobile de la scène (en bleu). Ensuite, un algorithme d'étiquetage est appliqué pour regrouper les cellules occupées par des objets mobiles en étiquettes. Finalement, chaque étiquette est projetée sur l'image de profondeur pour extraire sa silhouette.

profondeur qui sera utilisée par l'algorithme de suivi du mouvement de la personne décrit dans le chapitre 4.

Dans l'objectif de valider le choix de notre approche, nous présentons dans la section suivante les principales méthodes existantes que nous avons identifiées dans la littérature pour l'extraction de la silhouette à partir d'une caméra mobile de type RGB-D. Ensuite, un rappel sur les HMM sera fait dans la section 3.2. Les sections 3.3 et 3.4 décrivent les deux parties de la méthode que nous avons développée (Grille d'occupation + HMM et l'algorithme d'étiquetage).

3.1 Méthodes existantes

Les systèmes pour la détection des personnes à partir d'une caméra mobile de type RGB-D sont intéressants à étudier car ils proposent des alternatives aux méthodes de soustraction de fond classiques qui ne sont pas adaptées pour une caméra mobile dans un environnement dynamique.



(a)

FIGURE 3.2 – La sortie sous forme de boîtes englobantes d'un système de détection des personnes à partir d'une caméra mobile.

Ces systèmes permettent de détecter et suivre les personnes dans l'image, mais ne s'intéressent pas à l'extraction de la posture. Nous étudions particulièrement ces méthodes pour comprendre comment elles procèdent à l'extraction de la silhouette de la personne. La figure 3.2 montre le résultat typique de ces systèmes sous forme des boîtes englobantes représentant les personnes détectées dans la scène.

Comme illustration de ces méthodes, le système proposé par Zhang *et al.* [Zhang *et al.*, 2013] consiste à détecter les personnes à partir d'une caméra de type RGB-D montée sur un robot se déplaçant dans un environnement peuplé. Le fonctionnement de ce système est le suivant : Tout d'abord, les points 3D appartenant au sol et au plafond sont extraits avec une méthode classique de régression par moindres carrés couplée à l'algorithme de RANSAC ([Fischler and Bolles, 1981]) pour éliminer les points aberrants (voir figure 3.3(b)). Ainsi ces points sont éliminés du nuage de points. Ensuite la distribution de profondeur est calculée sur le reste des points 3D du nuage pour identifier les régions candidates pouvant contenir une personne (voir figure 3.3(c)). L'idée repose sur le fait que les personnes et les objets dans la scène possèdent un ensemble de points 3D avec des profondeurs proches. Les auteurs appellent ces régions candidates : *Depth Of Interest* (DOI), similaire à *Region Of Interest* (ROI) en traitement d'image. Pour extraire les DOI, un noyau (en anglais *Kernel*) gaussien (voir les travaux de Parzen [Parzen, 1962]) est appliqué sur la fonction de distribution de profondeur. Le nuage de points est ensuite filtré pour ne garder que les points 3D qui appartiennent aux DOI. Une nouvelle image de profondeur est ensuite régénérée à partir des ces points. Un algorithme d'étiquetage est ensuite appliqué sur l'image résultante et seuls les points 3D dont la profondeur des pixels correspondants appartiennent à la même étiquette sont conservés (voir figure 3.3(d)). Finalement, pour obtenir la silhouette des personnes, une série de détecteurs en cascade est appliquée sur les étiquettes pour éliminer celles qui ne correspondent pas à des personnes. Les trois premiers détecteurs rejettent les étiquettes dont la hauteur et la taille sont inférieures à un certain seuil. Un autre détecteur basé sur le calcul des vecteurs normaux des points 3D de chaque étiquette est utilisé pour filtrer les plans et les murs. Le dernier détecteur est le fameux détecteur de Dalal et Triggs [Dalal and Triggs, 2005] utilisé pour ne garder que les étiquettes correspondant aux personnes.

Une autre approche proposée par Spinello *et al.* [Spinello *et al.*, 2010] consiste à segmenter le nuage de points 3D en un ensemble de segments ou couches suivant leur hauteur. Ensuite, pour chaque segment, un ensemble de descripteurs (en anglais *descriptors*) est calculé. Chaque descripteur est défini par une fonction $f : S^M \rightarrow \mathbb{R}$, qui prend les M points du segment en entrée et retourne un scalaire. Les auteurs utilisent 17 descripteurs. Parmi ces descripteurs, il y a ceux qui calculent la surface, le ratio de la boîte englobante, et la surface de l'enveloppe convexe etc... (pour voir la liste complète, se référer à l'article [Arras *et al.*, 2007]). Finalement, pour détecter les personnes, pour chaque segment, un classificateur de type AdaBoost ([Freund and Schapire,

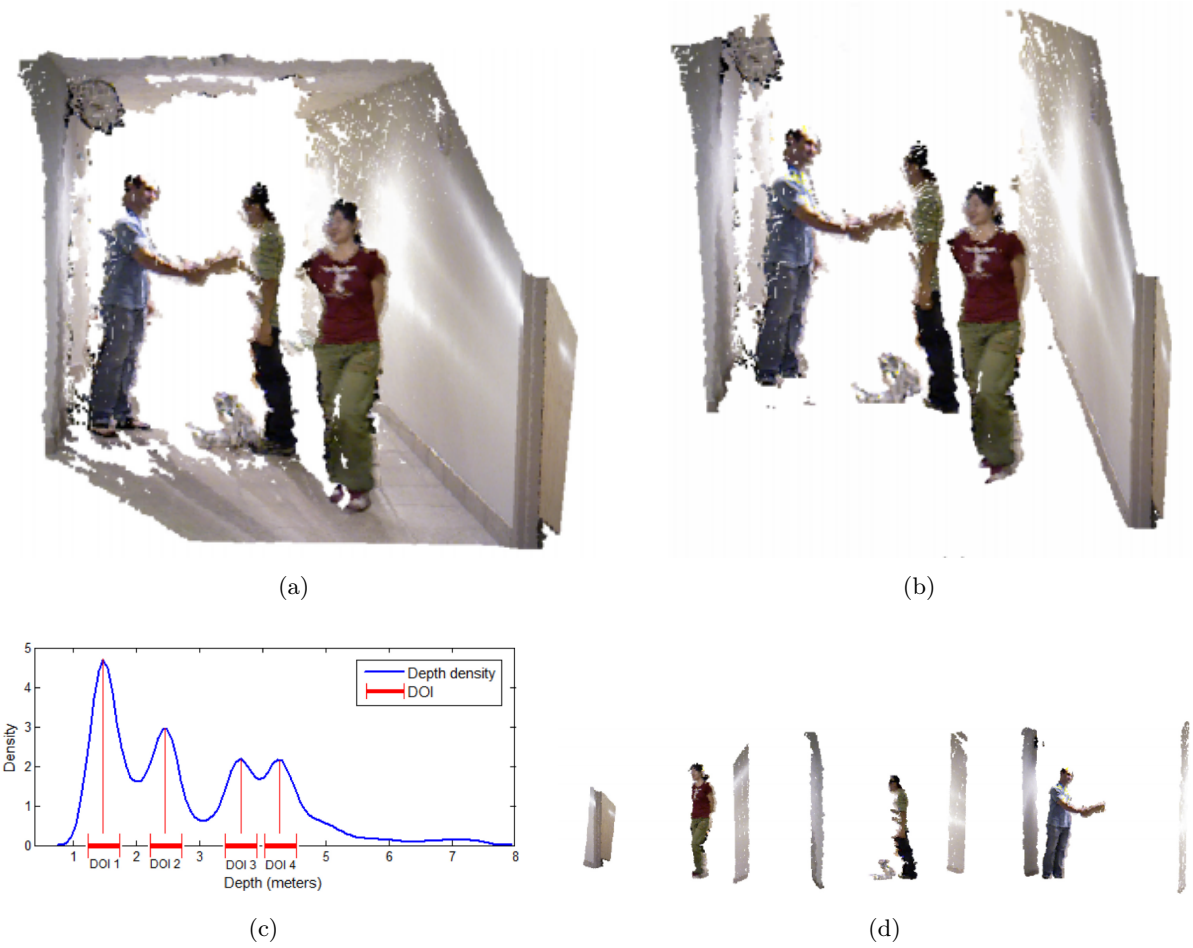


FIGURE 3.3 – La méthode d'extraction de la silhouette utilisée par Zhang *et al.* [Zhang *et al.*, 2013] pour la détection des personnes. a) Le nuage de point d'entrée. b) Elimination des points 3D appartenant au sol et plafond. c) Distribution de profondeur avec identification des zones candidates (*Depth of Interest*) DOI. d) Les étiquettes candidates.

1997]) est appris à partir des descripteurs déjà extraits.

Certaines approches (voir les travaux de Darel *et al.* [Darrell *et al.*, 1998] et Grest et Koch [Grest and Koch, 2004]) utilisent des méthodes d'extraction des objets de l'image (en anglais *Blob Detection*). Ces objets ont des caractéristiques communes (profondeur, couleur etc...). Pour détecter les *blobs* qui correspondent à des personnes, l'algorithme de détection de visage proposé par Viola et Jones [Viola and Jones, 2001] est utilisé.

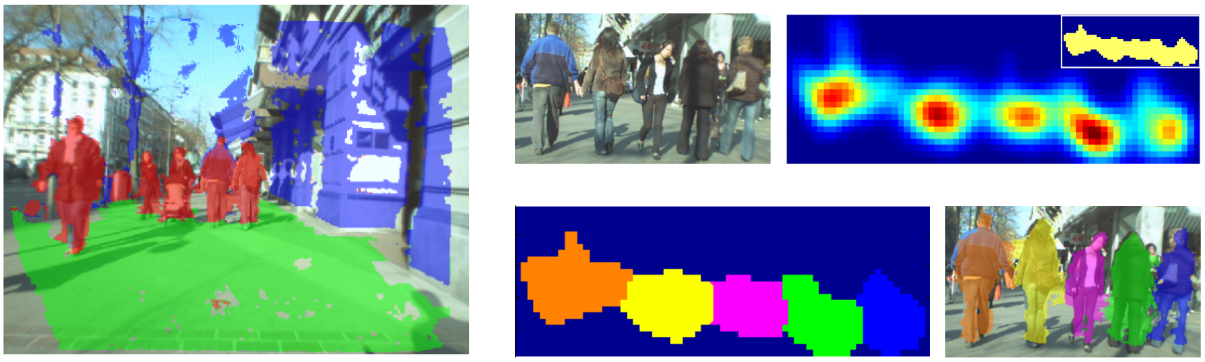
D'autres types d'approches utilisent une grille d'occupation 2D et une carte de hauteur pour la détection des personnes. Comme exemple, le système proposé par Jafari *et al.* [Jafari *et al.*, 2014] peut être cité. Ce système consiste à détecter les personnes à partir d'une caméra de type RGB-D montée sur un robot se déplaçant dans un environnement peuplé. Une méthode d'odométrie visuelle est d'abord utilisée pour aligner les images dans un repère commun. Ensuite, le nuage de points obtenu par fusion est segmenté en trois classes (voir figure 3.4(a)) : Sol (S), Objets (O) et décor (D). Cette classification est faite de la façon suivante : d'abord une méthode d'extraction de sol (plan horizontal) en utilisant l'algorithme de RANSAC est appliquée sur les points 3D dont la hauteur est inférieure à 2 mètres. Le nuage de points est ensuite projeté sur le sol qui forme une grille 2D d'occupation. Un histogramme de hauteur est ensuite calculé pour chaque cellule de la grille. Finalement pour extraire la silhouette, un algorithme d'étiquetage de type *Quick Shift* ([Vedaldi and Soatto, 2008]) est appliqué sur la projection des points 3D appartenant à la classe Objets (O) sur une grille d'occupation 2D. Les étiquettes 2D obtenues correspondant aux silhouettes des personnes dans la scène sont ensuite projetées sur l'image 2D pour obtenir les silhouettes finales (voir figure 3.4(b)). D'autres approches utilisant entre autres des grilles d'occupation pour modéliser le fond de la scène existent. Nous pouvons ainsi citer les travaux de Harville [Harville, 2002] et les travaux de Hayashi *et al.* [Hayashi *et al.*, 2004].

En conclusion, la majorité des méthodes de détection des personnes que nous avons étudiées sont orientées pour les applications de navigation autonome des véhicules, détection des piétons, surveillance et interaction homme machine. Dans ce type d'applications, les personnes sont supposées être toujours debout et c'est pour cela que la majorité de ces approches reposent sur une segmentation du nuage de points suivant la hauteur, pour simplifier la tâche d'extraction de la silhouette. Notre problématique est différente dans le sens où nous cherchons à suivre une personne chez elle qui peut être dans d'autres situations que debout, comme par exemple, assise, allongée dans un canapé, ou par terre suite à une chute. C'est principalement pour ça que nous n'utilisons pas une approche par segmentation de haut en bas de l'image. En revanche, l'utilisation de la grille d'occupation [2] pour apprendre le fond de la scène comme nous avons vu dans certains travaux cités ci-dessus, nous intéresse pour plusieurs raisons :

- Cette méthode nous permet d'avoir une représentation unique de l'environnement, et permet aussi la fusion de plusieurs types de capteurs hétérogènes, fixes ou mobiles.
- Cette méthode intègre un modèle d'incertitude sur les données bruitées du capteur, ainsi nous pouvons modéliser différents types de capteurs avec les grilles d'occupation.
- Dans le domaine de la robotique, les grilles d'occupation sont intensivement utilisées pour la cartographie de l'environnement. Ainsi, l'utilisation de la grille d'occupation pour construire notre méthode d'extraction de la silhouette donne une cohérence à notre approche et lui permet aussi de s'intégrer facilement avec les méthodes de localisation et de cartographie basées sur l'utilisation des grilles d'occupation.

Dans la suite, nous faisons un rappel sur le modèle de Markov caché dont nous avons besoin pour

2. Par définition, la grille d'occupation telle qu'elle a été introduite par Elfes [Elfes, 1989a] pour la localisation et la navigation des robots consiste à segmenter ou discrétiser l'espace 3D ou 2D en un ensemble fini de cellules de taille fixe.



(a) (b)

FIGURE 3.4 – La méthode d'extraction de la silhouette utilisée par Jafari *et al.* [Jafari *et al.*, 2014] pour la détection des personnes. a) La classification du nuage de points en trois classes : Sol (S), Objets (O) et décor (D). b) D'abord les points 3D appartenant à la classe Objets sont projetés sur une grille d'occupation 2D. Ensuite un algorithme d'étiquetage est appliqué pour extraire les différentes étiquettes dans l'image. Finalement, une projection sur l'image est faite pour chaque étiquette pour obtenir la silhouette de chaque personne.

la conception de notre méthode d'extraction de la silhouette.

3.2 Rappel sur le Modèle de Markov Caché

3.2.1 Modèle de Markov

Le modèle de Markov est un processus stochastique décrivant l'évolution de l'état d'un système à chaque instant t. Le modèle de Markov suit l'hypothèse Markovienne selon laquelle l'état à l'instant t ne dépend que de l'état à l'instant $t-1$. Cette hypothèse se traduit par :

$$P(Q_t \mid Q_1, Q_2, \cdots, Q_{t-1}) = P(Q_t \mid Q_{t-1}),$$

Q_t représentant l'état à l'instant t. Un modèle de Markov est composé d'un ensemble de N états discrets notés $(S_1, S_2, \cdots, S_N)$ et de transitions entre ces différents états. Cette transition représente la probabilité de passer d'un état S_i à un autre S_j et elle est notée $a_{ij} = P(Q_t = S_j \mid Q_{t-1} = S_i)$. Un exemple d'un modèle de Markov à deux états est représenté sur la figure 3.5. Le modèle de Markov tel qu'il est décrit ci-dessus, permet d'étudier l'évolution de la séquence d'états produite par le système à chaque instant.

3.2.2 Modèle de Markov caché MMC ou HMM

Dans un modèle de Markov caché (en anglais HMM *Hidden Markov Model*), les états du système ne sont pas observés directement. Seules les observations émises par les états sont observées ([Rabiner, 1989]). C'est pourquoi que nous introduisons la variable discrète Y_t pour représenter l'observation émise par le système à un instant t. Y_t peut prendre une des valeurs définies dans l'ensemble constitué de M symboles : $\{O_1, O_2, \cdots, O_M\}$. La figure 3.6 correspond à la représentation graphique d'un HMM, déroulée dans le temps, avec Q_t l'état caché du système et Y_t la variable observée.

Plus formellement, un HMM est défini par :

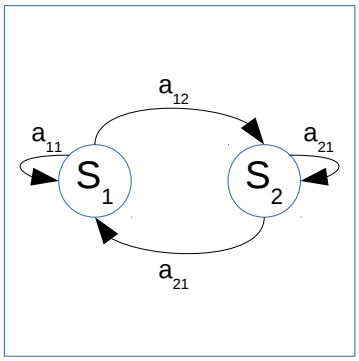


FIGURE 3.5 – Un modèle de Markov à deux états.

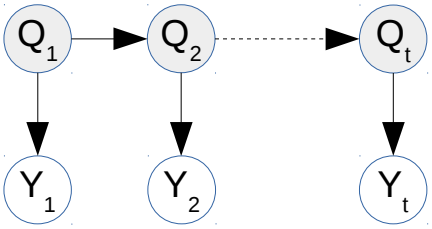


FIGURE 3.6 – Représentation graphique d'un HMM.

— Un ensemble de N états discrets $\{S_1, S_2, \cdots S_N\}$.
— La probabilité de transition d'un état S_i à un autre S_j défini par : $a_{ij} = P(Q_t = S_j \mid Q_{t-1} = S_i)$ avec la somme des probabilités de transition partant d'un état est égale à 1 :

$$\sum_{j=1}^{N} a_{ij} = 1.$$

— La probabilité initiale $\pi_i = P(Q_1 = S_i)$ pour chaque état S_i qui est la probabilité que le système se trouve dans l'état S_i au départ avec :

$$\sum_{i=1}^{N} \pi_i = 1.$$

— La probabilité d'émission d'un symbole O_j pour chaque état S_i défini par : $P(Y_t = O_j \mid Q_t = S_i)$. Les probabilités d'émission des symboles pour chaque état constituent une loi de probabilité et nous avons par conséquent :

$$\sum_{j=1}^{M} P(Y_t = O_j \mid Q_t = S_i) = 1.\ \forall i \in N.$$

L'inférence dans un HMM consiste à trouver la distribution de probabilités sur les variables cachées à un instant donné connaissant la séquences des observations émises. Les algorithmes *Forward-Backward* et *Viterbi* sont souvent utilisés pour résoudre le problème d'inférence ([Rabiner, 1989]).

3.3 Modèle HMM pour identifier les objets fixes et mobiles dans la scène

Comme nous l'avons vu précédemment, le schéma de la figure 3.1 montre les différentes étapes de la méthode que nous avons développée pour extraire la silhouette d'une personne dans une scène dynamique. La première étape utilise une grille d'occupation 3D qui consiste à segmenter ou discrétiser l'espace 3D en un ensemble fini de cellules de taille fixe. Dans notre approche, chaque cellule peut être dans 3 états (occupée par un objet fixe, mobile ou non occupée) et non pas deux (occupée ou non) comme dans les grilles d'occupations 2D classiques ([Elfes, 1989a], [Meyer-Delius *et al.*, 2012]). Pour cela, nous utilisons un HMM à 3 états pour identifier si chaque cellule de la grille d'occupation est occupée par un objet fixe ou mobile de la scène ou si elle est non occupée.

Pour mettre en place le HMM, nous avons besoin de définir l'espace d'états, les probabilités de transition entre les différents états, le modèle d'observation et la probabilité initiale pour chaque état.

Pour l'espace d'état, soit Q_t^i une variable discrète qui représente l'état à un instant t d'une cellule c_i de la grille $\mathbf{G}$. Une cellule peut être dans un des trois états suivants :

— Occupée ($\mathbf{F}$) : cellule occupée par un objet fixe de la scène.
— Mobile ($\mathbf{M}$) : cellule occupée par un objet mobile de la scène.
— Non occupée ($\mathbf{L}$) : cellule non occupée.

Ainsi l'espace d'états est défini par l'ensemble $S = \{\mathbf{F}, \mathbf{M}, \mathbf{L}\}$.

Pour les probabilités de transition entre les différents états, la figure 3.7 montre le HMM avec les probabilités de transitions α, β et γ entre les états avec :

— α la probabilité de passer de l'état $\mathbf{F}$ à $\mathbf{L}$.
— β la probabilité de passer de $\mathbf{L}$ à $\mathbf{M}$.
— γ la probabilité de passer de $\mathbf{M}$ à $\mathbf{L}$.

Nous remarquons qu'avec le HMM que nous avons défini (figure 3.7), le passage de l'état $\mathbf{L}$ à $\mathbf{F}$ et de $\mathbf{M}$ à $\mathbf{F}$ n'est pas possible. Cela signifie que si l'état d'une cellule est libre ou mobile à un instant donné, il ne peut jamais passer à fixe. Nous avons défini les valeurs de ces probabilités de façon empirique en prenant :

$$\alpha = 0.01, \beta = 0.1, \gamma = 0.4.$$

Pour le modèle d'observation, soit Y_t^i une variable aléatoire discrète représentant l'observation à un instant t pour une cellule donnée. Soit (X_i, Y_i, Z_i) les coordonnées du centre d'une cellule c_i. La projection du centre sur l'image de profondeur est faite en utilisant les équations de projection perspective :

$$u_i = \frac{X_i \times f_x}{Z_i} + c_x, \tag{3.1}$$

$$v_i = \frac{Y_i \times f_y}{Z_i} + c_y, \tag{3.2}$$

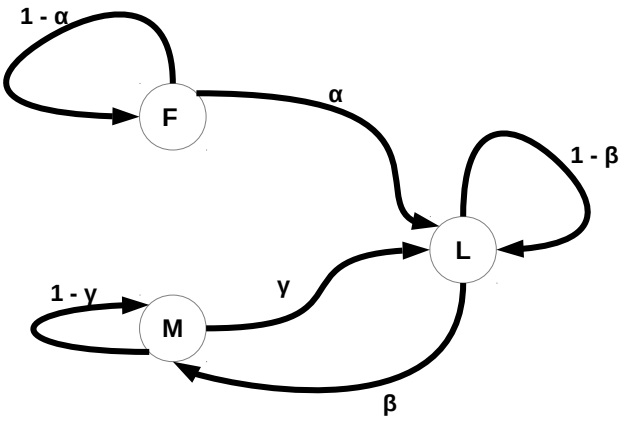


FIGURE 3.7 – HMM à trois états utilisé pour estimer l'état de chaque cellule de la grille.

où (f_x, f_y) et (c_x, c_y) sont respectivement la focale et le centre optique de la caméra. Soit d_i la profondeur du pixel (u_i, v_i) de l'image de profondeur I_t. Nous définissons la variable $\varepsilon_i = d_i - Z_i$. Trois cas de figures existent pour une cellule donnée (voir figure 3.8) :

- $\varepsilon_i < 0$ $(d_i < Z_i) \Rightarrow$ La cellule c_i est non observée (figure 3.8(a)).
- $\varepsilon_i = 0 \Rightarrow$ La cellule c_i est dite touchée, ce qui signifie qu'elle est occupée [3] (figure 3.8(b)).
- $\varepsilon_i > 0$ $(d_i > Z_i) \Rightarrow$ la cellule c_i est dite non touchée, ce qui signifie quelle est visible ou non occupée (figure 3.8(c)).

Ainsi, Y_t^i peut prendre une des valeurs de l'ensemble des symboles $\{hit, miss\}$, où *hit* correspond à une cellule touchée $(\varepsilon_i = 0)$ et *miss* correspond au cas où la cellule n'est pas touchée $(\varepsilon_i > 0)$ mais traversée. Pour le cas où $\varepsilon_i < 0$, la cellule est non observée et ainsi ce cas n'est pas inclus dans l'ensemble des observations. Nous définissons ensuite la probabilité d'émission de chaque symbole comme suit :

$$\begin{aligned} P(Y_t^i = hit \mid Q_t^i = F) &= 1, & P(Y_t^i = miss \mid Q_t^i = F) &= 0, \\ P(Y_t^i = hit \mid Q_t^i = M) &= 1, & P(Y_t^i = miss \mid Q_t^i = M) &= 0, \\ P(Y_t^i = hit \mid Q_t^i = L) &= 0, & P(Y_t^i = miss \mid Q_t^i = L) &= 1. \end{aligned} \tag{3.3}$$

Finalement, pour la probabilité initiale de chaque état, nous avons choisi :

$$\begin{aligned} P(Q_1^i = F) &= 0.5, \\ P(Q_1^i = L) &= 0.5, \\ P(Q_1^i = M) &= 0. \end{aligned}$$

3. En pratique ε_i n'est jamais égale à zéro. Nous montrons comment traiter ce cas en utilisant la fonction de vraisemblance que nous allons introduire dans la suite.

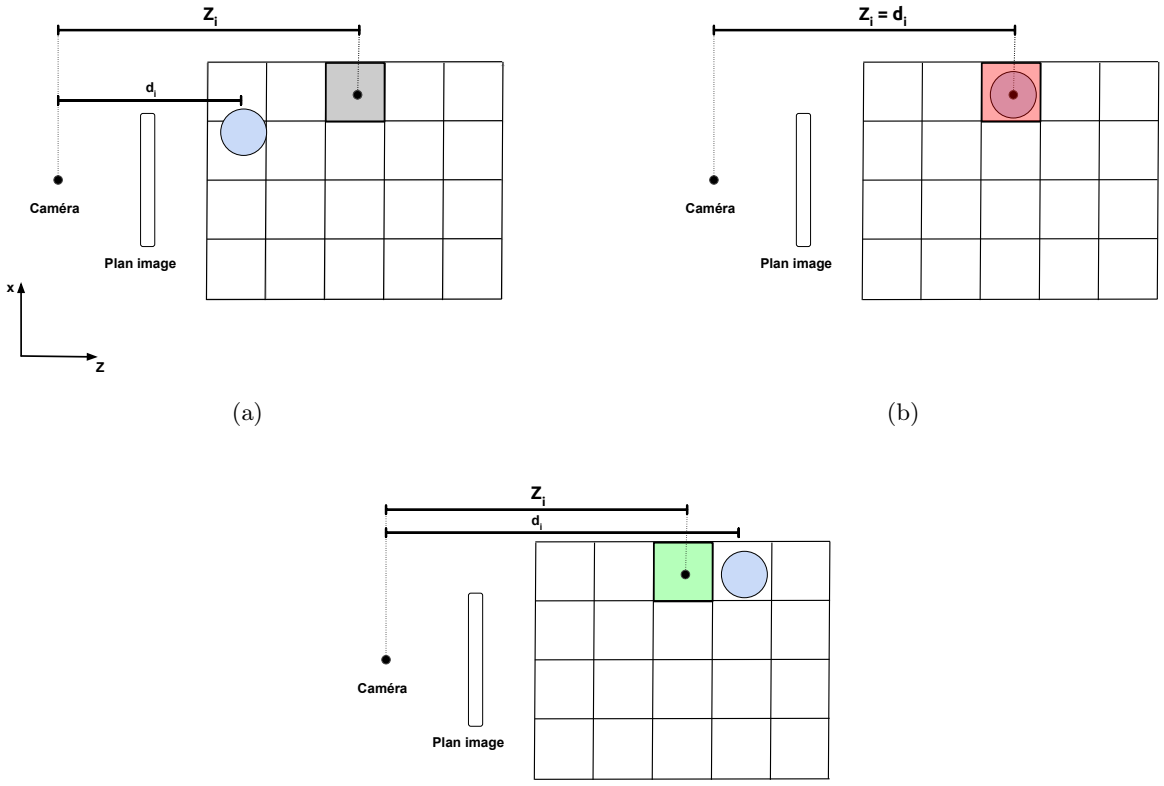


FIGURE 3.8 – L'état d'une cellule peut être dans un des trois cas suivants : a) invisible ($d_i < Z_i$), b) occupée $d_i = Z_i$ ou c) visible $d_i > Z_i$ (Notons que la valeur de la profondeur d_i correspond à la distance sur l'axe Z de l'objet le plus proche à la caméra).

3.3.1 Evidence virtuelle

Dans le HMM décrit ci-dessus, la variable observée Y_t^i représente la mesure obtenue à partir du capteur indiquant si la cellule a été touchée (*hit*) ou non (*miss*). Cependant, cette mesure n'est souvent pas précise à cause du bruit du capteur. Dans un réseau bayésien, il est possible de représenter la certitude sur la mesure d'une variable en lui associant une évidence virtuelle (en anglais *virtual evidence*) introduite par Pearl [Pearl, 1988]. Cette évidence représente une observation faite sur la variable Y_t^i et elle est quantifiée par une fonction de vraisemblance représentant le degré de certitude attribué à chaque valeur observée. Un HMM n'étant qu'un cas particulier d'un réseau bayésien dynamique ([Murphy, 2002]), il est possible d'y insérer une connaissance incertaine sous la forme d'évidences virtuelles. L'insertion de connaissances incertaines dans un HMM constitué de variables discrètes modélisant un raisonnement symbolique a était faite selon ce principe dans les travaux de Jeanpierre [Jeanpierre, 2002] et de Rose [Rose, 2011].

Ainsi, dans notre cas, nous introduisons l'évidence virtuelle $e_{Y_t^i}$ qui porte sur la variable Y_t^i et nous définissons les fonctions de vraisemblances $P(e_{Y_t^i} \mid Y_t^i = hit)$ et $P(e_{Y_t^i} \mid Y_t^i = miss)$ représentant la croyance que nous attribuons à chaque valeur que Y_t^i peut prendre. La nouvelle représentation graphique du HMM est montrée sur la figure 3.9.

Sur la figure 3.10 est affichée la fonction $f(\varepsilon_i)$ que nous avons utilisée pour modéliser l'incertitude associée à la mesure du capteur. Cette fonction couvre les trois cas suivants (voir figure 3.8) :

- $\varepsilon_i < 0$: la cellule est masquée par l'objet le plus proche à la caméra.
- $\varepsilon_i = 0$: la cellule est occupée par l'objet le plus proche à la caméra.
- $\varepsilon_i > 0$: la cellule est libre et le premier objet le plus proche à la caméra se situe derrière la cellule.

A partir de $f(\varepsilon_i)$, nous avons construit les deux fonctions de vraisemblance mesurant le degré de certitude associé à chaque valeur de Y_t^i telles que :

$$\begin{aligned} P(e_{Y_t^i} \mid Y_t^i = hit) &= f(\varepsilon_i), \\ P(e_{Y_t^i} \mid Y_t^i = miss) &= 1 - f(\varepsilon_i). \end{aligned} \tag{3.4}$$

Remarquons sur la figure 3.10, dans le cas où la cellule n'est pas observée ($\varepsilon_i < 0$) se traduit par une vraisemblance uniforme sur les deux états de $Y_t^i(0.5, 0.5)$ puisque aucune connaissance n'est apportée par la mesure.

3.3.2 Inférence

L'inférence dans le HMM représenté par le réseau bayésien dynamique de la figure 3.9 consiste à calculer la densité *a posteriori* $P(Q_t^i \mid e_{Y_{1:t}^i})$ d'une façon récursive à chaque fois une observation

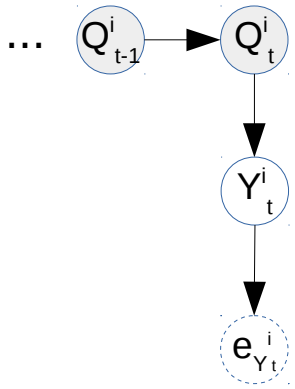


FIGURE 3.9 – Nouvelle représentation graphique avec l'introduction de l'évidence virtuelle $e_{Y_t^i}$.

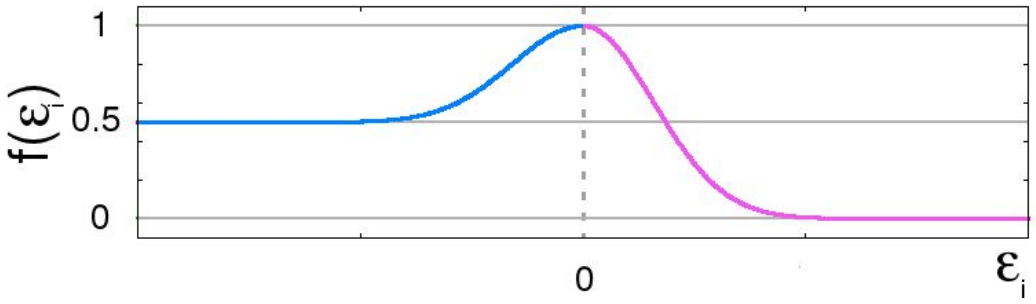


FIGURE 3.10 – La fonction $f(\varepsilon_i)$ représentant la vraisemblance de l'état occupé $P(e_{Y_t^i} \mid Y_t^i = hit)$.

est reçue. Nous avons alors :

$$
\begin{aligned}
P(Q_t^i \mid e_{Y_{1:t}^i}) &= P(Q_t^i \mid e_{Y_{1:t-1}^i}, e_{Y_t^i}) \\
&= K \cdot P(e_{Y_t^i} \mid Q_t^i, e_{Y_{1:t-1}^i}) \cdot P(Q_t^i \mid e_{Y_{1:t-1}^i}) \\
&= K \cdot P(e_{Y_t^i} \mid Q_t^i) \cdot P(Q_t^i \mid e_{Y_{1:t-1}^i}) \text{ Hypothèse Markovienne.} \\
&= K \cdot P(e_{Y_t^i} \mid Q_t^i) \cdot \sum_{Q_{t-1}^i} P(Q_t^i \mid Q_{t-1}^i, e_{Y_{1:t-1}^i}) \cdot P(Q_{t-1}^i \mid e_{Y_{1:t-1}^i}) \\
&= K \cdot P(e_{Y_t^i} \mid Q_t^i) \cdot \sum_{Q_{t-1}^i} P(Q_t^i \mid Q_{t-1}^i) \cdot P(Q_{t-1}^i \mid e_{Y_{1:t-1}^i}) \text{ Hypothèse Markovienne.}
\end{aligned}
\tag{3.5}
$$

K étant la constante de normalisation, représentant la probabilité de l'observation $P(e_{Y_t^i})$. $P(e_{Y_t^i} \mid Q_t^i)$ peut être écrite sous la forme suivante :

$$\begin{aligned} P(e_{Y_t^i} \mid Q_t^i) &= \sum_{Y_t^i} P(e_{y_t^i} \mid Y_t^i, Q_t^i) \cdot P(Y_t^i \mid Q_t^i) \\ &= \sum_{Y_t^i} P(e_{Y_t^i} \mid Y_t^i) \cdot P(Y_t^i \mid Q_t^i) \text{ car } Y_t^i \text{ d-sépare } e_{Y_t^i} \text{ et } Q_t^i. \end{aligned} \tag{3.6}$$

En remplaçant l'équation 3.6 dans 3.5, nous obtenons :

$$P(Q_t^i \mid e_{Y_{1:t}^i}) = K \cdot \sum_{Y_t^i} P(e_{Y_t^i} \mid Y_t^i) \cdot P(Y_t^i \mid Q_t^i) \cdot \sum_{Q_{t-1}^i} P(Q_t^i \mid Q_{t-1}^i) \cdot P(Q_{t-1}^i \mid e_{Y_{1:t-1}^i}), \tag{3.7}$$

avec $P(e_{Y_t^i} \mid Y_t^i)$ est donnée par les équations 3.4, $P(Y_t^i \mid Q_t^i)$ par les équations 3.3, $P(Q_t^i \mid Q_{t-1}^i)$ représente la probabilité de transition et $P(Q_{t-1}^i \mid e_{Y_{1:t-1}^i})$ la probabilité à $t-1$ supposée connue par récursivité.

3.3.3 Modélisation du fond dynamique

Le HMM décrit ci-dessus est capable [4] d'identifier les objets mobiles et fixes de la scène comme le montre la figure 3.11. Par contre, ce HMM ne peut pas apprendre le fond en continu. En effet, un objet identifié comme mobile ne peut jamais devenir fixe même s'il ne bouge plus (par exemple, une chaise qui a été déplacée d'un endroit à l'autre). Cela est lié au fait que le passage de l'état mobile à fixe n'est pas possible dans le HMM (voir figure 3.7).

Pour permettre à la méthode d'apprendre le fond en continu, nous avons modifié le HMM pour autoriser la transition de l'état mobile **M** à fixe **F**. Avec cette transition, un objet qui est dans un état mobile **M** peut redevenir fixe **F** s'il n'a pas bougé après un certain temps. Ainsi, cette nouvelle probabilité de transition, notée ψ, permet au modèle de réintégrer un objet mobile dans le fond s'il ne bouge pas pendant un certain temps. Le temps de transition de l'état **M** vers **F** dépend de la valeur de ψ. Le HMM modifié est représenté sur la figure 3.12. Le résultat obtenu avec ce nouveau HMM est illustré sur la figure 3.13.

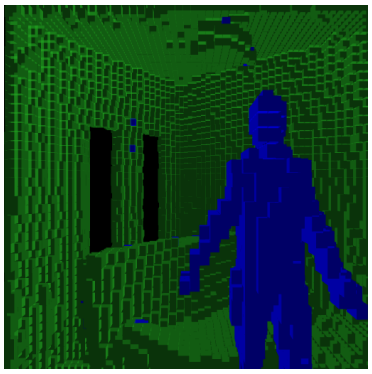


FIGURE 3.11 – Détection des objets mobiles (en bleu) et fixes (en vert) avec le HMM.

4. Nous présentons dans le chapitre 5 une évaluation détaillée de la capacité de la méthode à classifier les cellules de la grille d'occupation.

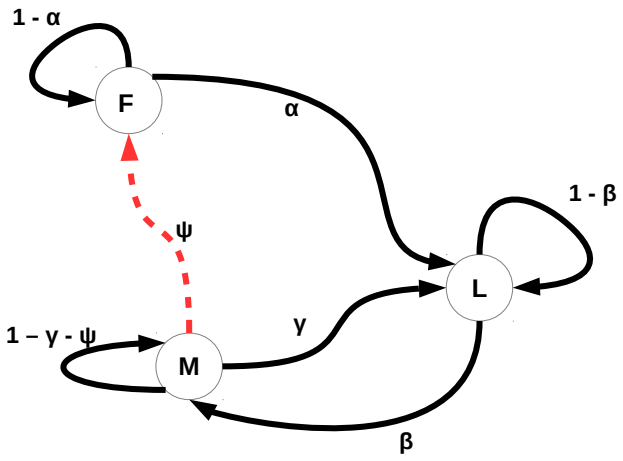


FIGURE 3.12 – Le HMM est modifié avec une nouvelle probabilité de transition ψ de l'état **M** à **F** pour permettre à un objet de passer de mobile à fixe s'il n'a pas bougé pendant un certain temps.

Les différentes étapes de la mise à jour de la grille d'occupation par le nouveau HMM sont représentées dans l'algorithme 5 (page 52) qui prend en entrée une grille d'occupation **G** et une image de profondeur I_t reçue à l'instant t. L'algorithme parcourt toute la grille, et pour chaque cellule c_i, les coordonnées de son centre (X_i, Y_i, Z_i) sont projetées sur le plan image et la valeur d_i qui correspond à la profondeur du pixel projeté est calculée. Ensuite, la valeur $\varepsilon_i = d_i - Z_i$ est calculée et l'état de chaque cellule est mis à jour par le HMM en utilisant l'équation 3.7 suivi d'une étape de normalisation. Finalement, la cellule est classée comme mobile, fixe ou libre en choisissant le maximum *a posteriori* (MAP) qui représente l'état le plus probable.

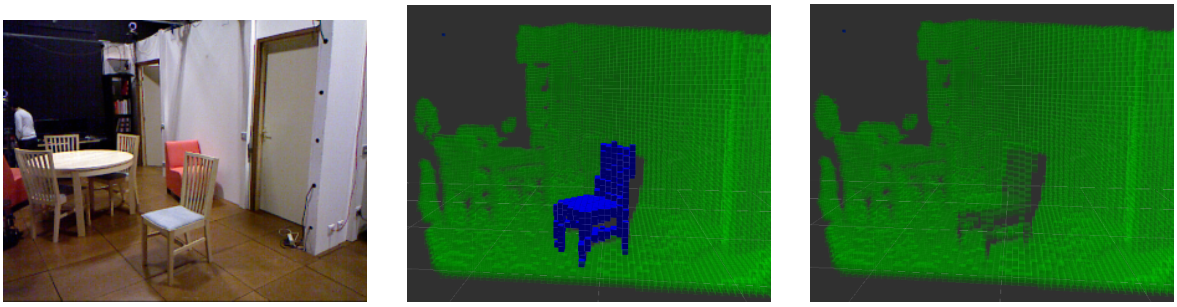
3.4 Extraction de la silhouette

Dans la section précédente, nous avons décrit la première partie de notre méthode qui modélise un environnement dynamique en utilisant une grille d'occupation dont l'état de chaque cellule est mis à jour avec un HMM à 3 états. Nous disposons maintenant d'un ensemble de cellules occupées par des objets mobiles (en bleu sur la figure 3.1) et fixes (en vert sur la figure 3.1). L'objectif de la deuxième partie est de regrouper les cellules mobiles en des étiquettes différentes, suivre ces étiquettes dans le temps et finalement extraire la silhouette de profondeur de chacune de ces étiquettes.

3.4.1 Étiquetage

Un algorithme d'étiquetage [5] (en anglais *Component Labeling*) est utilisé pour regrouper les cellules mobiles retournées par le HMM en des étiquettes différentes. Soit la fonction F définie

5. Il s'agit d'un algorithme classique d'analyse en composantes connexes



(a) (b) (c)

FIGURE 3.13 – a) Image avec une chaise qui a été bougée. b) La chaise est identifiée comme mobile. c) La même chaise est devenue fixe grâce à la version modifiée du HMM représenté sur la figure 3.12.

par :

$$F(c_i) = \begin{cases} 1, & \text{Si l'état de la cellule } c_i \text{ est mobile mobile (M)} \\ 0, & \text{sinon.} \end{cases} \tag{3.8}$$

Cette fonction a pour but d'assigner une valeur de 1 pour les cellules mobiles uniquement dans le but de pouvoir les identifier dans la suite. Soit e_i l'étiquette de la cellule c_i. L'initialisation de l'algorithme d'étiquetage consiste à affecter une étiquette temporaire m pour chaque cellule mobile :

$$e_i = \begin{cases} m, (m = m + 1), & \text{Si F}(c_i) = 1 \\ 0, & \text{sinon.} \end{cases} \tag{3.9}$$

Ensuite, l'algorithme procède itérativement sur l'ensemble des cellules en mettant à jour la valeur de l'étiquette de chaque cellule avec la fonction g :

$$e_i = \begin{cases} g(e_i) & \text{Si } e_i \neq 0 \\ 0, & \text{sinon,} \end{cases} \tag{3.10}$$

avec $g(e_i) = \min(e_i, \mathcal{N}(c_i))$ une fonction qui remplace la valeur de l'étiquette de c_i par la valeur minimale de l'étiquette de son voisinage $\mathcal{N}(c_i)$ dans une fenêtre de $3 \times 3 \times 3$. La fonction g est appliquée jusqu'à ce qu'il n'y ait plus de changement d'étiquette par la fonction g sur toutes les cellules. Au final, les cellules possédant le même index m correspondent à une seule étiquette. Un exemple de fonctionnement de cet algorithme en 2D est illustré sur la figure 3.14. La sortie de l'algorithme d'étiquetage est montrée sur la figure 3.1 où les deux personnes dans la scène sont identifiées comme deux étiquettes différentes. Notons que la version que nous avons implémentée n'est pas optimale en terme de vitesse d'exécution et il existe des variantes ([Cabaret and Lacassagne, 2014]) permettant d'accélérer cet algorithme.

Pour suivre ces étiquettes dans le temps, nous utilisons une méthode simple qui est la suivante : Soit L_t un ensemble d'étiquettes renvoyées par l'algorithme d'étiquetage à l'instant t. A l'instant $t+1$, un nouvel ensemble L'_{t+1} d'étiquettes est renvoyé. Pour suivre les étiquettes dans le temps, il faut associer à chaque étiquette dans L'_{t+1}, l'étiquette correspondante dans L_t. Cette association est faite en cherchant dans L_t l'étiquette dont la distance géométrique est la plus petite à une étiquette de L'_{t+1}.

Algorithm 5: Algorithme de la mise à jour de la grille d'occupation par le HMM.

entrée : grille d'occupation **G**, image de profondeur I_t

pour *chaque cellule* c_i *de la grille* $\boldsymbol{G}$ **faire**

- $u_i = \frac{X_i \times f_x}{Z_i} + c_x$
- $v_i = \frac{Y_i \times f_y}{Z_i} + c_y$
- **si** (u_i, v_i)*dans le plan de la caméra* **alors**
  - $d_i = I_t(u_i, v_i)$
  - $\varepsilon_i = d_i - Z_i$
  - $obs = f(\varepsilon_i)$
  - /* Mise à jour de l'état de chaque cellule par le HMM*/
  - -$P(Q_t^i = L \mid e_{Y_{1:t}^i}) = (1 - obs) \cdot [P(Q_{t-1}^i = F) \cdot \alpha + P(Q_{t-1}^i = M) \cdot \gamma + P(Q_{t-1}^i = L) \cdot (1 - \beta)]$
  - -$P(Q_t^i = M \mid e_{Y_{1:t}^i}) = obs \cdot [P(Q_{t-1}^i = L) \cdot \beta + P(Q_{t-1}^i = M) \cdot (1 - \gamma - \psi)]$
  - -$P(Q_t^i = F \mid e_{Y_{1:t}^i}) = obs \cdot [P(Q_{t-1}^i = F) \cdot (1 - \alpha) + P(Q_{t-1}^i = M) \cdot \psi]$
  - /* Normalisation*/
  - $somme = P(Q_t^i = L \mid e_{Y_{1:t}^i}) + P(Q_t^i = M \mid e_{Y_{1:t}^i}) + P(Q_t^i = F \mid e_{Y_{1:t}^i})$
  - -$P(Q_t^i = L \mid e_{Y_{1:t}^i}) = \frac{P(Q_t^i = L \mid e_{Y_{1:t}^i})}{somme}$
  - -$P(Q_t^i = M \mid e_{Y_{1:t}^i}) = \frac{P(Q_t^i = M \mid e_{Y_{1:t}^i})}{somme}$
  - -$P(Q_t^i = F \mid e_{Y_{1:t}^i}) = \frac{P(Q_t^i = F \mid e_{Y_{1:t}^i})}{somme}$

3.4.2 Reconstruction de la silhouette à partir d'une étiquette

Nous avons maintenant un ensemble d'étiquettes représentant les personnes se déplaçant dans l'environnement. Chaque étiquette est composée d'un ensemble de cellules mobiles. Dans cette section nous décrivons la méthode que nous avons développée pour extraire la silhouette de profondeur [6] de chaque étiquette.

Pour construire la silhouette de profondeur de chaque étiquette, nous calculons la boite englobante d'une étiquette comme le montre la figure 3.1. Cette boite est définie par ses coordonnées M_{min} et M_{max} qui représentent les points 3D minimal et maximal de la boite. Ensuite, pour obtenir la silhouette, les points M_{min} et M_{max} sont projetés sur l'image de profondeur en utilisant les équations de projection perspective 3.1 et 3.2. p_{min} et p_{max} sont les pixels correspondants respectivement à la projection de M_{min} et M_{max}. Chaque pixel $p = (u, v, d)$ contenu dans la zone 2D définie par p_{min} et p_{max} est projeté en 3D en utilisant les équations suivantes :

$$X = \frac{u - c_x}{f_x} \times Z,$$
$$Y = \frac{v - c_y}{f_y} \times Z,$$
$$Z = d,$$

avec $P(X, Y, Z)$ le point 3D qui correspond à un pixel p. Pour décider si p appartient à la

6. Par comparaison à une image de silhouette classique, une silhouette de profondeur est une image dont chaque pixel encode la distance (profondeur) de l'objet le plus proche à la caméra.

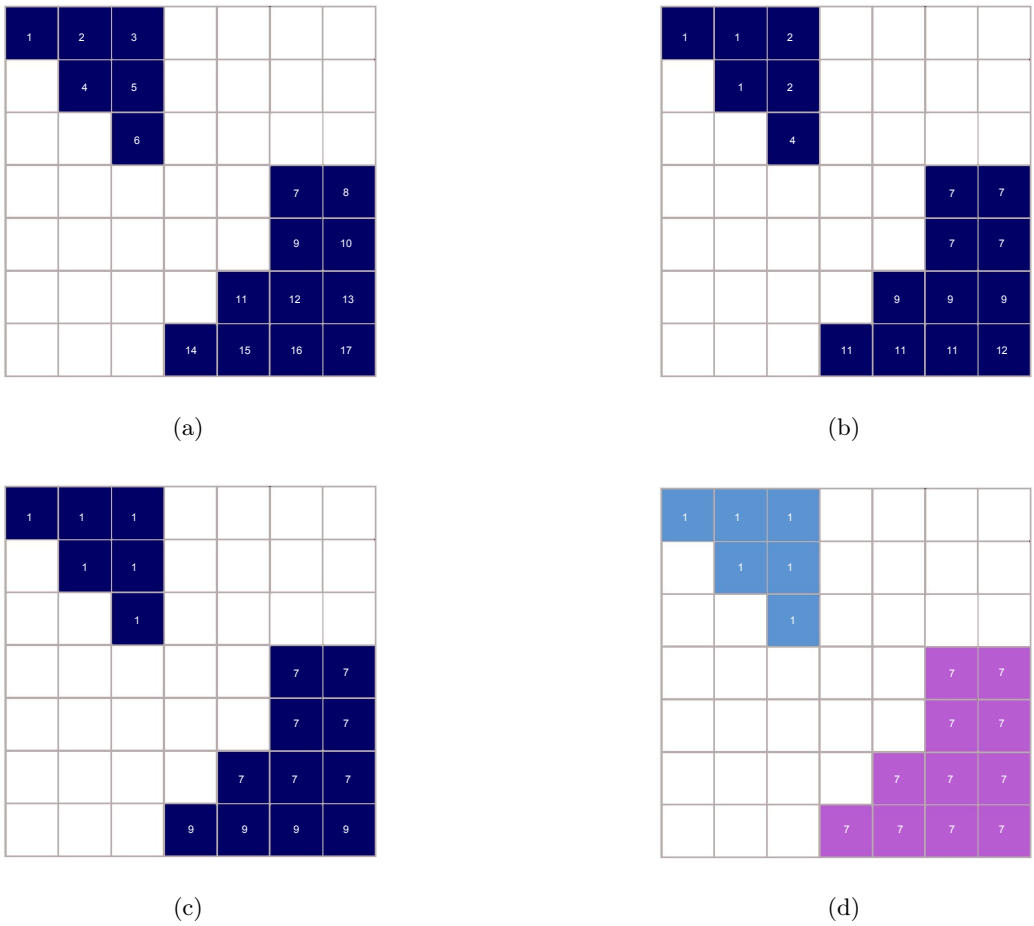


FIGURE 3.14 – Exemple de fonctionnement de l'algorithme d'étiquetage en 2D : a) A l'initialisation, une étiquette temporaire est assignée à chaque cellule mobile (en bleu). b) et c) A chaque étape de l'algorithme, l'étiquette d'une cellule est remplacée par la valeur minimale de l'étiquette de son voisinage dans une fenêtre de 3×3. d) A la fin, lorsqu'il n'y a plus de changement d'étiquettes, l'algorithme s'arrête.

silhouette, son point P correspondant est testé pour savoir s'il se situe à l'intérieur de la boite englobante de l'étiquette et s'il appartient à une cellule mobile. Si les deux conditions sont vérifiées, la profondeur du pixel p est utilisée pour la construction de la silhouette de profondeur. La figure 3.1 montre la silhouette de profondeur reconstruite pour chaque étiquette. L'avantage de cette technique d'extraction de la silhouette est que nous pouvons reconstruire la silhouette à la résolution native du capteur même avec une basse résolution de la grille d'occupation. Nous étudions dans le chapitre 5 la qualité de la silhouette extraite avec différentes résolutions de la grille d'occupation.

3.5 Conclusion

Dans ce chapitre, nous avons décrit la méthode que nous avons développée pour modéliser le fond dynamique de la scène et extraire la silhouette de la personne se déplaçant dans un environnement dynamique. Cette méthode consiste d'abord à discrétiser l'espace en un ensemble

de cellules qui constituent une grille d'occupation. Ensuite, un HMM à 3 états est utilisé pour identifier si chaque cellule est occupée par un objet fixe de la scène, un objet mobile ou si elle est non occupé. Ensuite, un algorithme d'étiquetage est appliqué sur l'ensemble des cellules mobiles pour les regrouper en étiquettes. L'extraction de la silhouette est faite en projetant la boite englobante de chaque étiquette sur l'image de profondeur et en prenant les pixels dont le point 3D correspondant se situe à l'intérieur de cette boite.

La méthode que nous avons développée tourne en temps-réel et possède plusieurs avantages :

- Elle permet d'adapter en continu le fond de la scène en intégrant les objets mobiles dans le fond s'ils ne bougent pas pendant un certain temps.
- Elle permet la reconstruction de la silhouette de la personne à la résolution native du capteur tout en travaillant avec une basse résolution de la grille d'occupation.
- L'utilisation de la grille d'occupation permet d'avoir une représentation unique de l'environnement et intègre naturellement les informations provenant d'un ou plusieurs capteurs hétérogènes fixes ou mobiles.
- La méthode ne nécessite aucune connaissance sur la nature et le type da scène.
- La méthode développée peut être adaptée facilement lorsque la caméra est mobile. Il suffit de mettre à jour la grille d'occupation avec le nuage de points connaissant la bonne transformation décrivant le déplacement de la caméra.
- Le HMM que nous avons mis en place, résout le problème d'*aliasing* qui se traduit par le fait que l'observation d'un objet statique dans la scène est la même que pour un objet mobile.

Notons que nous ne traitons pas l'aspect multi-personne dans ce travail.

En conclusion, nous disposons maintenant d'une méthode capable d'apprendre en continu le fond de la scène et d'extraire la silhouette de la personne dans un environnement dynamique. Cette silhouette sera utilisée par l'algorithme de reconstruction de la posture qui sera détaillé dans le chapitre suivant.

Chapitre 4

Suivi 3D du mouvement humain dans un environnement encombré

Sommaire

4.1 Modèle 3D virtuel 56
4.2 Fonction de vraisemblance 57
4.2.1 Choix de la fonction de vraisemblance 57
4.2.2 Construction de la fonction de vraisemblance 58
4.3 Choix de l'estimateur bayésien 61
4.4 Limites des méthodes existantes pour le suivi dans un environnement avec occlusions 62
4.5 Algorithme de suivi du mouvement humain dans un environnement avec occlusions 62
4.5.1 Nouveau diagramme de condensation 67
4.5.2 Prise en compte de l'état de visibilité du parent d'une partie du corps 69
4.5.3 Identification de l'état de visibilité des parties du corps 71
4.5.4 Recuit-simulé 74
4.5.5 Ré-échantillonnage 75
4.6 Conclusion 75

Nous avons vu dans le chapitre 2 que le problème de suivi 3D du mouvement humain peut être réduit à un problème d'estimation dynamique d'un vecteur d'état à partir d'une observation reçue de la caméra. Dans notre cas, ce vecteur d'état est la configuration d'un modèle 3D virtuel (appelé humanoïde ou avatar) dont nous cherchons à trouver la configuration maximisant la vraisemblance par rapport à la vraie posture humaine observée à partir de la silhouette. Nous avons aussi introduit le filtrage particulaire comme un cadre général permettant d'approximer le vecteur d'état d'un système dont les fonctions de transition et d'observation sont non linéaires. Nous avons ensuite introduit l'algorithme Condensation utilisé pour le suivi d'un objet dans une image. Pour le suivi du mouvement humain, cet algorithme possède un inconvénient majeur qui est le nombre très élevé de particules nécessaires pour garantir un suivi fiable. Nous avons ensuite détaillé trois techniques introduites pour améliorer les performances de cet algorithme. La première technique, appelée APF qui utilise le recuit-simulé avec une politique d'exploration à plusieurs couches en lissant fortement les sommets de la fonction d'observation des premières couches réduisant l'attraction des particules par les maxima locaux et permettant ainsi aux particules de mieux explorer l'espace d'état. Une deuxième technique appelée PS qui représente

l'espace d'état sous une forme factorisée et qui permet de réduire considérablement le nombre des particules requis pour le suivi. Finalement, nous avons détaillé une approche qui combine APF et PS, donnant de meilleurs résultats que les deux approches prises séparément.

Rappelons que notre objectif est de suivre la posture humaine dans un environnement encombré où la silhouette extraite n'est pas toujours observable entièrement à cause des occlusions. Ces conditions rendent la tâche du suivi du mouvement humain difficile à résoudre et les méthodes existantes donnent généralement de mauvais résultats (nous montrons ça dans la suite de ce chapitre). Ainsi, le suivi du mouvement d'une personne en présence des occlusions reste encore un défi scientifique ouvert comme nous l'avons constaté suite à la recherche bibliographique faite pendant cette thèse.

La contribution majeure présente dans ce chapitre est la proposition d'un nouvel algorithme basé sur le filtrage particulaire capable de suivre le mouvement de la personne en présence des occlusions.

Dans la suite de ce chapitre, nous détaillons le modèle 3D virtuel composé d'un ensemble de segments reliés entre eux. Ce modèle simule le mouvement humain et comporte un ensemble de degrés de liberté. Chaque configuration de ce modèle représente un vecteur d'état qui sera comparé à la vraie posture de la personne en utilisant une fonction mesurant la vraisemblance entre les deux. Le degré de similitude est calculé en comparant la projection 2D de la configuration du modèle avec l'image de silhouette extraite à partir de l'image reçue de la caméra (la méthode que nous avons développée pour extraire cette silhouette a été décrite dans le chapitre 3). Ensuite nous décrivons la fonction de vraisemblance que nous avons développée pour mesurer le degré de similitude entre chaque configuration du modèle 3D à la silhouette de la personne. Nous montrons ensuite avec une expérience que les méthodes existantes sont incapables de suivre correctement le mouvement de la personne en présence d'occlusions. Finalement, nous décrivons la modification que nous avons apportée à l'algorithme de filtre particulaire pour suivre le mouvement de la personne en présence d'occlusions.

4.1 Modèle 3D virtuel

Le modèle 3D utilisé est composé de 10 segments articulés, liés entre eux par un squelette en fils de fer comme le montre la figure 4.1. Chaque segment représentant une partie du corps (tête, torse, coude, genoux, cheville, etc) possède un ensemble de degrés de liberté qui correspond à son orientation dans l'espace. En théorie, chaque partie du corps possède 3 degrés de liberté représentant les angles d'Euler autour des axes $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Nous avons réduit le nombre de degrés de liberté pour certaines parties du corps pour obtenir au final un ensemble de 25 degrés de liberté (comme montré sur la figure 4.1). Ainsi le modèle virtuel est configuré à partir d'un vecteur d'état composé de 25 degrés de liberté. Chaque configuration du modèle est obtenue en variant les angles d'Euler des différentes parties du corps. La géométrie de chaque segment est représentée par un cylindre défini par son rayon et sa hauteur fixe. L'ensemble des segments forme une chaîne cinématique représentée sur la figure 4.2 dont le mouvement d'un nœud entraîne le mouvement des nœuds attachés. Chaque nœud de la chaîne représente une articulation possédant au maximum 3 degrés de liberté représentant les angles de rotations sur les différents axes à l'exception du nœud "racine". Ce dernier, possédant 6 degrés de liberté (translation et rotation), représente l'origine du repère attaché au modèle et permet de déplacer et tourner le modèle entièrement. Finalement, pour chaque articulation, nous définissons un intervalle de valeurs admissibles noté I^k décrivant les contraintes bio-mécanique obligeant chaque partie du corps à tourner dans un intervalle bien défini (par exemple, pour la jambe, cet intervalle est

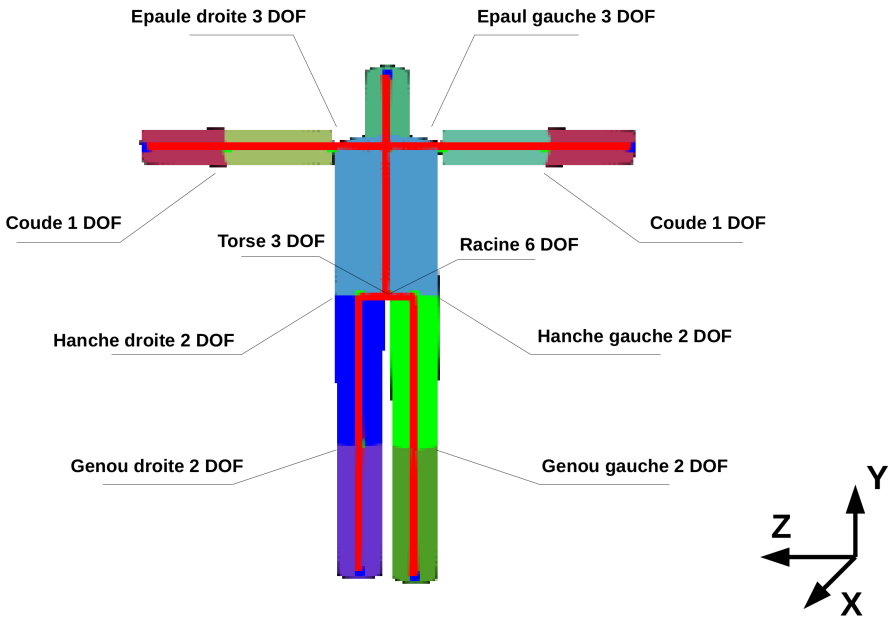


FIGURE 4.1 – Le modèle 3D utilisé pour le suivi est composé de 25 degrés de liberté. La géométrie de chaque partie du corps est représentée par un cylindre.

compris entre -30 et 60 degrés sur l'axe horizontal Z).

4.2 Fonction de vraisemblance

La fonction de vraisemblance a pour but de mesurer le degré de similitude entre une hypothèse donnée qui correspond à une configuration du modèle 3D et l'observation représentée par la silhouette de profondeur extraite de l'image de la caméra RGB-D. Dans la suite nous décrivons plusieurs techniques pour la construction de la fonction de vraisemblance ensuite nous décrivons la technique que nous avons développée.

4.2.1 Choix de la fonction de vraisemblance

Pour construire cette fonction, nous avons choisi d'utiliser uniquement les informations de profondeur fournies par la caméra (pour rappel, une caméra RGB-D fournit une image de couleur et de profondeur). Contrairement à l'image couleur classique, l'image de profondeur fournit des informations additionnelles sur la scène qui permettent d'avoir une meilleure fonction de vraisemblance. Une autre raison pour laquelle nous n'utilisons pas les informations couleurs est de rendre le système de suivi moins intrusif. Avant l'apparition de la caméra de profondeur *lowcost* en 2010, la fonction de vraisemblance était calculée soit à partir d'une image silhouette binaire ou d'une image de contour, soit par le calcul d'histogrammes de couleurs. Pour les caméras de profondeur, il existe actuellement deux méthodes pour construire la fonction de vraisemblance : la première

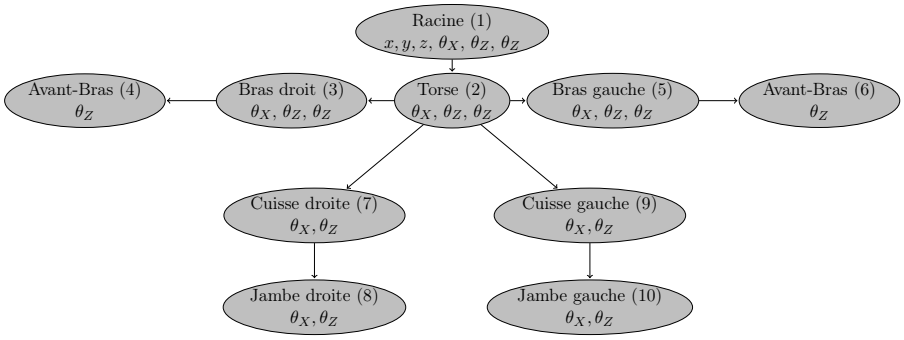


FIGURE 4.2 – Chaîne cinématique représentant le modèle 3D virtuel. Chaque nœud représente une articulation possédant un certain nombre de degrés de liberté.

utilise l'algorithme ICP (*Iterative Closest Point* [Besl and McKay, 1992]) et la deuxième utilise le principe du lancer de rayon. Dans le filtrage particulaire, l'étape la plus coûteuse est le calcul du degré de ressemblance entre chaque particule et l'observation. Ce calcul doit être le plus léger possible parce qu'il est répété pour chaque particule. Dans le cas contraire, l'algorithme risque de prendre beaucoup de temps, ce qui le rend inapproprié pour une application temps réel. L'utilisation d'ICP pour obtenir la fonction de vraisemblance nécessite une étape d'appariement entre les différents points 3D qui est coûteuse en termes de ressources de calcul alors que la méthode basée sur le lancer de rayon consiste à calculer la différence entre la silhouette de profondeur extraite de la caméra et l'image de profondeur synthétique obtenue suite à une projection 2D de la configuration du modèle 3D. Cette méthode ne nécessite pas une étape d'appariement. Ainsi, nous choisissons cette méthode qui est rapide à calculer surtout que nous allons utilisé la carte graphique pour projeter chaque configuration. Ce qui va permettre de calculer rapidement la différence entre les images. Nous détaillons par la suite cette méthode.

4.2.2 Construction de la fonction de vraisemblance

Pour comparer chaque configuration du modèle 3D et l'image de la silhouette, il faut d'abord projeter la configuration 3D dans un plan 2D pour obtenir une image de profondeur synthétique. Cette projection est appelée dans le domaine graphique le "rendu 3D" consistant à générer une image de profondeur 2D à partir d'une scène en 3D. Le principe du rendu 3D est illustré sur la figure 4.3. Un rendu est composé de deux étapes : la première consiste à projeter tous les points 3D de coordonnées (X, Y, Z) en pixels (u, v, d) en utilisant les équations de projection perspective :

$$\begin{aligned} u &= \frac{X \times f_x}{Z} + c_x \\ v &= \frac{Y \times f_y}{Z} + c_y \\ d &= Z, \end{aligned}$$

avec (f_x, f_y) et (c_x, c_y) représentant les coordonnées de la focale et du centre optique de la caméra et d représentant la profondeur du pixel égale à celle du point 3D correspondant. La deuxième étape dans le rendu est appelée "Rastérisation" ou "Matricialisation" et consiste à transformer les facettes entre les pixels projetés en un ensemble de pixels où la profondeur de chaque pixel est interpolée entre la profondeur des sommets du triangle correspondant. Ce processus de rendu

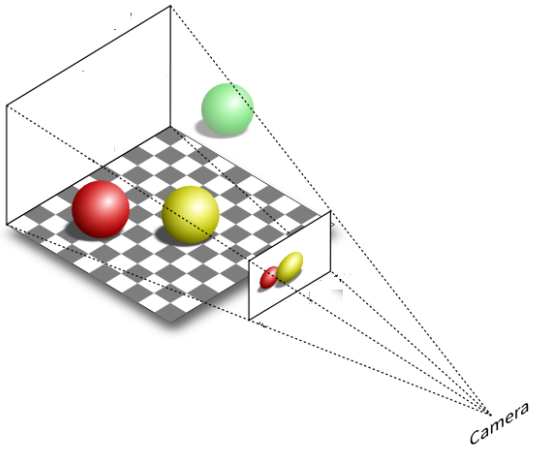


FIGURE 4.3 – Le principe du rendu 3D consiste à générer une image 2D à partir d'une scène 3D (image reproduite de Modern OpenGL tutorial).

3D est délégué à la carte graphique qui permet d'obtenir une image de profondeur synthétique représentant la projection d'une configuration du modèle 3D. Il suffit alors de fournir à la carte graphique les paramètres intrinsèques de la caméra et les points 3D représentant les cylindres et elle génère l'image de profondeur synthétique. Un exemple d'image de profondeur synthétique obtenue par un rendu 3D d'une configuration est montré sur la figure 4.4.a.

Nous pouvons maintenant définir une distance entre la silhouette de profondeur de la personne notée I_R (voir figure 4.4.b) et celle obtenue d'un rendu 3D d'une configuration du modèle notée I_S (voir figure 4.4.a). Cette distance est calculée de la façon suivante :
Soit $S_\cap$ l'intersection entre la projection 2D du modèle et la silhouette de l'image réelle (voir figure 4.4.c), et $S_\cup$ l'union du contour du modèle et la silhouette de l'image réelle. Pour chaque pixel p_i, nous définissons la distance entre sa profondeur $d_s(p_i)$ dans I_s et sa profondeur $d_r(p_i)$ dans I_R par :

$$f(d_s(p_i), d_r(p_i)) = \begin{cases} f_1 & \text{Si } p_i \in S_\cap, \\ 1 & \text{Si } p_i \in S_\cup \backslash S_\cap, \\ 0 & \text{Si } p_i \notin S_\cup, \end{cases}$$

avec la fonction f_1 égale à :

$$f_1 = \begin{cases} \frac{d_s(p_i) - d_r(p_i)}{D} & \text{Si } |d_s(p_i) - d_r(p_i)| \leq D, \\ 1 & \text{Si } |d_s(p_i) - d_r(p_i)| > D. \end{cases}$$

D est un seuil calculé empiriquement qui vaut 10 cm. La distance finale entre les deux images est calculée de la façon suivante :

$$\sigma(I_R, I_S) = \frac{1}{N} \cdot \sum_{i=1}^{N} [f(d_s(p_i), d_r(p_i))]^2, \tag{4.1}$$

Avec N le nombre total de pixels et p_i le pixel courant. $d_s(p_i)$ et $d_r(p_i)$ correspondent à la profondeur du pixel p_i dans I_S et I_R respectivement.

Finalement la fonction de vraisemblance est construite à partir de σ de la façon suivante :

$$p(\mathbf{y_t} \mid \mathbf{x_t}) = \exp(-\sigma). \tag{4.2}$$

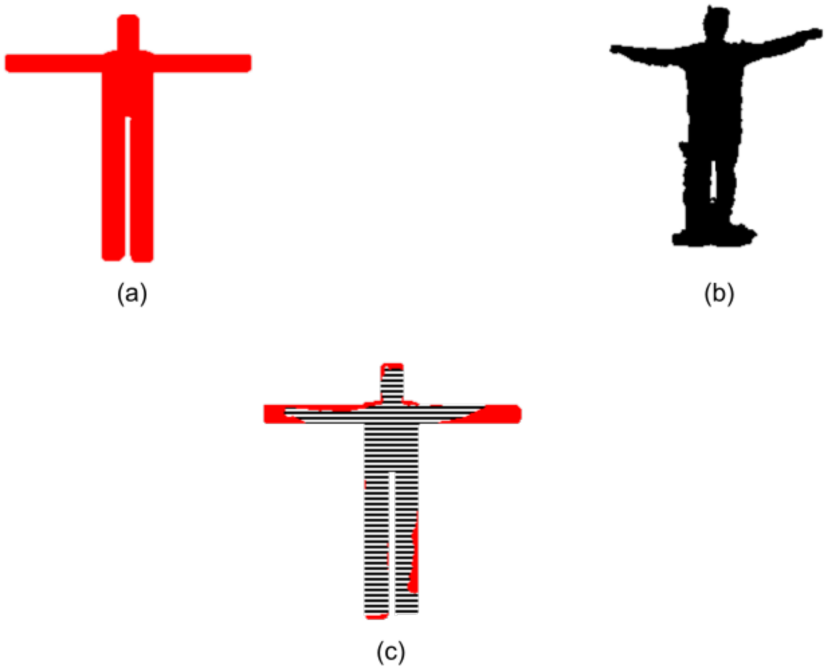


FIGURE 4.4 – a) Une image de profondeur synthétique I_S obtenue par un rendu 3D d'une configuration du modèle 3D. Chaque pixel dans cette image encode la valeur de profondeur qui correspond à sa distance par rapport à la caméra (l'image est représentée en rouge pour une meilleure visibilité). b) La silhouette de profondeur I_R obtenue par la méthode décrite dans le chapitre 3 (l'image est représentée en noire pour une meilleure visibilité). c) La superposition de l'image synthétique I_S et réelle I_R pour le calcul de la distance. La zone d'intersection $S_\cap$ est affichée en rayé.

Multi-Modalité de la fonction de vraisemblance

Pour étudier la forme de la fonction de vraisemblance, nous avons réalisé une expérience consistant à faire varier un seul degré de liberté du modèle 3D dans son intervalle de valeur admissible I^k et en comparant sa projection 2D avec la silhouette de profondeur d'une personne. Nous avons ensuite dessiné les différentes valeurs obtenues pour la fonction de distance $\sigma(I_R, I_S)$ (nous l'appelons par la suite σ)). Sur la figure 4.5(a), un déplacement du modèle 3D est fait sur les axes x, y et z séparément. Pour chaque déplacement, la courbe des valeurs de distance σ est affichée. Nous observons que la courbe obtenue est uni-modale dans ce cas et possède une forme convexe avec une valeur minimale unique. Sur la figure 4.5(b), le modèle 3D est tourné sur lui même dans un intervalle entre 0 et 360 degrés ; la courbe obtenue est bi-modale et possède deux minimums situés à 90 et 270 degrés signifiant que deux configurations du modèle dont l'une est tournée par rapport à l'autre de 180 degrés possèdent la même distance. La même expérience a été répétée pour la cuisse et la jambe gauche (voir figure 4.5(c) et 4.5(d)). La forme de la fonction de distance dans ces deux cas est multi-modale mais possède un minimum global. Finalement, cette expérience nous a permis de vérifier que la fonction de vraisemblance que nous avons développée est multi-modale.

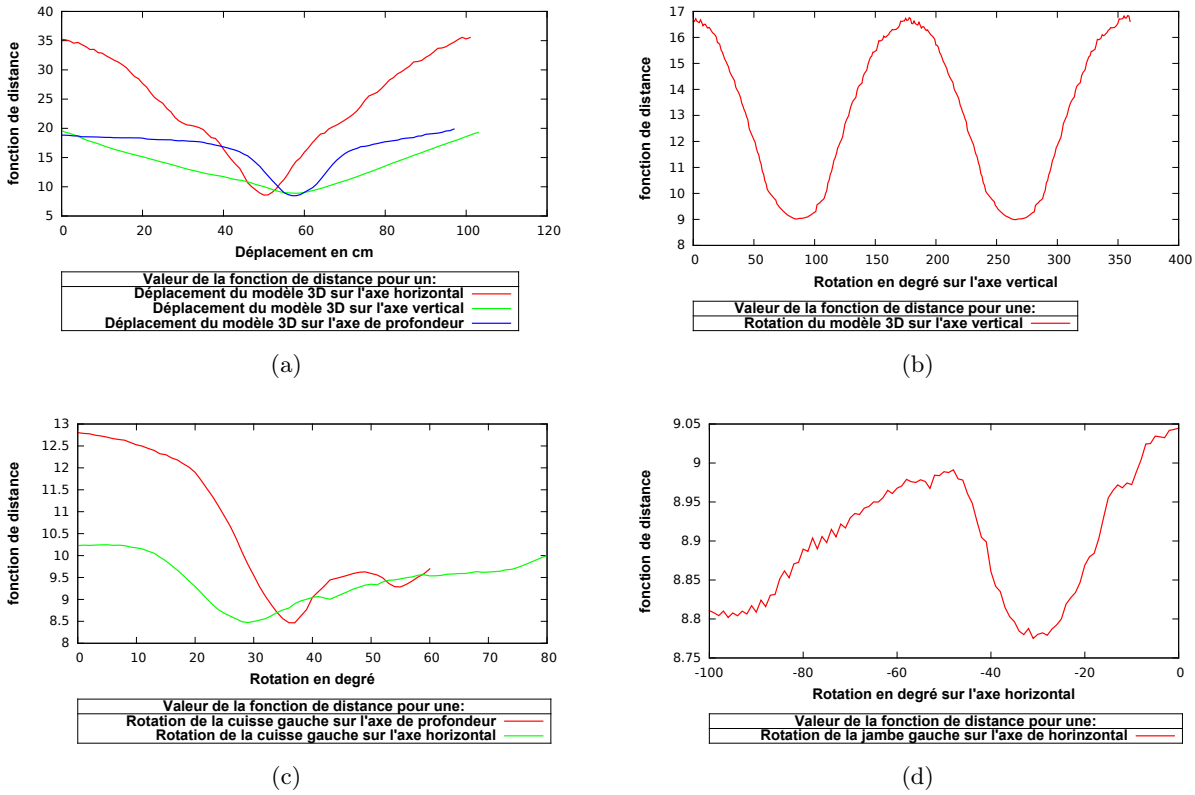


FIGURE 4.5 – La courbe des valeurs de la fonction de distance obtenue suite à une variation d'un seul degré de liberté à la fois.

4.3 Choix de l'estimateur bayésien

Le choix de l'estimateur bayésien dépend de la nature de la fonction d'observation et de la densité *a posteriori* que nous cherchons à estimer. Une contrainte supplémentaire s'ajoute dans notre cas, liée au fait que nous souhaitons suivre le mouvement d'une personne dont certaines parties du corps sont parfois non observables à cause des occlusions. Pour la fonction d'observation, l'expérience que nous avons réalisé pour étudier sa forme dans la section 4.2.2 montre qu'elle est multi-modale et non gaussienne. En effet, l'image synthétique utilisée pour le calcul de la distance avec l'image réelle n'est qu'une projection 2D de la configuration 3D du modèle. Ainsi, plusieurs configurations du vecteur d'état peuvent avoir la même distance, ce qui explique sa multi-modalité (voir la figure 4.5). Cette multi-modalité sera encore plus fréquente en présence des occlusions. Pour la densité *a posteriori*, la forme complexe de la fonction d'observation rend cette densité non gaussienne et multi-modale aussi. Notre conclusion sur la forme non gaussienne et non linéaire de la fonction d'observation et de la densité *a posteriori* pour le problème de suivi du mouvement humain à partir d'une caméra de profondeur rejoint celle obtenue par Deutscher *et al.* [Deutscher *et al.*, 1999] pour le suivi à partir d'une caméra couleur.

L'approche factorisée (PS) introduite dans la section 2.5.2 nous intéresse pour deux raisons. Tout d'abord, elle permet de réduire le nombre de particules en factorisant l'espace d'état en sous-espaces indépendants. Cette factorisation permet aussi de traiter les différentes parties du corps séparément ce qui est intéressant pour régler le problème lié aux occlusions. En effet, nous souhaitons détecter si chaque partie du corps est visible ou non, ainsi PS est un choix intuitif

pour nous. L'approche par recuit-simulé (APF) introduite dans la section 2.5.1 est très efficace dans le cas où la fonction de vraisemblance est multi-modale. Cette approche permet à une faible population de particules de s'échapper des maximums locaux et de migrer graduellement vers le maximum global, il est ainsi important pour nous de combiner l'approche APF avec PS pour un meilleur filtrage. Par contre ces deux approches fournissent des mauvais résultats en présence des occlusions. Nous montrons dans la suite avec une expérience que nous avons réalisé la limitation de ces méthodes dans ces conditions pour ensuite proposer la modification que nous avons apportée pour traiter le problème lié aux occlusions.

4.4 Limites des méthodes existantes pour le suivi dans un environnement avec occlusions

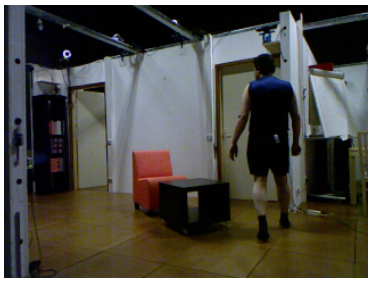
Nous présentons dans cette section l'expérience que nous avons réalisé pour montrer les limites des approches par filtrage particulaire pour suivre le mouvement d'une personne dans un environnement avec occlusions.

Sur la figure 4.6, à $t = 0$, une personne est initialement visible entièrement (figure 4.6(a)), sa silhouette est parfaitement extraite (figure 4.6(b), et le résultat du suivi par l'algorithme PS est affiché sur la figure 4.6(c). A $t = 2$ sec, la personne est dans une zone occultée (figure 4.6(d)), seule la partie haute de sa silhouette est observable (figure 4.6(e)). Une mauvaise posture est obtenue par l'algorithme PS comme le montre la figure 4.6(f). En effet, lorsque les jambes ne sont pas visibles, la silhouette observée (figure 4.6(e)) occupe une petite zone de l'image à cause des occlusions. Par conséquent, l'algorithme par sa nature génère des configurations dont la surface est réduite, ainsi il va essayer de plier les jambes sur le torse, déplacer le modèle loin de la caméra, tourner le torse dans l'axe de la caméra. Le résultat obtenu est celui montré sur la figure 4.6(f). L'impact des occlusions n'est pas juste limité à l'instant où elles se produisent, mais le suivi est également affecté après les moments d'occlusions car l'algorithme continue de générer des configurations non pertinentes. Sur la figure 4.6(g), malgré le fait que la personne soit sortie de la zone occultée et que sa silhouette devienne entièrement visible (figure 4.6(h)), l'algorithme de suivi est dans l'échec. Cette expérience a été répétée pour l'algorithme APF et nous avons obtenue le même résultat d'échec. Le résultat obtenu par cette expérience peut être généralisé pour tous les algorithmes minimisant une fonction de distance.

D'après ces différents résultats, nous pouvons voir que les occlusions posent un vrai problème pour le suivi du mouvement et sont inévitables dans notre cas, car nous souhaitons suivre le mouvement d'une personne dans un environnement du quotidien qui est souvent encombré.

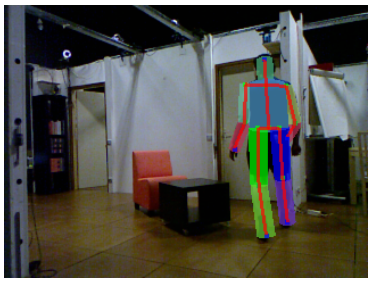
4.5 Algorithme de suivi du mouvement humain dans un environnement avec occlusions

L'algorithme que nous souhaitons développer doit pallier le problème causé par les occlusions que nous avons décrit dans la section précédente. Pour trouver une solution à ce problème, il faut étudier et comprendre son origine. Pour cela, reprenons la figure 4.6(d) où la personne est derrière un meuble et seule la partie supérieure de la silhouette de profondeur est observable (figure 4.6(e)). En effet, les jambes invisibles sont à l'origine du problème, et l'algorithme essaie de les positionner malgré leur absence, en produisant des configurations non pertinentes amenant au final l'algorithme à un échec complet. Une solution intuitive est d'identifier et de retirer les parties du corps non visibles du processus de l'estimation du mouvement. En d'autres termes,

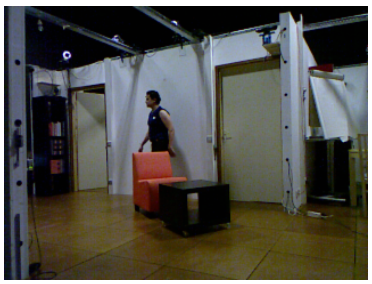


(a) $t = 0$: La personne visible entièrement

(b) $t = 0$ sec : La silhouette visible entièrement

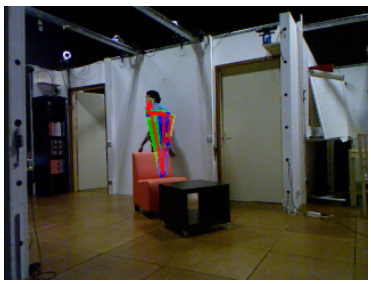


(c) $t = 0$ sec : Résultat du suivi par l'algorithme PS

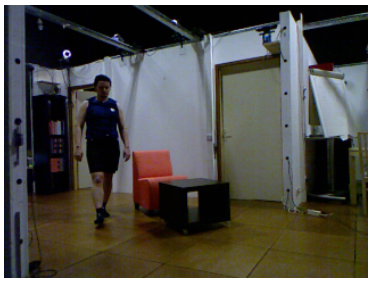


(d) $t+2$ sec : La personne dans une zone occultée

(e) $t+2$ sec : La silhouette partiellement observable

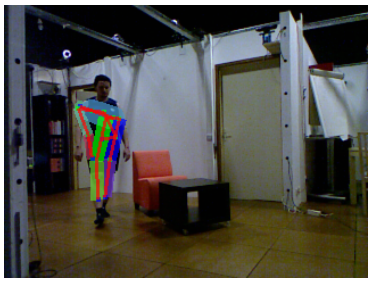


(f) $t+2$ sec : Résultat du suivi par l'algorithme PS



(g) $t+3.2$ sec : La personne visible à nouveau entièrement

(h) $t+3.2$ sec : La silhouette à nouveau visible entièrement



(i) $t+3.2$ sec : Résultat de suivi par l'algorithme PS

FIGURE 4.6 – Exemple montrant l'incapacité des algorithmes de filtrage particulaire à suivre le mouvement d'une personne dont la silhouette est partiellement observable.

si nous sommes capables à chaque instant d'identifier les parties du corps non visibles par la caméra, nous pouvons ainsi les retirer du processus de l'estimation du mouvement. De cette façon, l'algorithme continue à générer des configurations uniquement pour les parties visibles du corps et, réglant ainsi, le problème causé par les occlusions.

Pour mettre en place la solution décrite ci-dessus, nous avons besoin :

— d'une méthode d'estimation du mouvement de la personne qui soit capable d'ajouter ou de retirer des parties du corps suivant leurs états de visibilité et,

— d'un algorithme pour déterminer à chaque instant si une partie du corps est visible ou non.

Pour l'algorithme d'estimation du mouvement, nous avons choisi de reprendre la méthode factorisée (PS) et de la modifier pour qu'elle soit capable d'ajouter ou de retirer des parties du corps suivant leurs états de visibilité. Le choix de la méthode PS est judicieux puisque nous souhaitons traiter séparément chaque partie du corps pour savoir si elles sont visibles ou non et l'approche

PS permet d'avoir une représentation factorisée de l'espace d'état en sous-espaces.

Dans la suite nous décrivons la modification apportée à l'algorithme PS, et ensuite nous décrivons la méthode que nous avons développé pour identifier si une partie du corps est visible ou non.

Reprenons l'algorithme PS qui suppose que les espaces d'état X et d'observation Y peuvent être factorisés en un ensemble de K sous-espaces tel que : $X = X^1 \times \cdots \times X^K$ et $Y = Y^1 \times \cdots \times Y^K$. Dans notre cas, les K sous-espaces représentent les différentes parties du corps qui sont reliées suivant l'arbre cinématique de la figure 4.2.

Soit X_t le vecteur d'état à l'instant t qui représente une configuration du modèle 3D tel que : $X_t = (X_t^1, \cdots X_t^K)$ obtenue en variant les angles d'Euler des différentes parties du corps. Ces angles représentent les différents degrés de liberté de chaque partie du corps (voir figure 4.2). Soit $\{x_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N}$ l'ensemble de N particules utilisé pour approximer la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. Chaque particule $x_t^{(i)}$ représente une hypothèse sur le vecteur d'état et forme un vecteur de tuples tel que $x_t^{(i)} = (x_t^{(i),(j)})^{j:1\cdots K}$. La meilleure solution qui correspond à l'estimation de la configuration optimale retournée par PS noté χ_t est égale à la somme des particules pondérées par le poids de chacune :

$$\chi_t = \sum_{i=1}^{N} w_t^{(i)} \cdot x_t^{(i)}, \tag{4.3}$$

avec χ_t est un vecteur composé de K sous configurations $(\chi_t^1, \cdots \chi_t^K)$.

PS repose sur deux hypothèses, la première suppose que la dynamique du système peut être décomposée de la façon suivante :

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = f_t^K \circ f_t^{K-1} \circ \cdots f_t^j \circ \cdots \circ f_t^1(\mathbf{x}_{t-1}),$$

avec f_t^j modifie uniquement le sous-espace X^j et elle est égale à :

$$f_t^j = p(\mathbf{x}_t^j \mid \mathbf{x}_{t-1}^{(i),j}, \mathbf{x}_t^{(i),1:j-1}).$$

Pour le problème de suivi du mouvement humain, les sous-espaces représentent les différentes parties du corps. L'ordre dans laquelle ces sous-espaces sont explorés est arbitraire, et il est défini au moment de la conception du filtre. Dans notre cas, l'exploration est faite suivant l'ordre topologique indiqué sur la figure 4.2. La deuxième hypothèse suppose que la fonction d'observation se factorise sous forme d'un produit des fonctions d'observations des différents sous-espaces :

$$p(\mathbf{y}_t \mid \mathbf{x}_t) = \prod_{j=1}^{K} p_t^j(\mathbf{y}_t \mid \mathbf{x}_t^j, \mathbf{x}_t^{j-1}, \cdots, \mathbf{x}_t^1),$$

Ainsi la densité *a posteriori* est construite au fur et à mesure en appliquant un filtre particulaire classique, d'une façon séquentielle, sur chaque sous-espace comme l'indique le diagramme de condensation de la figure 4.7, avec le symbole $*$ représentant l'étape de propagation des particules par la fonction de proposition f_t^j qui représente la dynamique du système qui modifie uniquement le sous-espace X^j. Le symbole $\times$ correspond à l'étape de correction qui consiste à affecter un poids à chaque particule suivant la fonction d'observation p_t^j et le dernier symbole $\sim$ représente l'étape de ré-échantillonnage. Ce diagramme prend en entrée un ensemble de particules estimant $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, propage les particules par la fonction f_t^1 appliquée sur le sous-espace X^1, ensuite il les corrige en utilisant la fonction d'observation p_t^1 appliquée sur le sous-espace X^1 et puis il

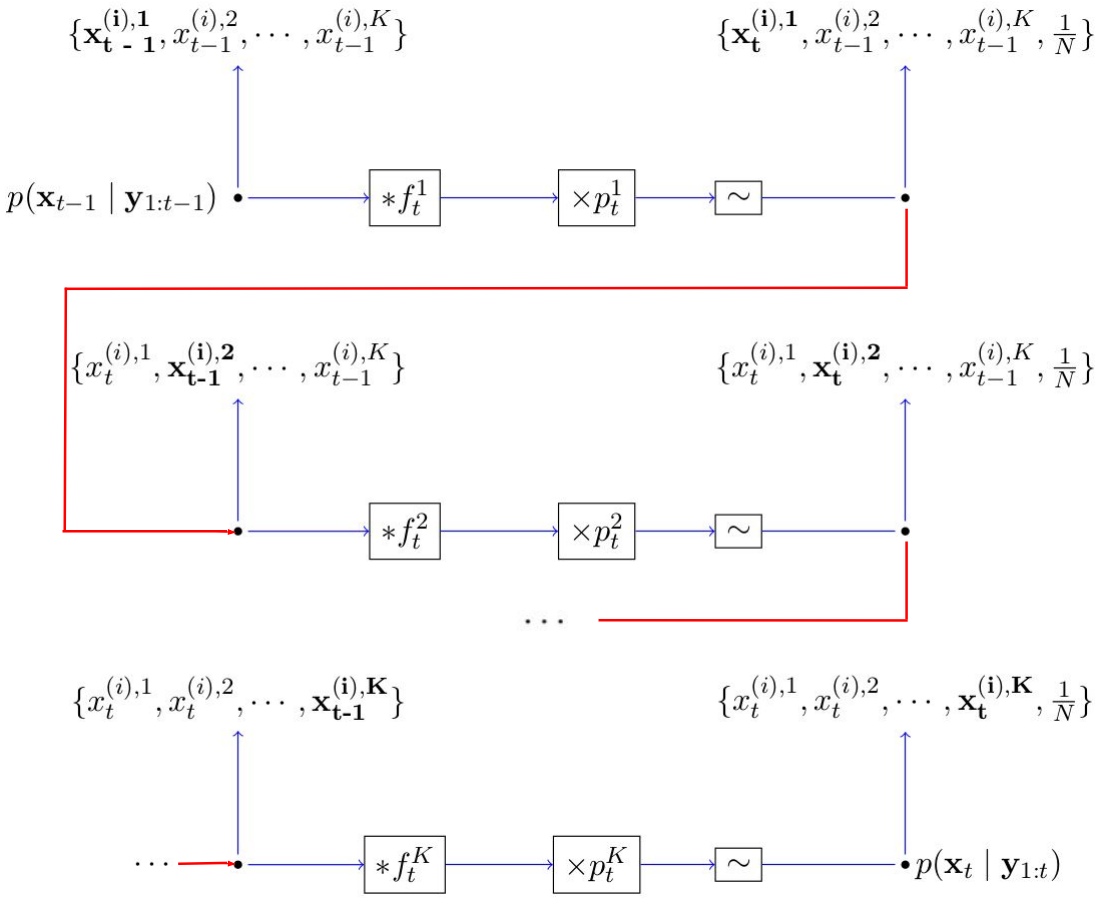


FIGURE 4.7 – Diagramme de condensation de l'algorithme PS prenant en entrée un ensemble de particules qui estiment $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, il propage les particules par la fonction de proposition f_t^1 appliquée sur le sous espace X^1, ensuite il les corrige en utilisant la fonction p_t^1 et puis il les ré-échantillonne. Le processus est ensuite répété pour les K sous-espaces et l'ensemble de particules obtenu au final estime la densité *a posteriori*.

les ré-échantillonne. Ce processus est répété pour les K sous-espaces et l'ensemble des particules obtenu au final estime la densité *a posteriori*.

Pour illustrer le fonctionnement de l'algorithme PS, prenons l'exemple de la figure 4.8 (page 68). Dans cet exemple, l'espace d'état X est décomposé en $K = 3$ sous-espaces tel que :

$$X = X^{\text{torse}} \times X^{\text{bras gauche}} \times X^{\text{bras droit}}.$$

La dynamique du système f_t peut être alors décomposée en $K = 3$ sous-espaces de la façon suivante :

$$f_t = f_t^{\text{bras droit}} \circ f_t^{\text{bras gauche}} \circ f_t^{\text{torse}},$$

avec :

$$\begin{aligned} f_t^{\text{torse}} &= p(\mathbf{x}_t^{\text{torse}} \mid \mathbf{x}_{t-1}^{(i),\text{torse}}), \\ f_t^{\text{bras gauche}} &= p(\mathbf{x}_t^{\text{bras gauche}} \mid \mathbf{x}_{t-1}^{(i),\text{bras gauche}}, \mathbf{x}_t^{(i),\text{torse}}) \\ f_t^{\text{bras droit}} &= p(\mathbf{x}_t^{\text{bras droit}} \mid \mathbf{x}_{t-1}^{(i),\text{bras droit}}, \mathbf{x}_t^{(i),\text{torse}}, \mathbf{x}_t^{(i),\text{bras gauche}}). \end{aligned}$$

De cette manière, la dynamique du système résulte de la composition des dynamiques associées à chaque sous-espace selon un ordre prédéfini qui constitue une stratégie d'exploration. Cette décomposition signifie que la dynamique des différentes parties du corps est traitée d'une façon séquentielle. Ceci permet de réduire considérablement le nombre de particules, puisque l'évaluation des hypothèses sur la position du torse permet de réduire l'espace de recherche sur les bras. $\{x_{t-1}^{(i)}\}_{i=1}^{N} = \{x_{t-1}^{(i),\text{torse}}, x_{t-1}^{(i),\text{bras gauche}}, x_{t-1}^{(i),\text{bras droit}}\}_{i=1}^{N}$ représente l'ensemble des particules estimant la densité *a priori* $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$. Chaque ligne de la figure 4.8 représente un filtre particulaire classique appliqué sur un sous-espace de X. L'algorithme commence par le torse, ensuite le bras gauche et termine avec le bras droit. Ainsi, la première étape de l'algorithme PS (la première ligne de la figure 4.8) consiste à propager les particules par la fonction f_t^{torse} appliquée sur le sous-espace X^{torse} de la façon suivante :

$$x_t^{(i),\text{torse}} \sim p(\mathbf{x}_t^{\text{torse}} \mid \mathbf{x}_{t-1}^{(i),\text{torse}}).$$

En pratique, cette propagation consiste à faire évoluer les particules dans le sous-espace du torse uniquement. Cette évolution utilise un déplacement aléatoire qui se traduit par l'ajout d'un bruit gaussien sur les différents angles d'Euler représentant les degrés de liberté du torse. Ensuite, les particules sont corrigées en utilisant la fonction d'observation du sous-espace X^{torse}. Cette étape revient à calculer un poids $w_t^{(i)}$ pour chaque particule de la façon suivante :

$$w_t^{(i)} \propto p_t(\mathbf{y}_t \mid \mathbf{x}_t^{(i),\text{torse}}).$$

Une étape de ré-échantillonnage est appliquée et un nouvel ensemble de particules de même poids est généré tel que :

$$\{\langle x_t^{(i),\text{torse}}, x_{t-1}^{(i),\text{bras gauche}}, x_{t-1}^{(i),\text{bras droit}}\rangle, \frac{1}{N}\}_{i=1}^{N}.$$

La deuxième étape de l'algorithme (la deuxième ligne de la figure 4.8) propage les particules par la fonction $f_t^{\text{bras gauche}}$ appliquée sur le sous-espace $X^{\text{bras gauche}}$ en partant de la position du torse retrouvée à l'étape précédente :

$$x_t^{(i),\text{bras gauche}} \sim p(\mathbf{x}_t^{\text{bras gauche}} \mid \mathbf{x}_{t-1}^{(i),\text{bras gauche}}, \mathbf{x}_t^{(i),\text{torse}}).$$

L'étape de correction est faite de la façon suivante (toujours en partant de la position du torse retrouvée à l'étape précédente) :

$$w_t^{(i)} \propto p_t(\mathbf{y}_t \mid \mathbf{x}_t^{(i),\text{torse}}, \mathbf{x}_t^{(i),\text{bras gauche}}).$$

Suite à l'étape de ré-échantillonnage, un nouvel ensemble de particules est généré tel que :

$$\{\langle x_t^{(i),\text{torse}}, x_t^{(i),\text{bras gauche}}, x_{t-1}^{(i),\text{bras droit}}\rangle, \frac{1}{N}\}_{i=1}^{N}.$$

La dernière étape de l'algorithme propage, corrige et ré-échantillonne les particules sur le sous-espace $X^{\text{bras droit}}$, avec l'étape de propagation définie comme suit :

$$x_t^{(i),\text{bras droit}} \sim p(\mathbf{x}_t^{\text{bras droit}} \mid \mathbf{x}_{t-1}^{(i),\text{bras droit}}, \mathbf{x}_t^{(i),\text{torse}}, \mathbf{x}_t^{(i),\text{bras gauche}}).$$

Et l'étape de correction :

$$w_t^{(i)} \propto p_t(\mathbf{y}_t \mid \mathbf{x}_t^{(i),\text{torse}}, \mathbf{x}_t^{(i),\text{bras gauche}}, \mathbf{x}_t^{(i),\text{bras droit}}).$$

L'ensemble de particules obtenu suite à l'étape de ré-échantillonnage :

$$\{\langle x_t^{(i),\text{torse}}, x_t^{(i),\text{bras gauche}}, x_t^{(i),\text{bras droit}}\rangle, \frac{1}{N}\}_{i=1}^{N},$$

estime la densité *a posteriori* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. De cette façon, PS construit $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ en appliquant une stratégie d'exploration qui consiste à explorer les différents sous-espaces de X l'un après l'autre suivant un ordre topologique défini au moment de la conception du filtre.

Dans l'algorithme PS, la stratégie d'exploration détermine l'ordre dans lequel les particules explorent l'espace d'état, et dépend de la manière dont l'espace est factorisé. Quelques exemples de stratégies d'exploration ont été illustrés sur la figure 2.4. Un algorithme PS typique possède une seule stratégie d'exploration définie au moment de la conception du filtre et l'algorithme n'est pas capable de modifier cette stratégie pendant l'exécution. Nous souhaitons développer un algorithme capable de modifier d'une façon automatique sa stratégie d'exploration en retirant ou en ajoutant des sous-espaces selon qu'ils soient visibles ou non. Par exemple, reprenons le cas de la figure 4.6, lorsque la silhouette de la personne est visible entièrement, l'algorithme doit générer la stratégie de la figure 4.9(a) qui commence par le torse, ensuite la cuisse droite, cuisse gauche, jambe droite, jambe gauche, etc (le numéro indiqué sur chaque partie du corps indique son ordre dans l'exploration). Dans le cas où les jambes et les cuisses ne sont pas visibles, l'algorithme doit générer automatiquement une nouvelle stratégie en retirant de l'exploration les sous-espaces correspondants aux parties du corps invisibles pour obtenir une nouvelle stratégie comme montré sur la figure 4.9(b). Nous décrivons dans la suite la modification que nous avons apportée à PS pour lui permettre d'adapter et de changer d'une façon automatique sa stratégie d'exploration.

4.5.1 Nouveau diagramme de condensation

Le choix d'inclure ou non un sous-espace X^j à l'instant t dans la stratégie d'exploration dépend de la visibilité ou non de la meilleure sous configuration χ_{t-1}^j trouvée par l'algorithme à l'instant $t-1$ (voir équation 4.3). Supposons alors que nous disposons d'une fonction $\bar{u}$ qui prend en paramètre une sous configuration de χ_t et renvoie 1 si elle est visible par la caméra et 0 sinon.

$$\bar{u}(\chi_t^j) = \begin{cases} 1 & \text{Si } \chi_t^j \text{ est visible,} \\ 0 & \text{Si } \chi_t^j \text{ n'est pas visible,} \end{cases}$$

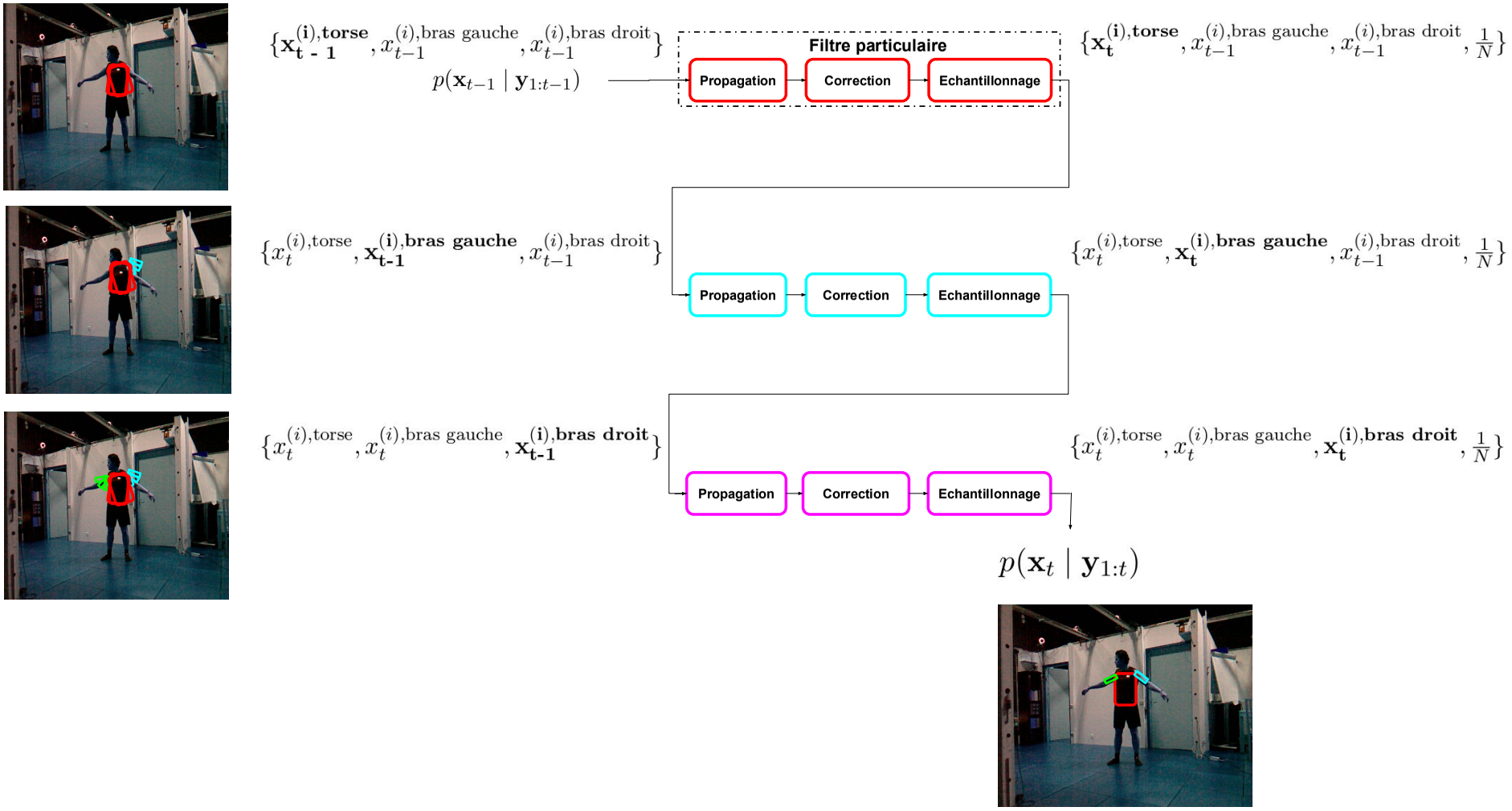
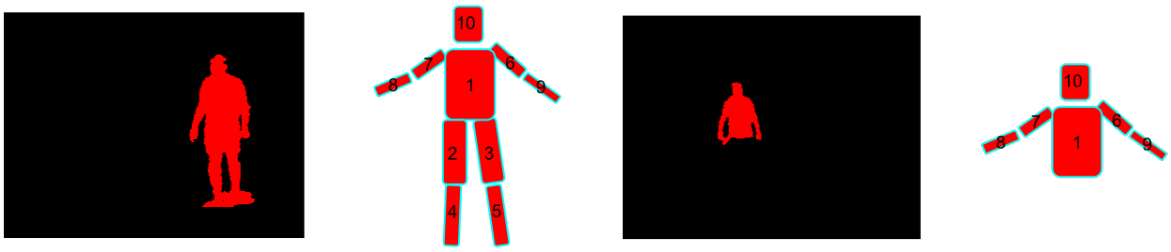


FIGURE 4.8 – Exemple de fonctionnement de l'algorithme PS.



(a) (b)

FIGURE 4.9 – Changement de la stratégie d'exploration. a) Lorsque toutes les parties du corps sont visibles, une stratégie possible consiste à explorer tous les sous-espaces dans l'ordre défini par le numéro affiché sur chaque partie du corps. b) Lorsque les jambes et les cuisses sont invisibles, une nouvelle stratégie d'exploration est générée consistant à retirer les parties basses du corps de l'exploration.

avec $1 \leq j \leq K$.

Nous définissons l'ensemble Q_t des sous-espaces tel que :

$$Q_t = \{X^j / \bar{u}(\chi_{t-1}^j) = 1\}^{j:1\cdots K},$$

avec $\text{Card}(Q_t) = P \leq K$. Q_t contient les sous-espaces dont la sous configuration de chacun de ses sous-espace est visible. Ainsi, une première version de l'algorithme consiste à construire Q_t comme un nouvel espace à explorer par l'ensemble des particules à chaque instant t en utilisant la connaissance *a priori* sur l'état du système représenté par χ_{t-1}. La figure 4.10.b montre le nouveau diagramme de condensation proposé avec l'introduction de la fonction $\bar{u}$.

Avec le diagramme actuel, χ_t ne contient que les sous configurations des parties visibles du corps. Pour pouvoir tester si les parties identifiées comme invisibles sont à nouveau visibles ou non, il faut mémoriser leurs dernières configurations lorsqu'elles étaient visibles. En effet, prenons l'exemple suivant : Supposons qu'une partie du corps X^j ait été identifiée comme invisible à l'instant t grâce à fonction $\bar{u}$ appliquée sur sa configuration χ_{t-1}^j. Ainsi χ_t renvoyée par l'algorithme à t ne contient que les configurations des parties du corps visibles. A $t+1$, il est impossible de tester la visibilité de la partie du corps X^j à nouveau puisque sa configuration χ_{t-1}^j n'a pas été mémorisée. Il faut donc introduire une mémoire dans ce nouveau diagramme pour garder la trace des parties invisibles du corps et continuer à tester leur visibilité dans les prochains instants. Pour introduire cette mémoire dans notre algorithme, nous définissons l'ensemble R_t qui contient toutes les sous configurations invisibles retrouvées tel que :

$$R_t = \{\chi_{t-1}^j / \bar{u}(\chi_{t-1}^j) = 0\}^{j:1\cdots K},$$

avec $\text{Card}(R_t) = K - P$. Nous complétons ainsi χ_t par les sous configurations de R_t tel que : $\chi_t = \chi_t \cup R_t$. Le nouveau diagramme avec mémoire est présenté sur la figure 4.11. Avec ce nouveau diagramme, les configurations des parties invisibles du corps sont propagées dans le temps ce qui permet à chaque instant de tester si elles sont visibles à nouveau ou pas.

4.5.2 Prise en compte de l'état de visibilité du parent d'une partie du corps

Une autre modification à apporter à PS consiste à prendre en compte l'état de visibilité du parent de chaque partie du corps pour éviter le problème que nous décrivons ici. Soit le scénario

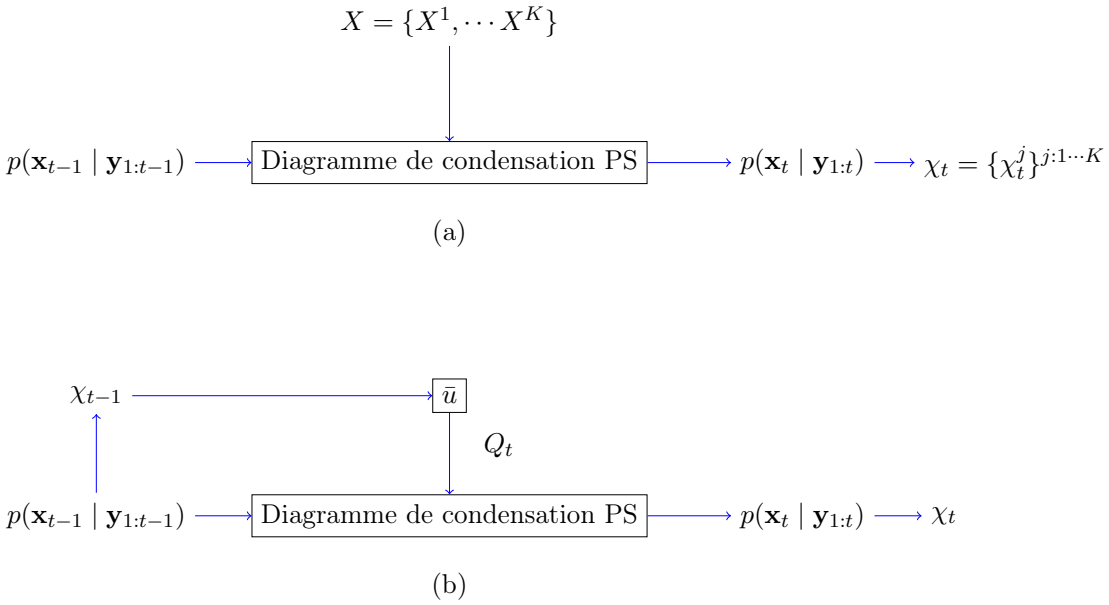


FIGURE 4.10 – Comparaison du diagramme de condensation classqiue de PS (a) et le nouveau diagramme proposé (b) : a) Diagramme de condensation classique de PS avec un espace d'état X qui ne change pas. b) Nouveau diagramme de condensation avec l'introduction de la fonction $\bar{u}$ permettant ainsi à chaque instant de générer automatiquement un nouvel espace Q_t en retirant ou rajoutant des sous-espaces de X suivant leurs états de visibilité.

de la figure 4.12, où la personne sort d'une situation d'occlusion et seule sa jambe droite est visible alors que son parent, la cuisse droite, est toujours invisible. Lorsque la jambe est devenue visible, l'algorithme produit une nouvelle stratégie d'exploration en ajoutant la jambe droite dans l'espace à explorer et ainsi génère des hypothèses sur la position de cette jambe en utilisant la configuration mémorisée de la cuisse droite comme point de départ. A cause des contraintes mécaniques entre les différentes parties du corps, il est impossible à l'algorithme de placer la jambe droite correctement puisque la configuration mémorisée de la cuisse droite ne le permet pas. Une solution simple à ce problème consiste à retirer une partie du corps de Q_t si son parent dans l'arbre cinématique n'est pas visible. Ainsi dans l'exemple ci-dessus, la jambe droite ne sera pas ajoutée dans l'espace à explorer puisque son parent n'est pas encore visible. Pour que l'algorithme puisse prendre en compte cette situation, nous définissons la fonction $\bar{Pa}$ qui prend en paramètre χ_{t-1}^j et renvoie son parent dans l'arbre cinématique :

$$\bar{Pa}(\chi_t^j) = \chi_t^p,$$

avec $1 \leq p \leq K$. Nous définissons aussi une nouvelle fonction $\bar{h}$ permettant de prendre en compte la visibilité d'une partie du corps et de son parent dans l'arbre cinématique :

$$\bar{h}(\chi_t^j) = \min(\bar{u}(\chi_t^j), \bar{u}(\bar{Pa}(\chi_t^j))).$$

Cette fonction renvoie 0 si au moins χ_t^j ou son parent $\bar{Pa}(\chi_t^j)$ est invisible.

Nous définissons à nouveau l'ensemble Q_t égale à :

$$Q_t = \{X^j / \bar{h}(\chi_{t-1}^j) = 1\}^{j:1\cdots K},$$

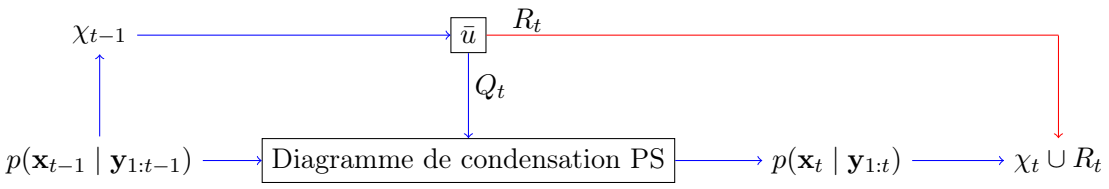


FIGURE 4.11 – Nouveau diagramme de condensation avec mémoire.

et R_t égale à :

$$R_t = \{\chi_{t-1}^j / \bar{h}(\chi_{t-1}^j) = 0\}^{j:1\cdots K}.$$

De cette façon, même si une partie du corps est visible elle ne sera pas incluse dans Q_t tant que son parent ne l'est pas.

4.5.3 Identification de l'état de visibilité des parties du corps

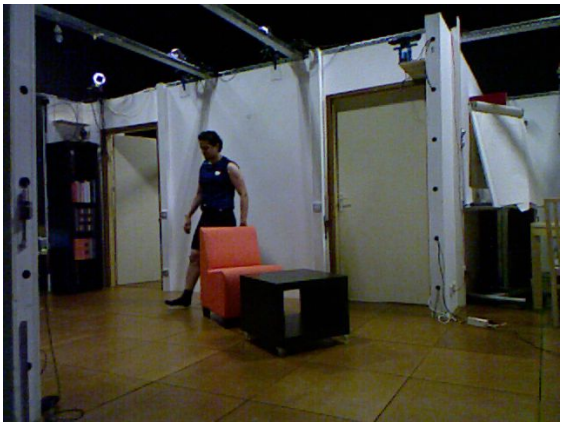
Dans la suite nous décrivons la méthode que nous avons développée pour la détection de l'état de visibilité des différentes parties du corps. Cette méthode représente la fonction $\bar{u}$ que nous avons introduite dans la section précédente. L'idée de la méthode est d'utiliser la meilleure configuration χ_t retrouvée à l'instant t pour prédire à l'instant $t+1$ les parties du corps invisibles.

Rappelons le fonctionnement du système : lorsqu'une image de profondeur est reçue à l'instant t, la grille d'occupation est mis à jour avec le HMM qui identifie si chaque cellule de la grille est occupée par un objet fixe de la scène, mobile ou non occupée. Ensuite un algorithme d'étiquetage est appliqué sur les cellules mobiles pour les rassembler en étiquettes distinctes. Les cellules de chaque étiquette sont ensuite projetées sur l'image de profondeur pour extraire la silhouette de profondeur de la personne qui sera utilisée pour construire la fonction de vraisemblance pour l'algorithme de filtrage particulaire. Finalement, l'algorithme de filtrage renvoie la meilleure configuration retrouvée χ_t du modèle 3D.

Le principe de la méthode pour identifier si une partie du corps est visible ou non est le suivant : A l'instant $t+1$, lorsqu'une nouvelle image est reçue, la grille d'occupation est mise à jour, ainsi que les positions des étiquettes. A ce stade, nous avons un nouvel ensemble d'étiquettes qui correspond à la nouvelle position 3D des personnes dans la scène. La position 3D de χ_t trouvée à l'instant t est déplacée vers le centre de masse de son étiquette correspondante. Ensuite, les cylindres constituant la meilleure configuration retrouvée χ_t sont approximés par un ensemble de sphères 3D comme le montre la figure 4.13. Les sphères associées à chaque cylindre sont projetées dans la grille d'occupation pour déterminer si chaque sphère se situe dans une cellule visible ou non. Si la moitié des sphères se projettent dans des cellules invisibles, la partie du corps est considérée invisible.

Dans la suite, nous décrivons d'abord l'algorithme développé pour identifier si une cellule de la grille est visible. Ensuite nous présentons l'algorithme pour approximer chaque cylindre par un ensemble de sphères. Finalement la méthode pour identifier si une partie du corps est visible ou non sera présentée.

Visibilité de chaque cellule de la grille d'occupation Soit c_i le centre d'une cellule de la grille d'occupation de coordonnées (X_i, Y_i, Z_i) avec $i : 1 \cdots M$ où M est le nombre des cellules dans la grille. Pour déterminer si la cellule est visible ou non, son centre est projeté dans l'image



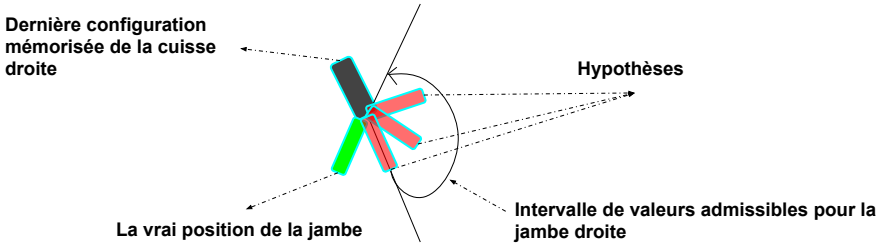


FIGURE 4.12 – Un exemple d'une personne en-train de sortir d'une situation d'occlusion où seule la jambe droite est visible alors que son parent qui est la cuisse droite est toujours invisible. L'algorithme est incapable de trouver correctement la position de la jambe droite (en vert) puisque la dernière configuration sauvegardée de la cuisse droite ne le permet pas puisque l'intervalle des valeurs admissibles de la jambe dépend de la position de la cuisse. Dans ce cas, il est inutile de chercher la position de la jambe si son parent (la cuisse) est encore invisible.

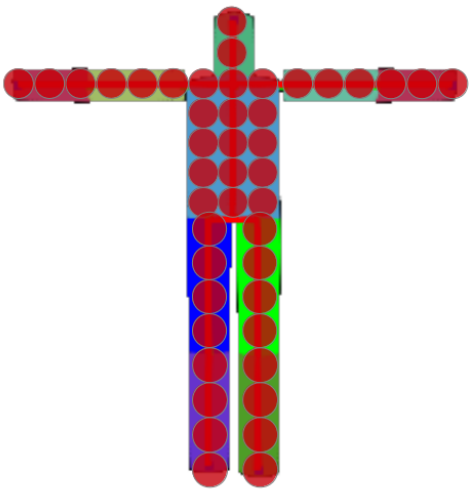


FIGURE 4.13 – L'approximation des cylindres par un ensemble de sphères.

de profondeur I_R en utilisant les équations de projection perspectives :

$$u_i = \frac{X_i \times f_x}{Z_i} + c_x,$$
$$v_i = \frac{Y_i \times f_y}{Z_i} + c_y.$$

Ensuite, la distance $\epsilon_i = I_R(u_i, v_i) - Z_i$ est calculée, avec $I_R(u_i, v_i)$ est la profondeur du pixel u_i, v_i qui correspond à la projection du centre d'une cellule dans l'image de profondeur reçue de la caméra. Si ϵ_i est inférieur à un certain seuil, la cellule est considérée invisible. En pratique, nous n'avons pas besoin de parcourir à nouveau toute la grille, puisqu'au moment de la mise à jour de la grille par le HMM, nous pouvons déterminer l'état de visibilité de chaque cellule.

Approximation d'un cylindre par un ensemble de sphères Les étapes pour approximer chaque cylindre par un ensemble de sphères sont décrites dans l'algorithme 6. En effet, Soit h et r la longueur et le rayon d'un cylindre. Le nombre des sphères pour approximer le cylindre est proportionnel à la hauteur du cylindre et il est calculé comme indiqué dans la ligne 2 de l'algorithme 6 et le pas d'échantillonnage est égal à l'inverse du nombre des sphères (ligne 4). Pour le cas particulier du torse, pour ne pas obtenir de grosses sphères, le cylindre du torse est d'abord échantillonné en un sous-ensemble de cylindres proportionnels au diamètre du cylindre d'origine. Ensuite, pour chaque sous cylindre l'algorithme 6 est appliqué.

Finalement, les sphères obtenues pour chaque cylindre d'une configuration χ_t^j sont transformées par la matrice décrivant la position et l'orientation dans l'espace de cette configuration.

Visibilité d'une partie du corps Maintenant que chaque cylindre est approximé par un ensemble de sphères. Pour identifier si une partie du corps est visible ou non, chaque sphère est projetée dans la grille d'occupation pour voir si elle se situe dans une cellule visible ou non. Si la moitié des sphères se projettent dans des cellules invisibles, la partie du corps est considérée invisible. Pour savoir dans quelle cellule une sphère se situe, nous avons utilisé une technique simple et rapide qui consiste à diviser les coordonnées du centre de la sphère par la résolution

de la grille et ensuite de prendre la partie entière résultante de la division comme index pour trouver la cellule la plus proche correspondante.

Algorithm 6: Approximation d'un cylindre par un ensemble de sphères

Entrée: Longueur d'un cylindre (h) et son rayon (r)
Sortie : Ensemble de sphères

```
/* nombre des sphères*/
nombreDeSphères = 0.5 × h/r
/* pas d'échantillonnage*/
pas = 1/nombreDePas
/* génération des sphères*/
for i = 0 to nombreDePas do
    alpha = i * pas
    Sphères[i].centre = Vecteur3D(PointBas.x, PointBas.y + alpha * h, PointBas.z )
    Sphères[i].rayon = r
```

4.5.4 Recuit-simulé

La fonction de distance possède des minima locaux comme nous l'avons démontré dans la section 4.2. Pour aider les particules à les éviter, la méthode de recuit-simulé (APF) décrite dans la section 2.5.1 permet aux particules de se diriger graduellement vers le maximum global de la fonction de vraisemblance en évitant les maxima locaux. Cette méthode lisse sur plusieurs niveaux (ou couches) la fonction de vraisemblance originale et modifie l'étape de propagation du filtrage particulaire classique. Pour combiner le recuit simulé avec notre approche, reprenons la fonction de vraisemblance $p(\mathbf{y_t} \mid \mathbf{x_t})$ de l'équation 4.2 que nous la notons par la suite w :

$$w = p(\mathbf{y_t} \mid \mathbf{x_t}) = \exp(-\sigma).$$

Soit M le nombre de couches de recuit-simulé. L'algorithme commence de la couche M jusqu'à atteindre la première couche. Pour un niveau m (avec $1 \leq m \leq M$), la fonction de vraisemblance est lissée à l'aide d'un facteur β_m. Ainsi à chaque niveau m, la nouvelle fonction de vraisemblance est égale à :

$$w^{(m)} = w^{\beta_m} = \exp(-\beta_m \cdot \sigma),$$

avec $1 = \beta_0 > \beta_1 > \cdots > \beta_m$. Ensuite, l'ensemble des N particules $S(t)^{(m)} = \{{x_t}^{(i),(m)}, {w_t}^{(i),(m)}\}^{i:1..N}$, à l'instant t est itérativement ré-échantillonné et propagé M fois pour chaque partie du corps X^j suivant la nouvelle dynamique :

$${x_t}^{(i),(j),(m-1)} = {x_t}^{(i),(j),(m)} + B^{(j),(m)},$$

Avec $B^{(j),(m)}$ une loi normale multidimensionnelle avec une variance P_m^j et une moyenne de zéro. P_m^j pour chaque couche est obtenue de la façon suivante :

$$P_m^j = P_0^j \cdot 0.5^{(M-m)},$$

avec P_0^j la variance pour chaque articulation qui est une constante bio-mécanique liée à la vitesse de déplacement de l'articulation. Avoir une variance P_m^j qui se décroit d'une couche à l'autre permet aux particules dans les premières couches de ne pas se focaliser sur les caractéristiques

locales de la fonction de vraisemblance. Par exemple, pour le premier niveau $m = M$, nous avons $P_m^j = P_0^j$, les particules ainsi possèdent le plus grand intervalle de diffusion et au fur et à mesure que m décroit, P_m^j décroit aussi (n'oublions pas qu'en même temps $w^{(m)}$ devient moins lissée), et par conséquent, l'intervalle de diffusion sera de plus en plus réduit, laissant aux particules la possibilité de se focaliser sur les caractéristiques locales dans les dernières couches (un schéma illustratif est présent sur la figure 2.2).

4.5.5 Ré-échantillonnage

Il nous reste à définir la fonction utilisée pour le ré-échantillonnage. Rappelons que le ré-échantillonnage dans un filtre particulaire assure la convergence du filtre en régularisant les poids des particules à chaque itération. Comme nous l'avons décrit dans la section 2.4, le phénomène de dégénérescence des poids des particules au cours du temps est inévitable dans le filtrage particulaire. En effet, après un certain temps, très peu de particules possèdent des poids élevés et contribuent à l'approximation de la densité *a posteriori* alors que les autres auront des poids très faibles, ainsi la capacité d'exploration du filtre est perdue. Le ré-échantillonnage a pour but d'éviter ce phénomène de dégénérescence. L'objectif de l'échantillonnage est de générer un nouveau système de N particules non pondérées à partir du même ensemble de N de particules pondérées en dupliquant celles qui ont un poids élevé. Nous utilisons, dans notre méthode, le ré-échantillonnage multinomial décrit dans l'annexe A.

L'algorithme 7 fait une synthèse de l'ensemble du processus : à chaque instant t, l'algorithme reçoit un ensemble des particules $S_{t-1} = \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}^{i:1\cdots N}$, la meilleure configuration χ_{t-1} obtenue à $t-1$, un espace d'état X et une observation $\mathbf{y}_t$. La première étape de l'algorithme consiste à identifier si chaque configuration χ_{t-1}^j est visible ou non en utilisant la fonction $\bar{h}$ dont l'implémentation a été décrite dans la section 4.5.3. Les ensembles Q_t, contenant les parties du corps visibles, et R_t, contenant les configurations des parties du corps invisibles sont construits. Ensuite l'algorithme parcourt chaque partie du corps visible dans Q_t et applique une recherche multi-couches pour trouver la meilleure configuration χ_t^j. A la fin, l'ensemble des configurations des parties du corps invisibles sont fusionnées dans χ_t.

4.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode pour suivre le mouvement d'une personne dans un environnement encombré avec occlusions. D'abord, nous avons décrit le problème causé par les occlusions présentes dans la scène qui rend le suivi difficile puisque la silhouette est partiellement observable. Nous avons ensuite montré que les approches classiques par filtrage particulaire sont incapables de suivre le mouvement dans ces situations. C'est pour cela qu'il est nécessaire de les adapter pour les rendre robustes aux occlusions. Nous avons ainsi proposé une nouvelle méthode de suivi de mouvement qui modifie l'algorithme de filtre particulaire factorisé (PS) pour le rendre robuste aux occlusions. Cette modification permet de produire à chaque instant une nouvelle stratégie d'exploration en retirant ou rajoutant des parties du corps suivant leur état de visibilité. Nous avons aussi développé une nouvelle méthode qui utilise les connaissances *a priori* du système pour identifier si chaque partie du corps est visible ou non à l'instant t. Cette méthode consiste à approximer les cylindres des différentes parties du corps par un ensemble de sphères et ensuite de projeter chaque sphère dans la grille d'occupation pour tester si elle se situe dans une cellule visible ou non. Finalement, nous avons vu que notre fonction de vraisemblance est multi-modale suite à une expérience que nous avons réalisée. C'est pour

Algorithm 7: Algorithme de filtrage particulaire pour le suivi du mouvement avec gestion des occlusions.

entrée: $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}^{i:1\cdots N}$, χ_{t-1}, X, $\mathbf{y}_t$
sortie : $\{x_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N}$

/* identifier quelle partie du corps est visible en utilisant la fonction $\bar{h}$ que nous avons développé dans la section 4.5.3*/
pour $\chi_{t-1}^j \in \chi_{t-1}$ **faire**
  si $\bar{h}(\chi_{t-1}^j) = 1$ **alors**
    $Q_t = Q_t \cup X^j$
  sinon
    $R_t = R_t \cup \chi_{t-1}^j$
/* pour chaque partie du corps visible*/
pour $X^j \in Q_t$ **faire**
  /** initialisation pour la première couche M*/
  pour $i : 1 \cdots N$ **faire**
    - Propagation ${x_t}^{(i),(j),(M)} = {x_t}^{(i),(j),(0)} + B^{(j),(M)}$
    - Correction $w_t^{(i),(j),(M)} = (w_t^{(i)})^{\beta_M}$.
  /* appliquer une recherche multi-couche */
  pour $m : M \cdots 1$ **faire**
    - Calculer $P_m^j = P_0^j \cdot 0.5^{(M-m)}$
    - Calculer la somme totale des poids : $S = \sum_{i=1}^{N} w_t^{(i),(m)}$
    pour $i : 1 \cdots N$ **faire**
      - Normalisation des poids : $w_t^{(i),(m)} = \frac{w_t^{(i),(m)}}{S}$
    - Ré-échantillonnage :
    $\{{x_t}^{(i),(m)}, w_t^{(i),(m)} = \frac{1}{N}\} = Echantillonner(\{{x_t}^{(i),(m)}, w_t^{(i),(m)}\})$
    pour $i : 1 \cdots N$ **faire**
      - Propagation ${x_t}^{(i),(j),(m-1)} = {x_t}^{(i),(j),(m)} + B^{(j),(m)}$
      - Correction $w_t^{(i),(m-1)} = (w_t^{(i)})^{\beta_{m-1}}$.
  - Calculer la meilleure configuration de la partie du corps j avec les poids normalisés :
  $\chi_t^j = \sum_{i=1}^{N} {x_t}^{(i),(j),(0)} \cdot w_t^{(i),(j),(0)}$
  - Mettre à jour la meilleure configuration globale $\chi_t \leftarrow \chi_t^j$
- Mémoriser les dernières configurations des parties du corps cachées :
$\chi_t = \chi_t \cup R_t$

cela que nous avons introduit dans notre approche finale la technique de recuit-simulé connue pour être robuste aux maxima locaux .

Dans le chapitre suivant, nous allons dans un premier temps évaluer les performances de notre approche d'une façon quantitative et qualitative sur un ensemble de données d'un *Benchmark* que nous avons développé et mis en ligne sur le lien suivant [motion capture dataset, 2015]. Ce *Benchmark* est composé de plusieurs séquences d'une personne se déplaçant dans une scène avec des occlusions. Dans un deuxième temps nous évaluons la méthode d'extraction de la silhouette décrite dans le chapitre 3.

Chapitre 5

Evaluation

Sommaire

5.1 Construction d'une base de données de capture de mouvement avec vérité terrain 80
5.1.1 Principe de fonctionnement du système Qualisys 80
5.1.2 Calibration de la Kinect par rapport au système Qualisys 80
5.2 Comparaison avec les algorithmes APF et PS 82
5.2.1 Choix des paramètres 83
5.2.2 Initialisation 83
5.2.3 Critères d'évaluation 84
5.2.4 Erreur moyenne et écart type 85
5.2.5 Pourcentage d'erreur 86
5.2.6 Comparaison Qualitative 88
5.2.7 Identification des parties cachées du corps 89
5.3 Evaluation de la méthode d'extraction de la silhouette 90
5.3.1 Simulation 90
5.3.2 Etat initial de la grille d'occupation 93
5.3.3 Influence de la résolution de la grille d'occupation sur la silhouette 93
5.3.4 Adaptation aux changements da la scène 93
5.4 Vitesse d'exécution et implémentation sur carte graphique . . . 94
5.5 Conclusion 95

Dans ce chapitre, nous présentons le *benchmark* que nous avons construit et mis en ligne sur le lien suivant [motion capture dataset, 2015]. Ce *benchmark* est composé d'un ensemble de séquences vidéo enregistrées avec une caméra RGB-D filmant une personne se déplaçant dans une scène avec occlusions. La vérité terrain est obtenue en utilisant un système de capture de mouvement à l'aide de marqueurs. Dans le but d'évaluer de combien notre méthode améliore le suivi dans un environnement avec occlusions, nous comparons notre approche à PS et APF en utilisant ce *benchmark*, et nous montrons que notre méthode possède des meilleures performances que ces deux méthodes et produit un suivi stable et robuste contre les occlusions. Nous montrons aussi la capacité de notre méthode à identifier et à retirer les parties invisibles du corps du processus d'estimation du mouvement.

Dans la section 5.3 de ce chapitre, nous évaluons la méthode d'extraction de la silhouette, développée dans le chapitre 3, qui utilise une grille d'occupation couplée à un HMM pour la classification des cellules de la grille. Nous utilisons d'abord un simulateur 3D pour montrer la

capacité du HMM à classifier correctement les cellules. Nous étudions ensuite l'influence de la résolution de la grille d'occupation sur la qualité de la silhouette extraite. L'objectif de cette étude et de montrer que la méthode développée permet de reconstruire la silhouette de la personne à la résolution native du capteur tout en travaillant avec une faible résolution de la grille. Finalement, nous montrons la capacité de la méthode à apprendre en continu le fond dynamique de la scène.

A la fin de ce chapitre, nous discutons de la vitesse d'exécution du système complet (extraction de la silhouette et suivi du mouvement).

5.1 Construction d'une base de données de capture de mouvement avec vérité terrain

Dans le but d'évaluer les performances de notre approche dans des conditions réelles, il nous faut une base de données avec vérité terrain d'une personne se déplaçant dans une scène dynamique avec des obstacles. Après avoir effectué une recherche sur les bases existantes, nous n'avons pas trouvé de base qui réponde à nos besoins. En effet, toutes les bases que nous avons trouvées ont été réalisées dans des scènes sans occlusions. C'est principalement pour cette raison que nous avons décidé de construire notre propre base de référence en utilisant le système Qualisys de capture de mouvement présent dans l'appartement expérimental au LORIA, lequel il est composé de 8 caméras infra-rouge. L'emplacement de ces caméras et de la caméra Kinect sont indiquées sur la figure 5.1.

5.1.1 Principe de fonctionnement du système Qualisys

Avant de présenter les étapes de construction de la base, nous décrivons brièvement le fonctionnement du système Qualisys basé sur des marqueurs lumino-réfléchissants. D'abord, des petites boules réfléchissantes sont fixées sur l'objet dont nous souhaitons suivre le mouvement en 3D (dans notre cas, ce sera sur les différentes articulations de la personne). Chaque caméra infra-rouge émet un signal qui est réfléchi par ces boules vers le récepteur de chaque caméra. Ensuite, par des techniques de vision, la localisation 3D des boules réfléchissantes est déterminée. C'est dans ce but que le système a besoin de plusieurs caméras. Le système Qualisys fonctionne à 300 images par seconde et avec une précision de l'ordre du millimètre, ce qui va nous permettre d'obtenir avec une haute précision la position 3D des articulations de la personne dont on va se servir comme vérité terrain pour évaluer notre approche.

5.1.2 Calibration de la Kinect par rapport au système Qualisys

La procédure de calibration de la caméra Kinect par rapport au système Qualisys pour obtenir un système de coordonnées global est la suivante : Nous avons fixé 4 boules réfléchissantes sur les coins d'une boîte de dimension 20×30 cm. Le système Qualisys renvoie les coordonnées de ces marqueurs dans son repère. Pour trouver les coordonnées de ces points dans le repère caméra, nous avons développé une interface graphique pour sélectionner manuellement ces 4 points dans l'image de profondeur et reconstruire les points 3D correspondants dans le repère caméra en

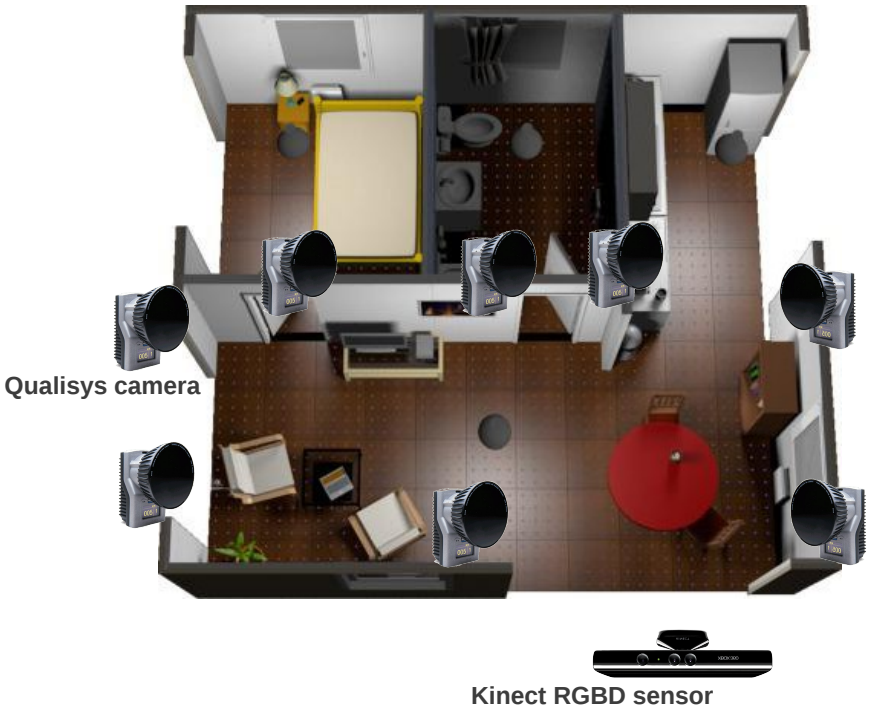


FIGURE 5.1 – L'emplacement des 8 caméras du système Qualisys et de la caméra Kinect dans l'appartement expérimental au LORIA.

utilisant les équations de re-projection suivantes :

$$X = \frac{u - c_x}{f_x} \times d,$$
$$Y = \frac{v - c_y}{f_y} \times d,$$
$$Z = d,$$

avec (u, v) les coordonnées du pixel dans le plan image ; f_x, f_y, c_x, c_y les paramètres intrinsèques de la caméra obtenues par une procédure de calibration et d la profondeur du pixel obtenue à partir de l'image de profondeur. La boîte est positionnée à plusieurs endroits différents dans la scène et à chaque fois nous sauvegardons les coordonnées des points 3D dans les deux repères. La transformation (rotation et translation) qui permet de passer d'un repère à l'autre est obtenue en minimisant la fonction quadratique suivante :

$$E(R, T) = \sum_{i=1}^{n} \|R \times p_i + T - m_i\|^2,$$

avec R et T la rotation et la translation entre les deux repères, n le nombre de points 3D appariés. p_i et m_i les coordonnées d'un point 3D dans le repère caméra et du système Qualisys respectivement. Nous avons utilisé la fonction *estimateAffine3D* de la bibliothèque de OpenCV pour trouver R et T.

Après avoir calibré et synchronisé la caméra Kinect avec le système Qualisys, nous avons enregistré 12 séquences vidéo de 30 secondes chacune, d'une personne se déplaçant devant une caméra dans différentes scènes meublées contenant des endroits occultés. La vitesse d'acquisition de la Kinect 1 est 30 Hz. La figure 5.2 montre quelques captures d'écrans de la base construite. Cette base est mise à disposition pour la communauté sur le lien suivant [motion capture dataset, 2015]. Un code source en C++ est également disponible à cette même adresse permettant de rejouer les séquences de la base avec la position des marqueurs projetés sur les images. Ce programme a pour but de faciliter le travail des personnes ou chercheurs souhaitant utiliser notre base de données pour évaluer et comparer leurs approches. La figure 5.3 montre la sortie du programme C++ fourni avec la base. Notons que nous avons été contacté par des collègues qui travaillent sur la capture du mouvement et qui sont intéressés par notre base.

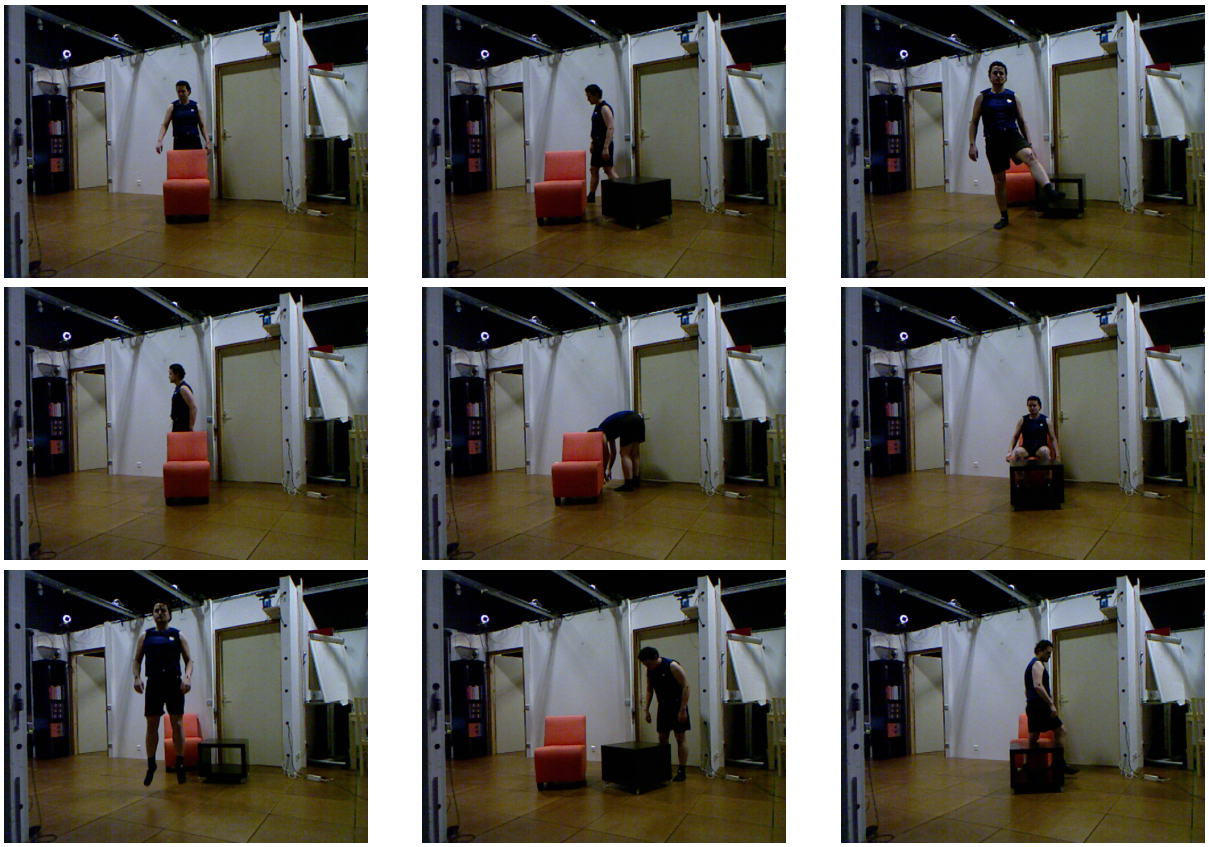


FIGURE 5.2 – Quelques captures d'écran du *Benchmark* que nous avons construit.

5.2 Comparaison avec les algorithmes APF et PS

Nous avons comparé notre méthode avec l'approche par recuit-simulé décrite dans la section 2.5.1 qui sera notée dans la suite APF, et avec l'approche factorisée décrite dans la section 2.5.2 qui sera notée par PS dans la suite. L'objectif de cette comparaison est de valider que notre approche améliore les performances de suivi dans des scènes en présence des occlusions.

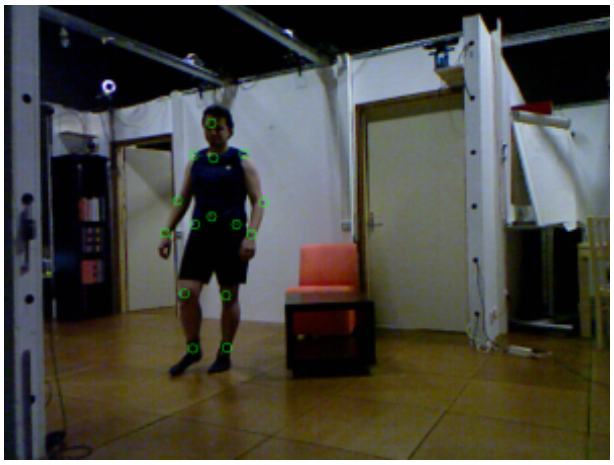


FIGURE 5.3 – Capture d'écran du programme fourni avec la base de données de capture de mouvement permettant d'afficher les marqueurs (cercles en vert) sur les images.

5.2.1 Choix des paramètres

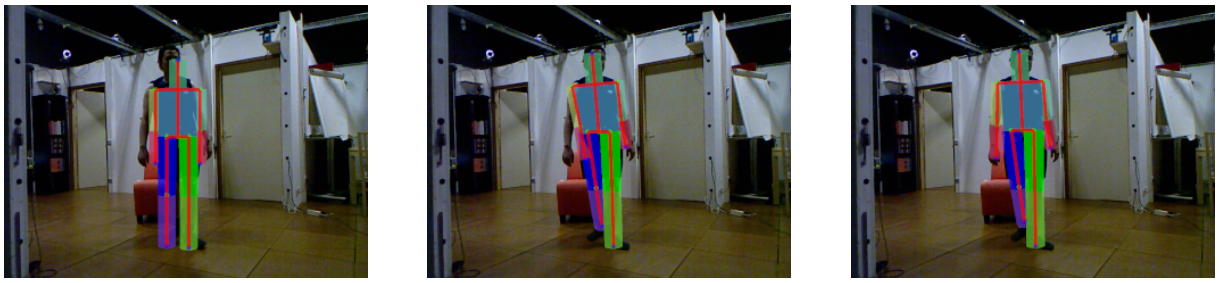
Pour la grille d'occupation, nous avons choisi une grille de taille $5 \times 5 \times 2.5$ mètres ($2.5m$ étant sur l'axe vertical) et de résolution de 5 cm. Pour le HMM, nous avons fixé les probabilités de la matrice de transition comme suit (voir section 3.3) :

$$\alpha = 0.01, \beta = 0.1, \gamma = 0.4, \psi = e^{-38}.$$

Les trois méthodes utilisent la silhouette de profondeur renvoyée par la méthode d'extraction de la silhouette que nous avons développée (voir chapitre 3). Pour le choix du nombre des particules pour chaque méthode, ce choix est fait d'une façon précise sur une séquence de la base ne contenant pas d'obstacles (séquence 12). En effet, nous avons lancé plusieurs fois chaque méthode sur cette séquence avec des paramètres différents. Et finalement pour chacune des méthodes, nous avons retenu le nombre le plus petit de particules ainsi que de couches dans le recuit-simulé qui mènent à une faible erreur moyenne. De cette manière, pour notre filtre particulaire, qui utilise une approche combinée entre APF et PS, 200 particules avec 5 niveaux de recuit-simulé sont utilisées. Pour la méthode APF, nous avons utilisé 200 particules avec 10 niveaux de recuit-simulé. La méthode PS est testé avec 2000 particules.

5.2.2 Initialisation

Pour l'initialisation des filtres particulaires à $t = 0$, la position 3D du centre de la personne, renvoyée par l'algorithme d'étiquetage (voir chapitre 3), est utilisée comme point de départ. A l'initialisation, nous fixons la configuration de départ du modèle 3D à celle d'une personne en face de la caméra. Par conséquent, la personne doit se positionner d'une manière similaire à celle du modèle 3D, pour permettre aux algorithmes (PS, APF et le notre) de démarrer correctement. Néanmoins, il n'est pas nécessaire d'avoir une correspondance très précise entre la position de départ de la personne et la configuration initiale du modèle 3D, parce que les algorithmes sont capables de converger vers la bonne configuration dés les deux premières images de la séquence, comme le montre la figure 5.4.



(a) image à $t = 0$ (b) $t = 0 + 1$ image (c) $t = 0 + 2$ images

FIGURE 5.4 – Initialisation des filtres particulaires. a) A $t = 0$, le modèle 3D est initialisé avec sa position par défaut et la personne est légèrement penchée. b) et c) Le filtre particulaire s'adapte à la position effective de la personne au bout des deux premières images successives.

5.2.3 Critères d'évaluation

Pour comparer ces méthodes entre elles, nous avons employé les mêmes critères d'évaluation utilisés par les autres bases existantes en ligne, notamment ceux de HumanEva ([Sigal *et al.*, 2010]). Pour chacune des 12 séquences, la performance moyenne et l'écart type sont calculés en utilisant les équations suivantes :

$$\mu = \frac{1}{T}\sum_{t=1}^{T} D(X_t, \hat{X}_t), \tag{5.1}$$

et :

$$\sigma = \sqrt{\frac{1}{T}\sum_{t=1}^{T} D(X_t, \hat{X}_t) - \mu)^2}, \tag{5.2}$$

avec T le nombre d'images dans chaque séquence, $D(X_t, \hat{X}_t)$ la distance moyenne entre la position de référence des marqueurs $X_t = \{x_1, x_2, ...x_M\}_t$ et celle estimée par chaque méthode $\hat{X}_t = \{\hat{x}_1, \hat{x}_2, ...\hat{x}_M\}_t$ égale à :

$$D(X_t, \hat{X}_t) = \sum_{m=1}^{M} \frac{\|x_m - \hat{x}_m\|}{M}, \tag{5.3}$$

avec M le nombre des articulations, $x_m \in \mathbb{R}^3$ la position du marqueur qui correspond à l'articulation m et $\hat{x}_m \in \mathbb{R}^3$ la position de l'articulation estimée par chaque méthode.

En plus de ces deux critères μ et σ, nous avons introduit 3 critères de comparaison supplémentaires qui ont pour but d'évaluer pour une méthode donnée sa qualité de suivi. Ces critères sont : Le pourcentage de temps où l'erreur $D(X_t, \hat{X}_t)$ ne dépasse pas 10, 15 et 20 cm. Plus les valeurs de ces pourcentages sont élevées, plus la qualité de suivi est bonne.

La table 5.1 dans la page 91 montre les résultats obtenus pour chacune des méthodes. La séquence 12 est la seule qui ne contient pas d'obstacles. Dans les séquences 1, 4, 7 et 10, la personne reste dans les zones occultées pendant un temps plus long par comparaison aux autres séquences. Dans la suite nous allons discuté en détails les résultats obtenus.

5.2.4 Erreur moyenne et écart type

Les colonnes 6 et 7 de la table 5.1, montre l'erreur moyenne μ et l'écart type σ obtenus par chaque méthode. Les valeurs en gras correspondent aux meilleures valeurs.

Sur toute les séquences sauf la séquence 3, notre méthode possède l'erreur moyenne la plus basse. Sur la séquence 3, APF possède une erreur moyenne légèrement inférieure à la notre (15.15 contre 16.13 cm). Pour les séquences 1, 4, 7 et 10, notre méthode possède une erreur qui est largement inférieure à celles de APF et PS. Par exemple, sur la séquence 1 du tableau, l'erreur moyenne de notre méthode est de 17.25 cm alors que pour APF elle est égale à 37.4 cm et pour PS elle vaut 24.38 cm. Sur la séquence 10 aussi, notre méthode a une erreur moyenne de 15.28 cm qui est deux fois inférieure à celle de APF (30.30 cm) et PS (35.48 cm).

Pour l'écart type, notre méthode a la plus faible variation par rapport à APF et PS sauf pour les séquences 3, 5 et 6. Sur la séquence 3, c'est APF qui possède l'écart type le plus faible, lequel reste proche de notre méthode, alors que pour les séquences 5 et 6, les trois méthodes possèdent des variations proches. Sur les séquences 1, 4, 7 et 10, l'écart type de notre méthode est largement en dessous de ceux d'APF et PS surtout pour les séquences 1 et 4.

Une autre représentation graphique sous forme d'une barre d'erreur de l'erreur moyenne et de l'écart type de chacune des méthodes sur les séquences 1, 4, 7, 10 est affichée sur la figure 5.5. Pour rappel, ces séquences sont considérées difficiles puisque le sujet reste dans des zones occultées pendant un temps plus long que les autres séquences. Cette figure montre que l'erreur de notre méthode est largement inférieure à celle de APF et PS pour ces séquences.

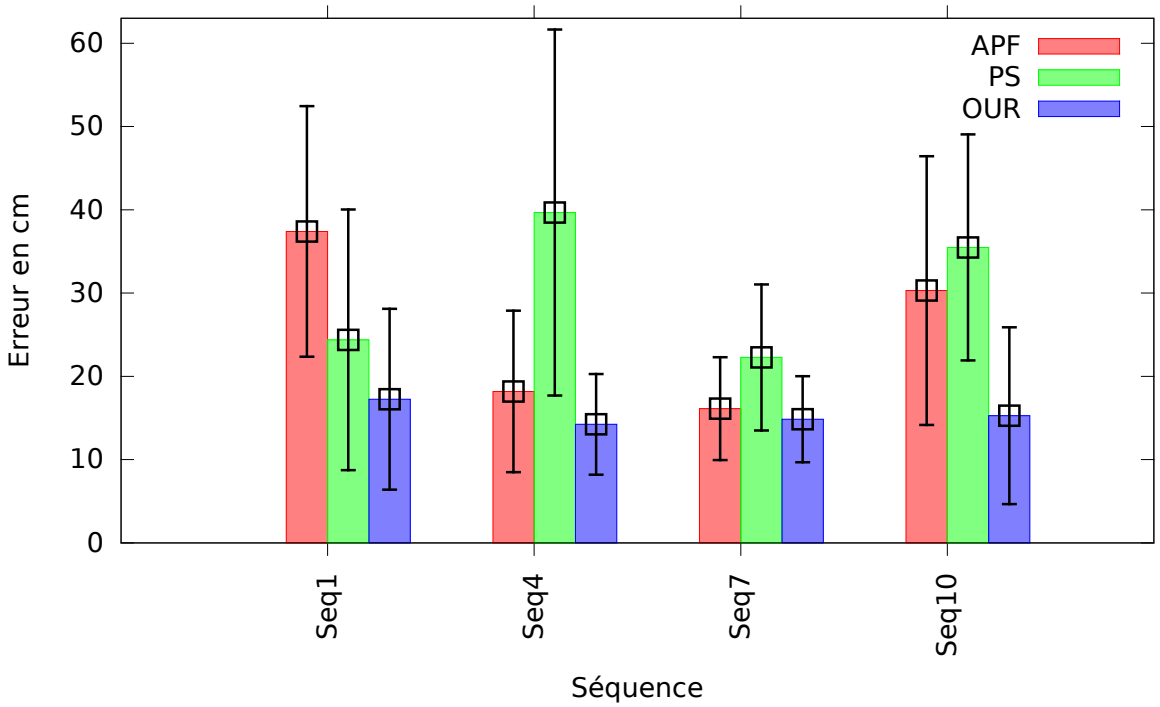


FIGURE 5.5 – Comparaison de l'erreur moyenne et l'écart type sous forme d'une barre d'erreur des trois méthodes sur les séquences 1, 4, 7 et 10.

5.2.5 Pourcentage d'erreur

Dans le but d'évaluer la qualité de suivi de chaque méthode d'une image à l'autre, nous avons calculé le pourcentage de temps où l'erreur $D(X_t, \hat{X}_t)$ ne dépasse pas 10, 15 et 20 cm pour chaque méthode sur toutes les séquences. Ces indicateurs sont liés à la qualité de suivi : plus ces pourcentages sont élevés, meilleur est le suivi.

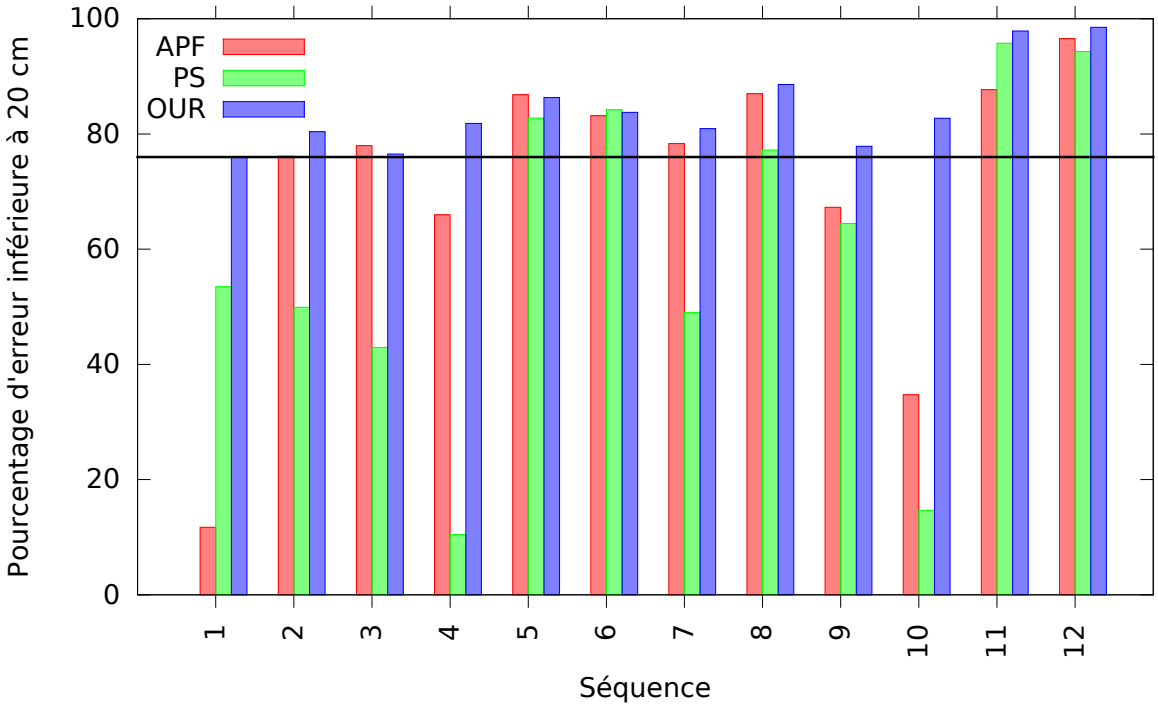


FIGURE 5.6 – Pourcentage de temps où l'erreur ne dépasse pas 20 cm. Plus ce pourcentage est élevée, meilleur est le suivi.

La figure 5.6 montre le pourcentage de temps où l'erreur est inférieure à 20 cm pour les 3 méthodes sur toutes les séquences. Pour notre méthode, ce pourcentage est toujours supérieure à 76% pour toutes les séquences. Pour APF et PS, ce pourcentage descends en dessous du 20 % dans certains cas (séquences 1, 4 et 10). Par exemple, pour la séquence 1, ce pourcentage est égale à 11.72 % pour APF, 53.4 % pour PS alors que pour notre méthode il vaut 76.07 %. Pour la séquence 10, ce pourcentage vaut 34.72 % pour APF, 14.66 % pour PS et pour notre méthode, il est égale à 82.72 %.

Un autre indicateur qui reflète la qualité de suivi est le pourcentage de temps où l'erreur ne dépasse pas 15 cm. Plus ce pourcentage est élevé, meilleur est le suivi. La figure 5.7 représente ce pourcentage pour les trois méthodes. Sur toutes les séquences de la base, notre méthode possède un pourcentage supérieure à 50% et il dépasse les 80% sur certaines séquences (séquences 8, 11 et 12). Pour les séquences 1, 4, 7 et 10, notre méthode possède un pourcentage largement supérieur à celui d'APF et PS. Par exemple sur la séquence 10, le pourcentage est de 22.16 % pour APF et 9.41 % pour PS alors que pour notre méthode il est égale à 69.75 %.

Pour le pourcentage de temps où l'erreur est inférieure à 10 cm, d'après la figure 5.8, notre méthode possède le pourcentage le plus élevé sur toute les séquence. Sur certaines séquences (Séquence 1, 4, 7 et 10), ce pourcentage dépasse largement celui d'APF et PS. Par exemple, pour la séquence 10, le pourcentage obtenue par notre méthode est égale à 39.35 % alors que pour

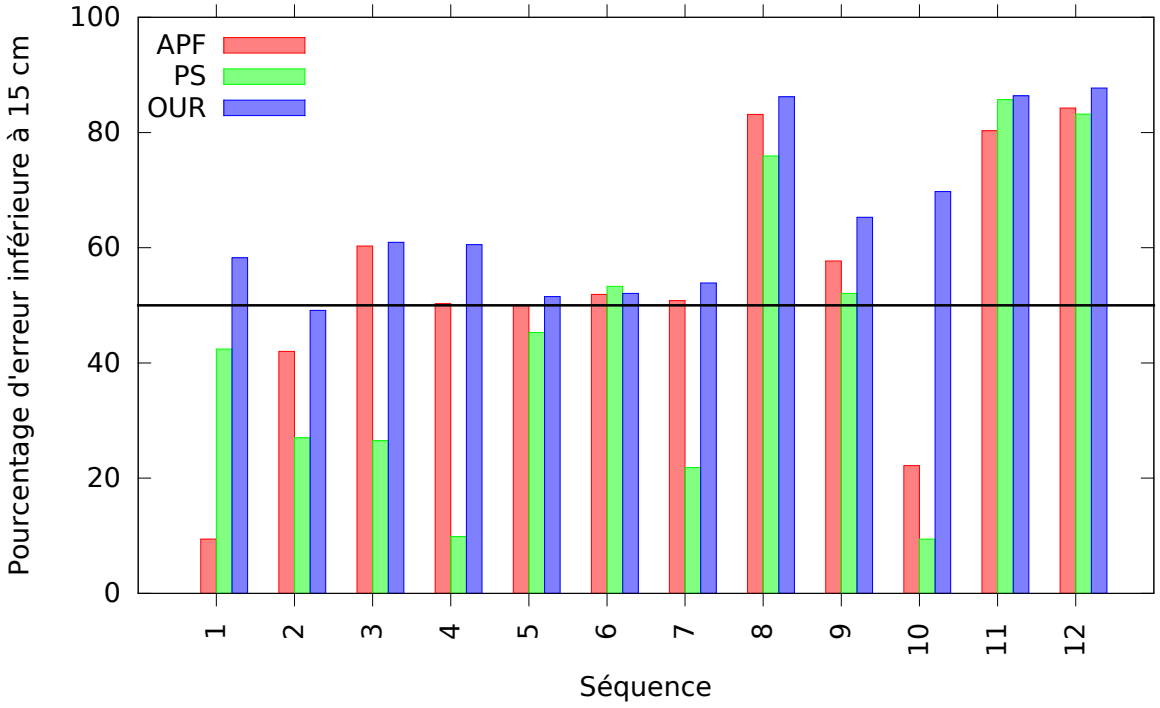


FIGURE 5.7 – Pourcentage de temps où l'erreur ne dépasse pas 15 cm. Plus ce pourcentage est élevé, meilleur est le suivi.

APF il vaut 13.43 % et pour PS il est égale à 3.7 % seulement. Par contre, nous remarquons que ce pourcentage reste bas pour toutes les méthodes (dans le meilleur cas, pour la séquence 11, 52.22% avec notre méthode). Cette basse précision pourrait s'expliquer par le fait que le modèle 3D utilisé pour le suivi comporte des cylindres pour représenter la forme des différentes parties du corps. Cette représentation reste approximative et ne décrit pas parfaitement la forme de la personne. Ainsi, un décalage est toujours présent entre le squelette extrait et la vraie posture de la personne. Une autre raison qui pourrait expliquer ce décalage vient du fait que la position des marqueurs du système Qualisys ne correspond pas exactement à la position des articulations dans le modèle 3D. Effectivement, prenons la figure 5.9 correspondant à la séquence 12 qui ne contient pas d'obstacle. La courbe montre l'erreur obtenue avec les différentes méthodes, et sur cette séquence, toutes réussissent à suivre correctement le mouvement de la personne (ce résultat a été vérifié visuellement). Par contre, nous remarquons que l'erreur n'est jamais nulle et un décalage existe toujours comme il peut être observé sur la figure 5.10. Nous considérons que l'utilisation d'un modèle 3D plus fin pour le suivi pourrait aider à réduire ce décalage.

Une autre conclusion importante que nous pouvons tirer de ces expériences est que la méthode PS donne de mauvaises résultats sur l'ensemble des données. Cette conclusion n'est pas surprenante car l'approche factorisée est connue pour avoir de mauvaises résultats dans les situations d'occlusion, tandis que la méthode APF donne des résultats meilleurs que PS. Et dans des situations où les occlusions sont de faibles durées, nous avons remarqué que APF réussit dans certains cas à reprendre le suivi une fois que la personne est visible à nouveau. Mais quand le sujet reste longtemps dans une zone occultée, APF génère des postures non pertinentes et le suivi est souvent perdu (Cela est bien visible sur les séquences 1, 4, 7 et 10). Une deuxième conclusion à souligner est que, grâce au couplage du recuit-simulé avec PS, et à la méthode de gestion des

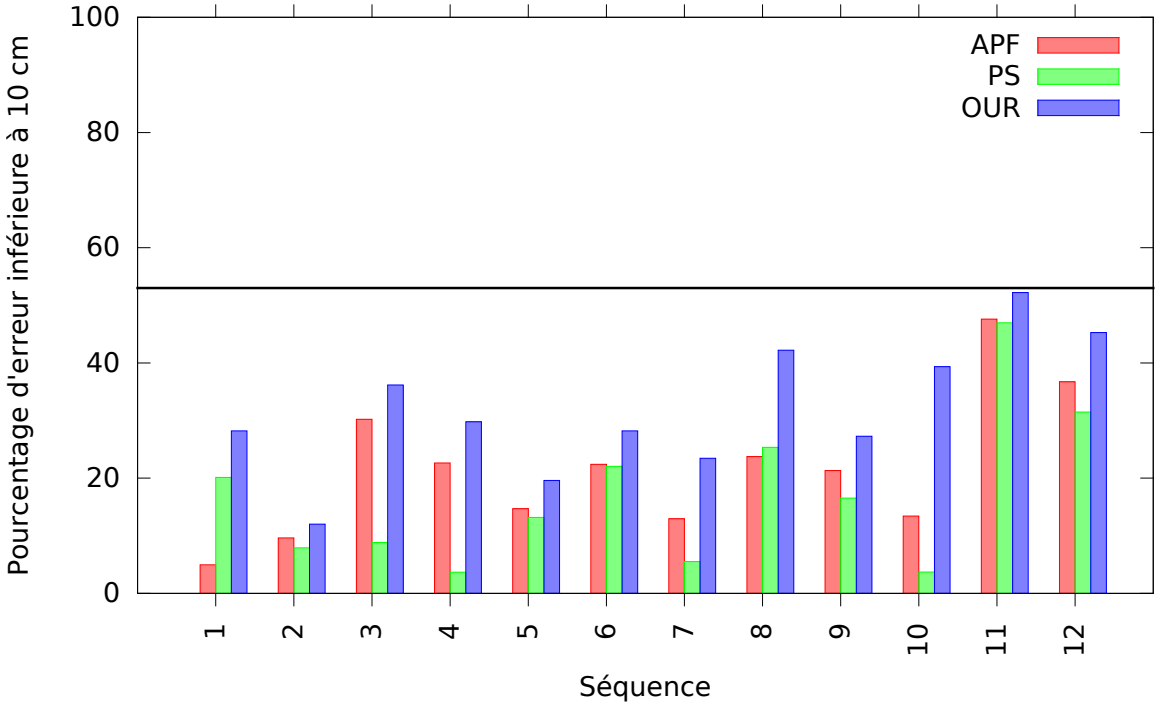


FIGURE 5.8 – Pourcentage de temps où l'erreur ne dépasse pas 10 cm. Plus ce pourcentage est élevé, meilleur est le suivi.

occlusions, notre méthode fournit des meilleurs résultats que APF et PS avec un nombre réduit de particules (notre méthode utilise 200 particules avec 5 couches de recuit-simulé).

Finalement, nous avons observé visuellement le résultat de suivi des trois méthodes sur toutes les séquences pour voir quand le suivi atteint un échec global. Cet échec global est atteint si les jambes sont inversées ou le torse est mal placé, ou si l'orientation du modèle est fausse. Ce résultat visuel est représenté dans le dernière colonne de la table 5.1 (indiqué en rouge). Comme l'indique cette colonne, sur toutes les séquence, notre approche a réussi à suivre le mouvement de la personne sur toutes les séquences. APF a échouée sur les séquences 1, 4, 10. Pour l'approche PS, un échec global est obtenu sur les séquences de 1 à 10. Pour les séquences 11 et 12, les trois approches ont réussit avec une erreur moyenne faible. Cela est expliqué par le fait que la séquence 11, la personne fait des passages très rapides dans des zones occultées laissant le temps à APF et PS de se rattraper lorsqu'elle sort de ces zones. Sur la séquence 12, la scène était vide et ne contenait aucun obstacle.

5.2.6 Comparaison Qualitative

Dans cette partie, nous comparons le comportement de notre méthode avec APF dans des situations d'occlusion pour montrer la capacité de notre méthode à gérer ces situations et à reprendre le suivi lorsque la personne est visible à nouveau.

La figure 5.11 (page 92) montre la courbe d'erreur obtenue par notre méthode et APF sur la séquence 1. Sur cette figure, nous identifions principalement 5 zones (voir les annotations sur la figure). Dans les zones 1, 3 et 5, la personne est visible en entier. Pour les zones 2 et 4, la personne est dans une situation d'occlusion. Nous remarquons que dans les zones 2 et 4, APF

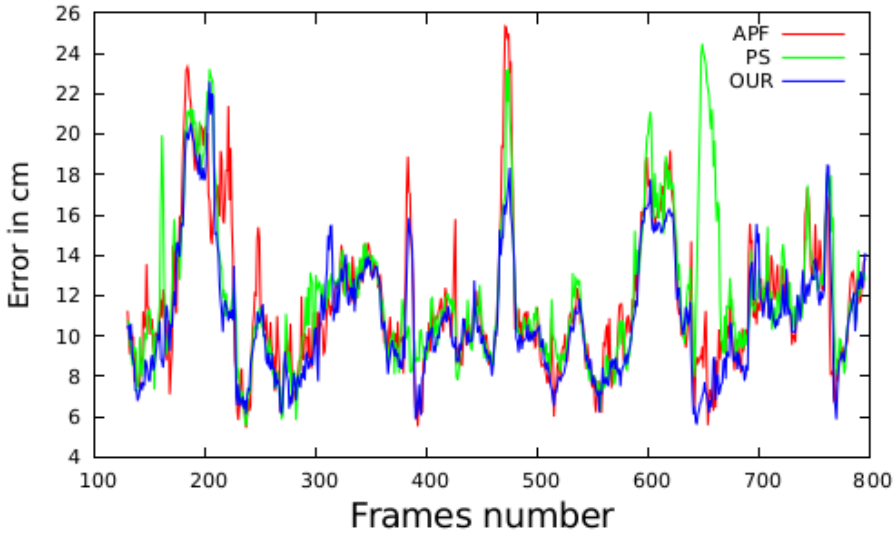


FIGURE 5.9 – Courbe d'erreur au cours du temps, obtenue par notre méthode (OUR), APF et PS sur la séquence 12.

possède une erreur très élevée (70 cm) indiquant une perte de suivi. Ce qui est aussi intéressant à remarquer, est que quand la personne devient visible à nouveau, APF n'est pas capable de reprendre le suivi et l'erreur d'APF reste élevée (zone 3 et 5). Par contre, notre méthode réussit à suivre le mouvement de la personne sur toute la séquence avec une erreur plus faible que celle obtenue par APF. Plus précisément, nous remarquons que lorsque la personne devient visible à nouveau (zone 3 et 5), notre méthode réussit à reprendre le suivi avec une fabile erreur. La raison pour laquelle l'erreur de notre méthode augmente dans les zones 2 et 4 est liée au fait que dans le calcul de l'erreur de l'équation 5.3, la position de la dernière configuration mémorisée des parties invisibles du corps est utilisée.

Sur la figure 5.12 (page 97) est affiché un scénario montrant une personne entrant (figure 5.12(a)) et sortant (figure 5.12(b)) d'une situation d'occlusion. Sur cette figure, nous remarquons que APF échoue complètement lorsque la personne n'est pas visible en entier (figure 5.12(c)). Lorsque la personne sort de la zone occultée (figure 5.12(d)), APF n'est pas capable de reprendre le suivi. Alors que notre méthode réussit à détecter et retirer les parties invisibles du corps (figure 5.12(e)) et reprend le suivi lorsque la personne est à nouveau visible en entier (figure 5.12(f)).

En conclusion, cette expérience montre l'importance de retirer les parties du corps cachées pour résoudre d'une façon efficace le problème des occlusions. En effet, cette étude et une autre que nous avons présentée dans la section 4.4 montrent que les occlusions posent un vrai problème pour les algorithmes de suivi en général, et leur impact n'est pas juste limité à l'instant où elles se produisent. Elles affectent le suivi même après la fin des occlusions. Notre méthode, qui identifie et retire les parties invisibles du corps, produit un suivi stable et robuste et améliore considérablement la qualité de suivi.

5.2.7 Identification des parties cachées du corps

Ici nous montrons, avec des résultats visuels, la capacité de notre méthode à identifier correctement les parties invisibles du corps et à produire une estimation correcte du mouvement de la personne dans différentes types de scènes et en présence d'occlusions. Nous rappelons que notre

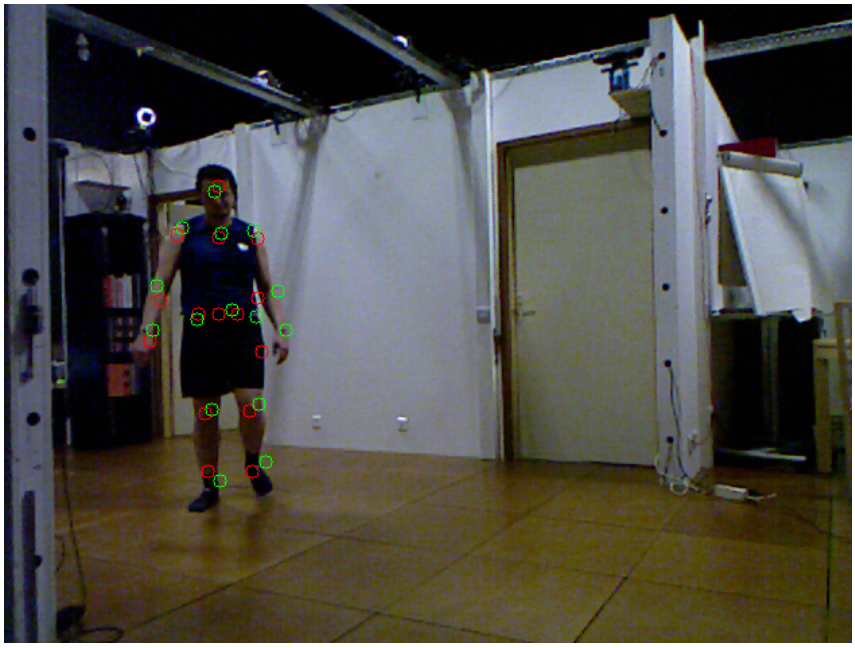


FIGURE 5.10 – Les cercles en vert représentent la position des marqueurs du système Qualisys. Les cercles en rouge représentent la position des différentes articulations du squelette extrait par notre méthode.

méthode de gestion des occlusions approxime chaque cylindre par un ensemble de sphères, qui sont ensuite projetées dans la grille d'occupation afin de tester leur visibilité. Cette méthode, simple à mettre en œuvre et peu coûteuse en temps de calcul, fournit des bons résultats sur l'ensemble des séquences du *benchmark*. Les figures 5.13 et 5.14 (pages 98, 99) montrent que notre méthode est capable d'identifier les parties non visibles du corps, et ensuite les retirer du processus d'estimation du mouvement, suite à différents types d'occlusions.

Ces résultats montrent aussi que notre méthode est capable de suivre le mouvement de la personne dans des environnements variés, et qu'elle ne nécessite aucune connaissance *a priori* sur le type de la scène ou l'emplacement des objets et les obstacles dans l'environnement.

5.3 Evaluation de la méthode d'extraction de la silhouette

Dans cette section nous évaluons la méthode d'extraction de la silhouette qui utilise une grille d'occupation couplée à un HMM pour identifier si chaque cellule est occupée par un objet mobile ou fixe de la scène (voir chapitre 3). Dans un premier temps, nous évaluons avec un simulateur 3D la capacité de la méthode à classifier correctement les cellules d'une scène virtuelle. Ensuite nous étudions la sensibilité du HMM vis-à-vis des différentes valeurs pour la probabilité initiale de chaque état. Nous étudions aussi l'influence de la résolution de la grille d'occupation sur la qualité de la silhouette extraite. Finalement nous montrons la capacité de notre méthode à apprendre en ligne le fond dyanmique de la scène.

5.3.1 Simulation

Pour évaluer la capacité de la méthode à classifier correctement les cellules de la grille d'occupation, il faut comparer la sortie de la grille d'occupation avec une image de référence. Or

Séquence	Erreur Méthode	<10cm [%]	<15cm [%]	<20cm [%]	Err moyenne (μ) [cm]	Ecart type (σ) [cm]	Statut
Seq1	APF	4.95	9.41	11.72	37.40	15.05	ECHEC
	PF	20.13	42.41	53.47	24.38	15.65	ECHEC
	OUR	**28.22**	**58.25**	**76.07**	**17.25**	**10.86**	
Seq2	APF	9.64	42.02	76.15	17.17	6.63	
	PF	7.90	27.01	49.92	25.92	15.86	ECHEC
	OUR	**12.01**	**49.13**	**80.41**	**16.45**	**6.17**	
Seq3	APF	30.23	60.29	**77.97**	**15.15**	**7.66**	
	PS	8.84	26.53	42.93	26.07	13.16	ECHEC
	OUR	**36.17**	**60.93**	76.53	16.13	9.96	
Seq4	APF	22.63	50.29	65.96	18.19	9.70	ECHEC
	PS	3.68	9.86	10.44	39.67	21.97	ECHEC
	OUR	**29.79**	**60.54**	**81.82**	**14.24**	**6.04**	
Seq5	APF	14.70	50.00	**86.82**	14.85	**4.53**	
	PS	13.18	45.27	82.77	15.63	4.96	ECHEC
	OUR	**19.59**	**51.52**	86.32	**14.77**	4.91	
Seq6	APF	22.40	51.91	83.16	15.53	6.54	
	PS	22.05	**53.30**	**84.20**	15.28	**6.11**	ECHEC
	OUR	**28.20**	52.08	83.74	**15.05**	6.48	
Seq7	APF	12.94	50.81	78.32	16.12	6.18	
	PS	5.50	21.84	49.03	22.27	8.77	ECHEC
	OUR	**23.46**	**53.88**	**80.91**	**14.85**	**5.17**	
Seq8	APF	23.76	83.15	87.00	14.43	9.89	
	PS	25.36	75.92	77.21	16.96	11.97	ECHEC
	OUR	**42.22**	**86.20**	**88.60**	**12.20**	**5.88**	
Seq9	APF	21.32	57.69	67.27	21.27	16.48	
	PS	16.53	52.07	64.46	22.87	15.73	ECHEC
	OUR	**27.27**	**65.29**	**77.85**	**17.28**	**12.28**	
Seq10	APF	13.43	22.16	34.72	30.30	16.14	ECHEC
	PS	3.70	9.41	14.66	35.48	13.57	ECHEC
	OUR	**39.35**	**69.75**	**82.72**	**15.28**	**10.61**	
Seq11	APF	47.62	80.30	87.68	12.57	6.96	
	PS	46.96	85.71	95.73	11.15	3.96	
	OUR	**52.22**	**86.37**	**97.87**	**10.85**	**3.68**	
Seq12	APF	36.73	84.26	96.55	11.61	3.56	
	PS	31.48	83.21	94.30	12.01	3.63	
	OURS	**45.28**	**87.71**	**98.50**	**10.87**	**3.14**	

TABLE 5.1 – Performances de notre méthode (OUR), APF et PS sur le *benchmark* que nous avons construit.

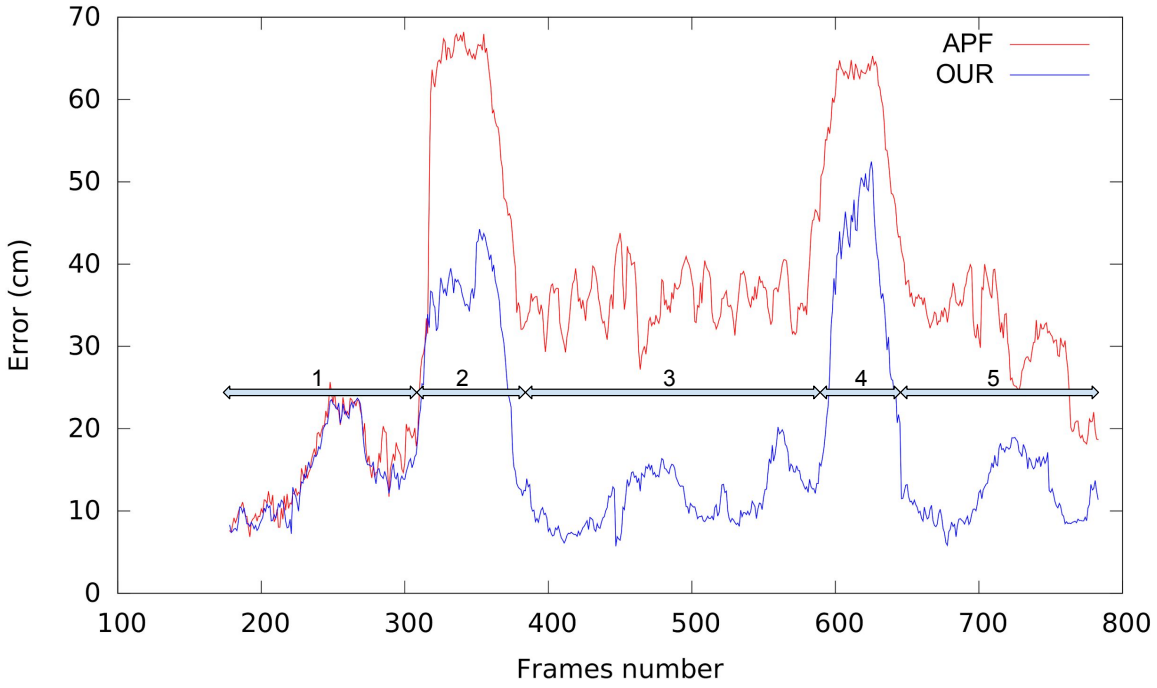


FIGURE 5.11 – Courbe d'erreur obtenue par notre méthode et par APF lorsqu'une situation d'occlusion est rencontrée (séquence 1). Dans les zone 1, 3 et 5, la personne est visible en entier. Dans les zones 2 et 4, la personne est dans une situation d'occlusion.

		Référence	
		Pixels mobiles	Pixels statiques
Détecté	Pixels mobiles	1 294 005	1 068 236
	Pixels statiques	190 958	71 278 387

TABLE 5.2 – Tableau de contingence

l'obtention de cette référence est difficile puisqu'elle nécessite d'étiqueter une image réelle pour identifier si chaque pixel appartient à un objet mobile ou fixe de la scène. C'est pour cela que nous avons utilisé un simulateur 3D que nous avons développé pour l'évaluation. Avec ce simulateur, nous avons enregistré l'activité d'un modèle 3D articulé qui se déplace dans une scène simulant l'appartement expérimental du LORIA. Le simulateur génère une image de référence qui indique si chaque pixel rendu de la scène appartient à un objet fixe ou mobile. En même temps, une caméra virtuelle simulant une Kinect génère des images de profondeur qui sont sauvegardées. Ces images sont utilisées pour mettre à jour notre grille d'occupation pour la classification. L'image résultante de notre système est comparée à l'image de référence. Les figures 5.15(a) et 5.15(b) (page 100) montrent le simulateur et le résultat de la classification obtenue avec le HMM à un instant donné. Nous avons enregistré une séquence de 430 images et nous avons ensuite calculé sur la totalité de la séquence, la sensibilité et la spécificité du système. Nous avons obtenue 87.14% pour la sensibilité et 98.52% pour la spécificité (voir la table 5.2). Ces résultats montrent que notre méthode réussit à classifier correctement les cellules de la grille.

5.3.2 Etat initial de la grille d'occupation

Rappelons que le HMM a besoin en plus du modèle de transition et d'observation, de la probabilité initiale pour chaque état (Libre "L", Fixe "F" et Mobile "M") (voir section 3.3). Dans cette section nous étudions la sensibilité du HMM face aux différentes valeurs de probabilité initiale pour chaque état. Nous notons $P_0(L), P_0(M), P_0(F)$ la probabilité initiale de l'état "L", "M" et "F". Nous avons choisi $P_0(M) = 0$ et $P_0(L) = P_0(F) = 0.5$. Avec ces valeurs, le HMM réussit à classifier correctement les cellules dans la scène, les cellules occupées par des objets mobiles sont identifiés dés que la personne commence à effectuer des mouvements dans la scène. Nous avons testé le HMM avec d'autres valeurs pour la probabilité initiale. Par exemple, nous avons essayé avec une probabilité égale à 1/3 pour les 3 états et les résultats étaient les mêmes. En effet après un court temps de confusion entre mobile et fixe, le HMM converge vers le même résultat obtenu pour le cas précédent. Nous avons essayé autres valeurs initiales, et nous avons obtenues le même résultat. Ce résultat confirme que la classification des cellules repose essentiellement sur le modèle de transition du HMM entre les différents états et ainsi aucune connaissance *a priori* sur la scène n'est nécessaire.

5.3.3 Influence de la résolution de la grille d'occupation sur la silhouette

Nous avons vu dans la section 3.4.2 que la silhouette de profondeur, utilisée pour construire la fonction de vraisemblance, est extraite en projetant les cellules occupées par des objets mobiles (M) appartenant à une étiquette dans l'image de profondeur. L'avantage de cette technique d'extraction est qu'elle permet la reconstruction de la silhouette à la résolution native du capteur même avec une basse résolution de la grille d'occupation. Dans cette section, nous étudions la qualité de la silhouette de profondeur obtenue pour les différentes résolutions de la grille.

Sur la figure 5.16 (page 101) est affichée la silhouette obtenue pour différentes valeurs de la résolution de la grille. Nous remarquons que pour une résolution de 2.5 cm et 5 cm, la silhouette reconstruite est presque identique. Ce résultat est intéressant, puisque cela nous permet de travailler sur une résolution basse de la grille d'occupation pour identifier les objets mobiles dans la scène et pouvoir reconstruire la silhouette à la résolution native de l'image sans perte d'information. Pour illustrer l'avantage de cette technique, le tableau 5.3 (page 94) récapitule le temps d'exécution de la méthode complète pour l'extraction de la silhouette pour une grille d'occupation de taille $5 \times 5 \times 2.5$ avec différentes résolutions. Nous remarquons que le temps d'exécution pour une grille de résolution 2.5 cm est 6 fois plus lent que le temps nécessaire pour une grille de résolution 5 cm, alors que la silhouette obtenue au final est identique (visuellement) pour les deux résolutions. Pour les résolutions 7.5 et 10 cm, nous remarquons que la qualité de la silhouette est moins bonne (le bras droit n'est pas extrait correctement).

La figure 5.17 (page 102) montre la sortie de la grille d'occupation et la silhouette extraite. Les résultats affichés sur cette figure montrent la capacité de la méthode à classifier correctement les cellules de la grille et à extraire la silhouette de la personne dans différentes situations y compris lorsqu'elle est partiellement observable. Ce qui est intéressant à remarquer, sur la deuxième ligne de la figure 5.17, la silhouette est bien extraite (bras séparés du torse) alors que sur la sortie de la grille d'occupation les bras et le torse sont fusionnés à cause de la faible résolution de la grille d'occupation (ici 5 cm).

5.3.4 Adaptation aux changements da la scène

Un changement de la scène se traduit généralement par un objet (une chaise par exemple), qui a été déplacé d'un endroit à l'autre. Nous montrons dans cette section comment notre méthode

Résolution	Temps d'exécution
2.5cm	0.05 sec
5cm	0.008 sec
7.5cm	0.003 sec
10cm	0.001 sec

TABLE 5.3 – Temps d'exécution de la méthode d'extraction de la silhouette pour une grille d'occupation de taille $5 \times 5 \times 2.5$ mètres avec différentes résolutions.

fait face à ces changements en adaptant le fond grâce à la probabilité de transition ψ que nous avons introduite dans le HMM (voir section 3.3.3).

Pour montrer la capacité de notre méthode à apprendre le fond lorsqu'il change, prenons le scénario de la figure 5.18 (page 103) qui montre une personne en train de déplacer une chaise d'un endroit à un autre. A $t = 0$, la personne est identifiée comme un objet mobile et la chaise comme fixe appartenant au décor de la scène (figure 5.18(a)). A $t = 14$ images, la personne déplace la chaise, et nous remarquons que la chaise est maintenant identifiée comme un objet mobile (figure 5.18(b)). Après un certain temps, les cellules de la grille appartenant à la chaise, passent à nouveau de l'état Mobile (M) à Fixe (F) et la chaise appartient à nouveau au décor de la scène (figures 5.18(c) et 5.18(d)). Ce passage de l'état Mobile à Fixe, pour une cellule donnée, est possible grâce à la probabilité de transition ψ introduite dans le HMM (section 3.3.3).

5.4 Vitesse d'exécution et implémentation sur carte graphique

Avant de discuter de la vitesse d'exécution de la méthode, nous décrivons son implémentation. En effet, pour le développement du système de capture de mouvement, nous avons adopté une approche hybride qui utilise à la fois la puissance du CPU et GPU. L'extraction de la silhouette (grille d'occupation + étiquetage) tourne sur le CPU et a été accélérée avec la librairie Intel Threading Building Blocks qui permet de paralléliser l'exécution d'un programme sur les différents cœurs disponibles sur les processeurs de dernière génération. L'algorithme de filtrage particulaire utilise la carte graphique (GPU) pour effectuer le rendu 3D des particules sur des textures OpenGL. Il est connu dans un filtre particulaire que l'étape la plus coûteuse est celle qui calcule la distance entre chaque particule représentant une configuration du modèle 3D et la silhouette de la personne (voir équation 4.1). Nous avons implémenté cette distance entièrement sur le GPU en utilisant le langage de programmation OpenGL Shading Language qui permet d'utiliser la carte graphique comme une plate-forme de calcul généraliste (en anglais *GPGPU*) où chaque configuration du modèle est rendu sur une texture de profondeur en utilisant la technique de rendu hors ligne (en anglais *offscreen rendering with frame buffer objects*). Cette texture est ensuite passée à un *Shader* pour effectuer une soustraction pixel par pixel entre la texture de profondeur d'une configuration et la texture de profondeur qui correspond à la silhouette de la personne. Le résultat de la soustraction est stocké dans une texture intermédiaire. Finalement, pour obtenir la distance finale de l'équation 4.1), cette dernière texture est réduite en utilisant la fonctionnalité Mip-mapping de OpenGL qui permet de calculer la moyenne de tous les pixels d'une façon accélérée matériellement sur le GPU.

Avec cette implémentation, le système actuel tourne à 4 images par secondes sur un processeur de type intel i7 avec 4 cœurs avec une carte graphique programmable de type Nvidia Quadro K2000. Le grand défaut de l'architecture décrite ci-dessus est le nombre de transfert CPU vers

GPU. En effet, les configurations du modèle sont passées d'une façon séquentielle au GPU pour l'évaluation, et chaque distance calculée par le GPU nécessite un transferts vers le CPU en passant par le bus. Donc pour N particules il faut N transferts CPU/GPU. Or chaque initiation d'un transfert demande un temps de latence avant de démarrer. Ce constat a été vérifié par la charge CPU et GPU qui était faible indiquant que les deux processeurs ne sont pas assez chargés et passent leur temps à attendre la fin du transfert synchrone. Dans le but de réduire ce temps, nous avons commencé à mettre en place une nouvelle architecture qui consiste à faire un rendu de plusieurs configurations d'une façon simultanée dans une grande texture en utilisant une extension de OpenGL dédiée au rendu multiple (en anglais Instanced Rendering). Cette texture est ensuite réduite avec la même technique de Mip-mapping et finalement un seul transfert vers le CPU est effectué renvoyant les distances de l'ensemble des particules. De cette façon le transfert CPU/GPU n'est effectué qu'une seule fois. Nos premiers tests réalisés sont très concluants et nous visons à la fin de la mise en place de cette architecture un gain de vitesse multiplié par 4 rendant ainsi l'implémentation temps réel.

5.5 Conclusion

Dans ce chapitre, nous avons d'abord présenté le *benchmark* que nous avons réalisé et mis ligne. Ce *benchmark* est composé d'un ensemble des séquences vidéos prises à partir d'une caméra RGB-D filmant une personne se déplaçant dans différentes types de scène avec occlusions. La vérité terrain est obtenue avec un système de capture de mouvement basé sur ces marqueurs.

Nous avons ensuite utilisé ce *benchmark* pour montrer que la méthode que nous avons développée améliore considérablement la stabilité et la qualité de suivi des méthodes de filtrage particulaire existantes tout en utilisant un faible nombre de particules. Plus précisément, nous avons montré, avec des résultats quantitatifs et qualitatifs, la capacité de notre méthode à identifier et à retirer les parties invisibles du corps du processus d'estimation du mouvement ce qui améliore nettement la qualité du suivi. Nous avons aussi montré que notre méthode est capable de suivre le mouvement de la personne dans des environnements variés, et elle ne nécessite aucune connaissance *a priori* sur le type de la scène ou l'emplacement des objets et les obstacles dans l'environnement.

Dans la deuxième partie de ce chapitre, nous avons évalué la méthode d'extraction de la silhouette que nous avons développée qui utilise une grille d'occupation couplée à un HMM pour la classification des cellules de la grille d'occupation. Pour cela, nous avons utilisé un simulateur 3D pour valider la capacité du HMM à classifier correctement les cellules. Nous avons montré que cette classification dépend essentiellement du modèle de transition et elle n'est pas sensible à la probabilité initiale du HMM. Nous avons aussi montré qu'avec une grille d'occupation de faible résolution, la méthode est capable d'extraire une silhouette à la résolution native de l'image de la caméra. Nous avons montré la capacité de la méthode à adapter en continu le fond de la scène.

Finalement, nous avons discuté des premiers résultats que nous avons obtenus pour accélérer la version actuelle, qui emploie le GPU pour certaines étapes, en utilisant une extension de OpenGL dédiée au rendu multiple qui a pour but de réduire le transfert entre le CPU et le GPU.

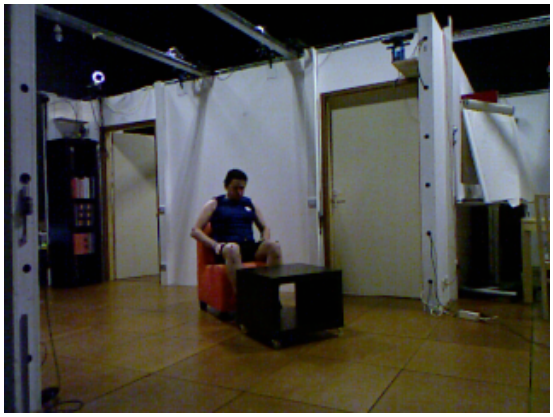
Par contre, notre méthode possède certaines limitations que nous listons ci-dessous :

1. Initialisation : le filtre particulaire nécessite une phase d'initialisation pour partir d'une estimation relativement correcte de la pose de la personne. Cette limitation pose un problème pour le déploiement du système dans des situations réelles.
2. Dépendance de l'état précédent pour la méthode de gestion des occlusions : celle-ci repose sur la meilleure estimation retrouvée à l'instant précédent. Si cette estimation n'est pas

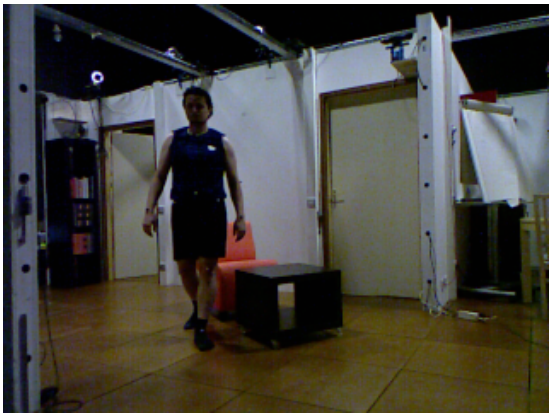
correcte, la méthode d'identification des parties invisibles du corps risque d'échouer.

3. Modèle 3D : le modèle 3D utilisé est composé de cylindres, qui reste approximatif et ne décrit pas parfaitement la forme des parties de corps. Nous avons vu dans ce chapitre qu'un décalage existe toujours entre le squelette extrait et la vraie posture de la personne. Additionnellement, ce modèle 3D est générique et donc n'est pas adapté à la forme de chaque personne dont on extrait la posture.
4. Temps de calcul : un inconvénient majeur des méthodes de filtrage particulaire concerne leur temps de calcul. L'évaluation des différentes hypothèses est coûteuse en terme de vitesse d'exécution.

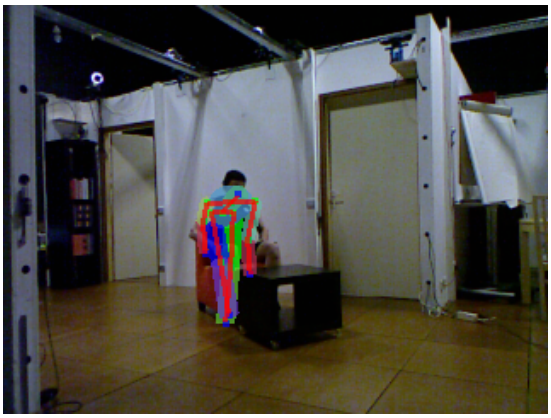
Pour le point 1, une solution potentielle intéressante serait de combiner notre approche avec une méthode qui n'utilise pas un modèle *a priori*. Plus précisément, les méthodes de classification par apprentissage, comme celle proposée par Shotton *et al.* [Shotton *et al.*, 2011] qui utilise les forêts aléatoires, ou celle de Jiu *et al.* [Jiu *et al.*, 2014] qui utilise le *Deep Learning*. Ces deux approches ne nécessitent pas une phase d'initialisation . Pour le point 2, il serait intéressant d'utiliser l'historique sur l'état du système sur plusieurs pas de temps pour ne pas dépendre uniquement de l'état à l'instant précédent, et par conséquent augmenter la robustesse de la méthode de gestion des occlusions. Concernant le troisième point, l'utilisation d'un modèle 3D plus fin, qui caractérise plus précisément la posture de la personne, peut améliorer considérablement la qualité du squelette extrait. Par contre, l'adaptation automatique du modèle 3D aux différentes types de posture des personnes reste difficile. Une solution potentielle serait de paramétrer le vecteur d'état avec les dimensions des segments, et ensuite d'apprendre ces paramètres. De cette façon les paramètres intrinsèques de chaque partie du corps seraient appris pendant le suivi où lors d'une phase d'initialisation, par exemple. Pour le dernier point, un travail d'optimisation, consistant à utiliser la puissance de la carte graphique pour accélérer la vitesse d'exécution du système, est en cours de réalisation.



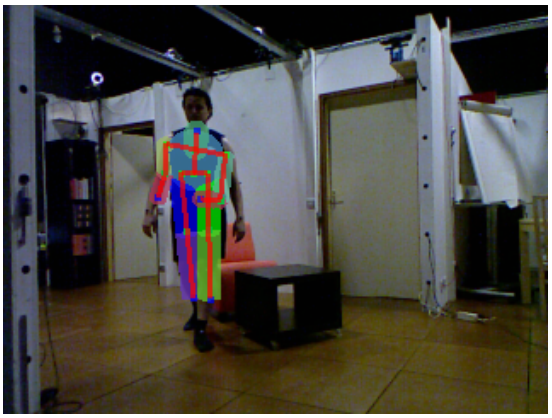
(a) Une personne dans une zone occultée



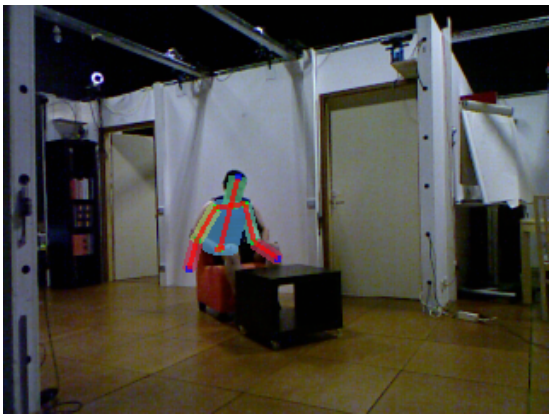
(b) La personne sort de la zone occultée



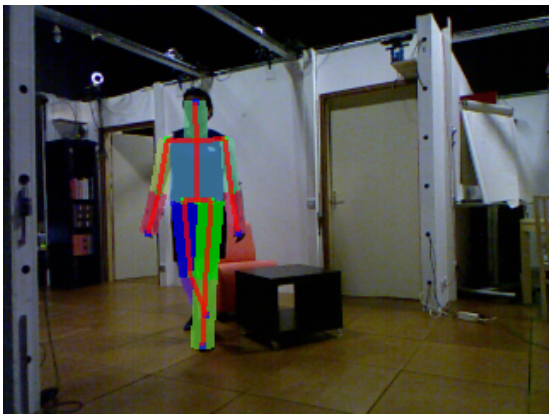
(c) APF génère des fausses postures



(d) APF perd le suivi une fois que la personne est visible à nouveau



(e) Notre méthode réussit à détecter et éliminer les parties du corps cachées et à maintenir un suivi correcte de la partie supérieure du corps



(f) Notre méthode reprend avec succé le suivi lorsque la personne est visible à nouveau

FIGURE 5.12 – Comparaison de notre méthode avec APF lorsque la personne entre a) et sort b) d'une zone occultée. c) et d) squelette obtenu par APF. e) et f) squelette obtenu par notre méthode.

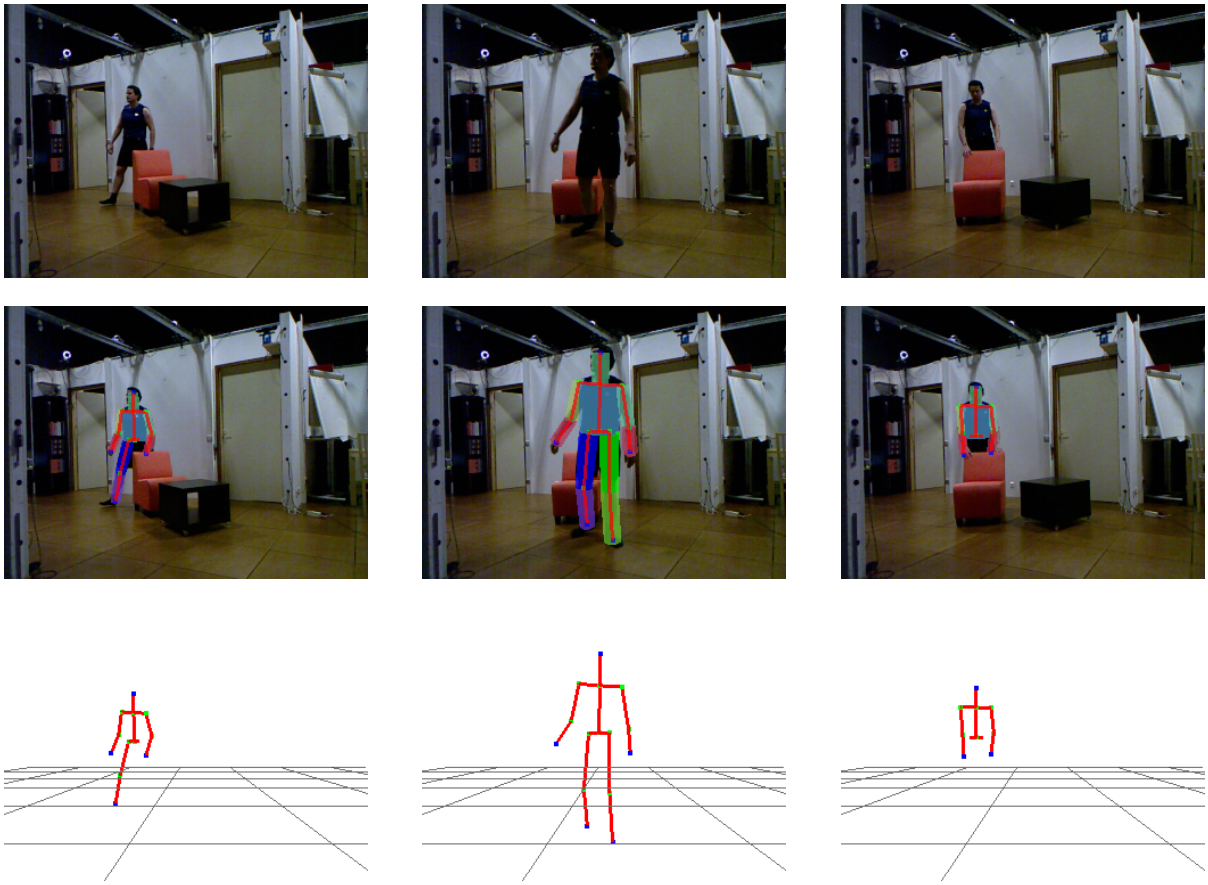


FIGURE 5.13 – Résultats visuels de suivi obtenus par notre méthode sur des situations prises de notre base de données de capture de mouvement qui montrent la capacité de notre approche à produire une estimation correcte du mouvement de la personne dans des situations d'occlusion.

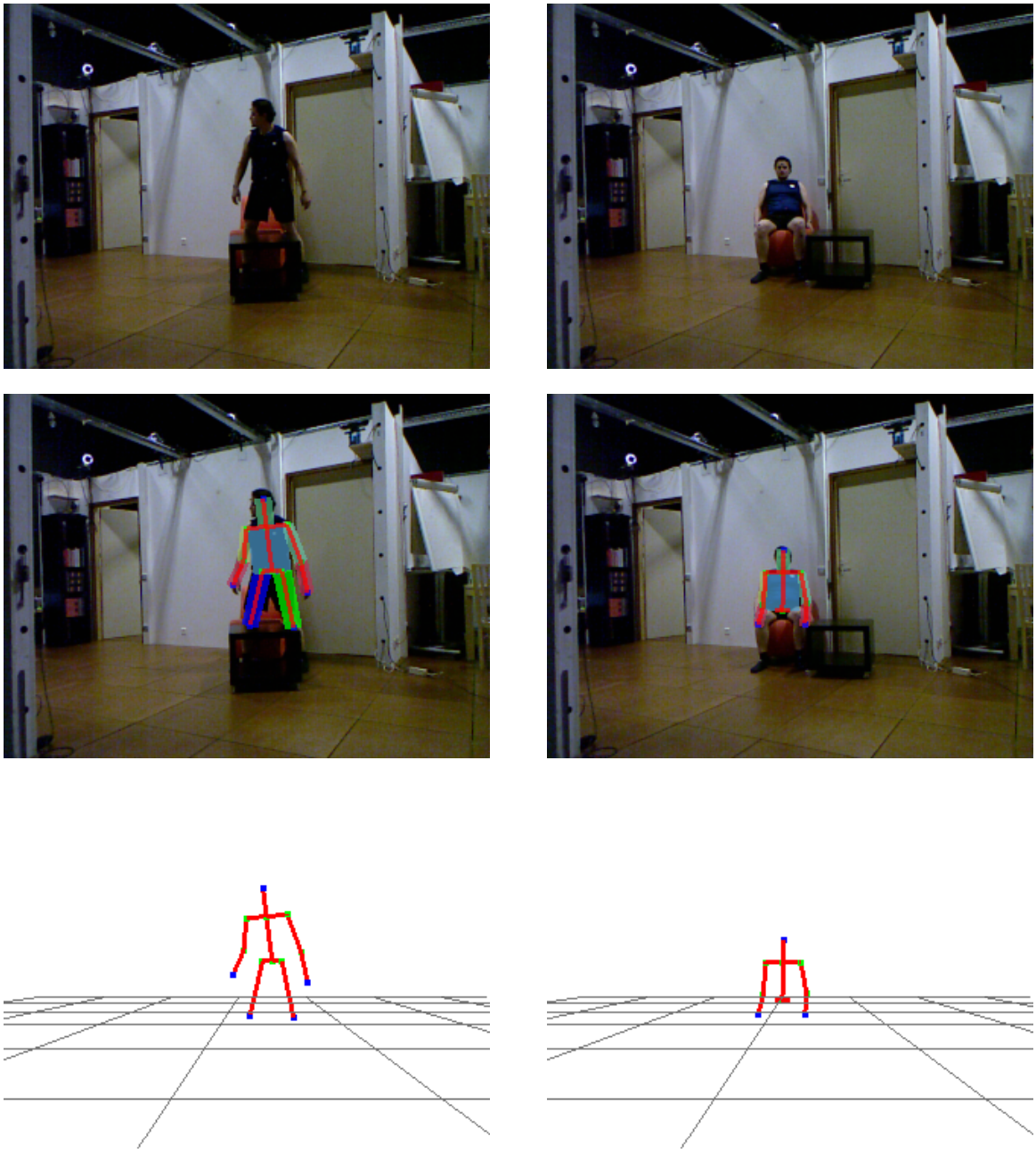
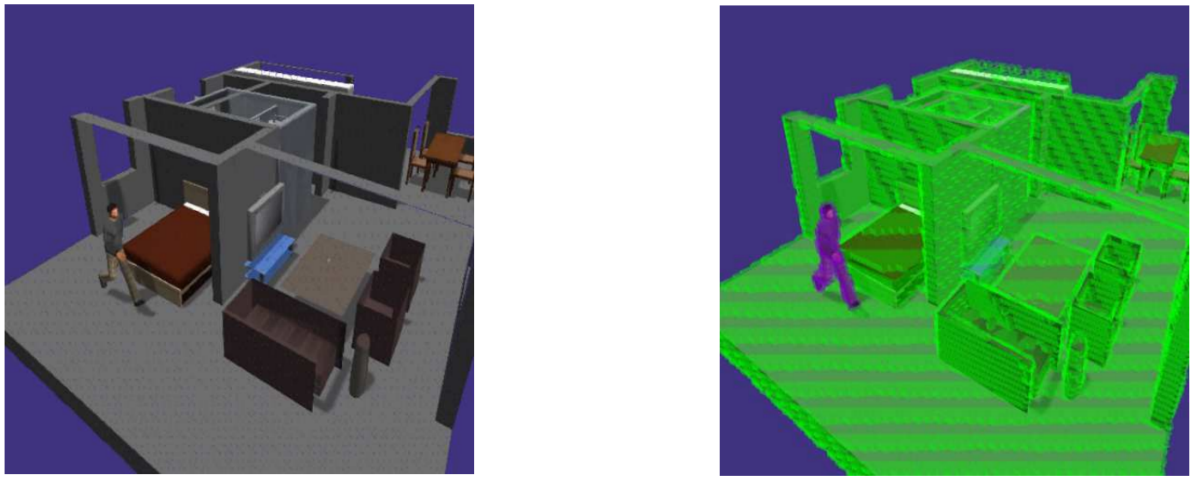


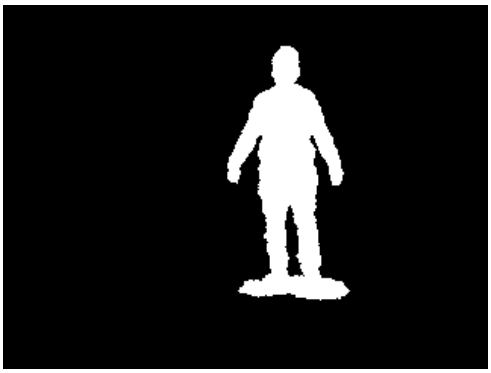
FIGURE 5.14 – D'autres résultats visuels de suivi obtenus par notre méthode sur des situations prises de notre base de données de capture de mouvement montrant la capacité de notre méthode à suivre correctement le mouvement de la personne dans des situations d'occlusion.



(a) (b)

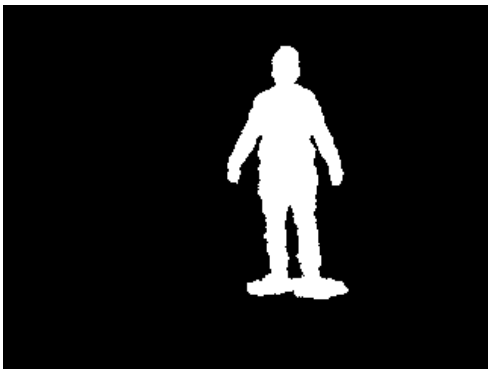
FIGURE 5.15 – a) Capture d'écran du simulateur b) Résultat de la classification des cellules avec le HMM.

(a) Grille avec une résolution de 2.5 cm

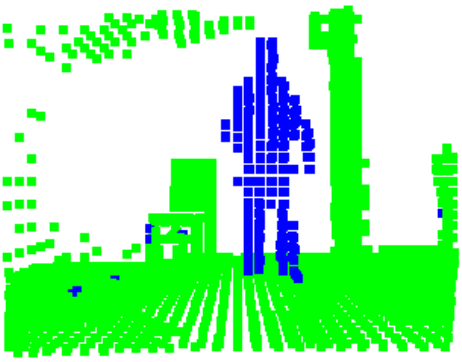


(b) Silhouette reconstruite pour grille de 2.5 cm

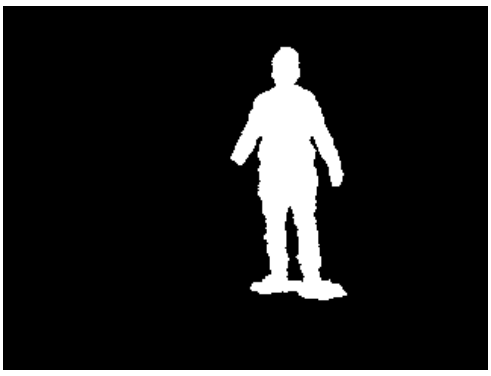
(c) Grille avec une résolution de 5 cm



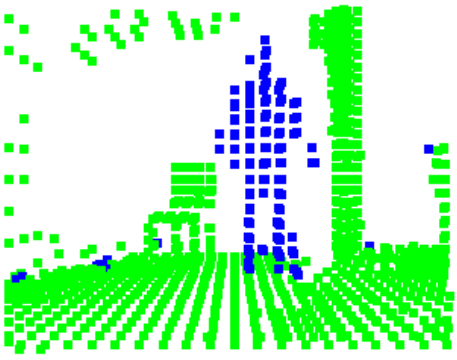
(d) Silhouette reconstruite pour grille de 5 cm



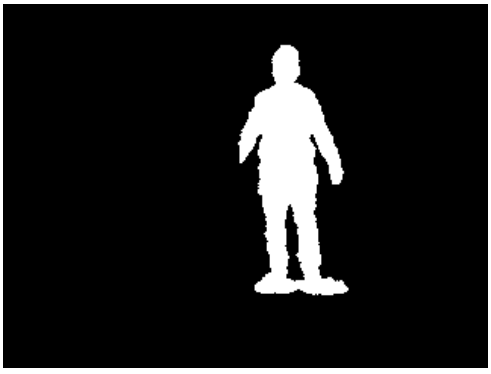
(e) Grille avec une résolution de 7.5 cm



(f) Silhouette reconstruite pour grille de 7.5 cm



(g) Grille avec une résolution de 10 cm



(h) Silhouette reconstruite pour grille de 10 cm

FIGURE 5.16 – Impact de la résolution de la grille d'occupation sur la qualité de la silhouette extraite. La grille en question est de taille $5 \times 5 \times 2.5$ mètres. Seul le centre des cubes composant la grille est affiché pour mieux illustrer le changement de la résolution.

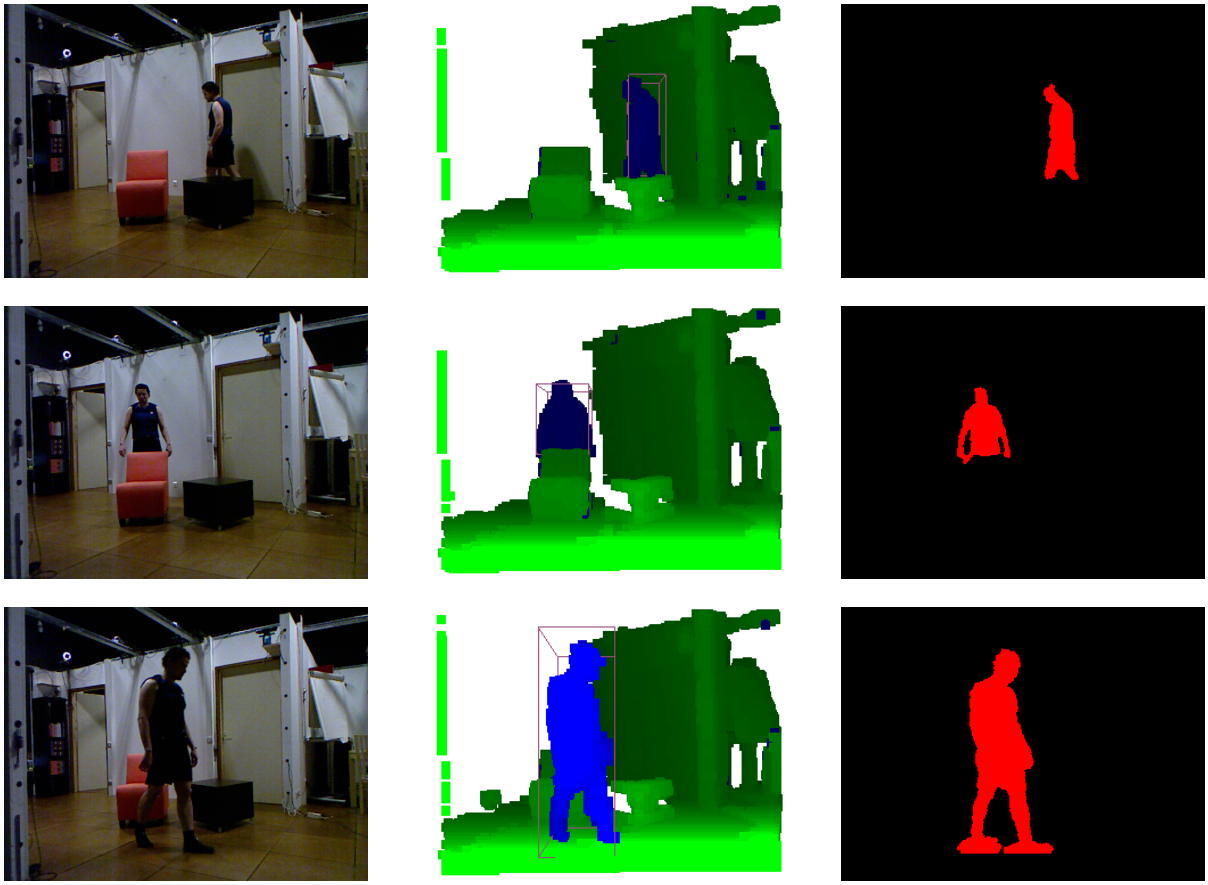
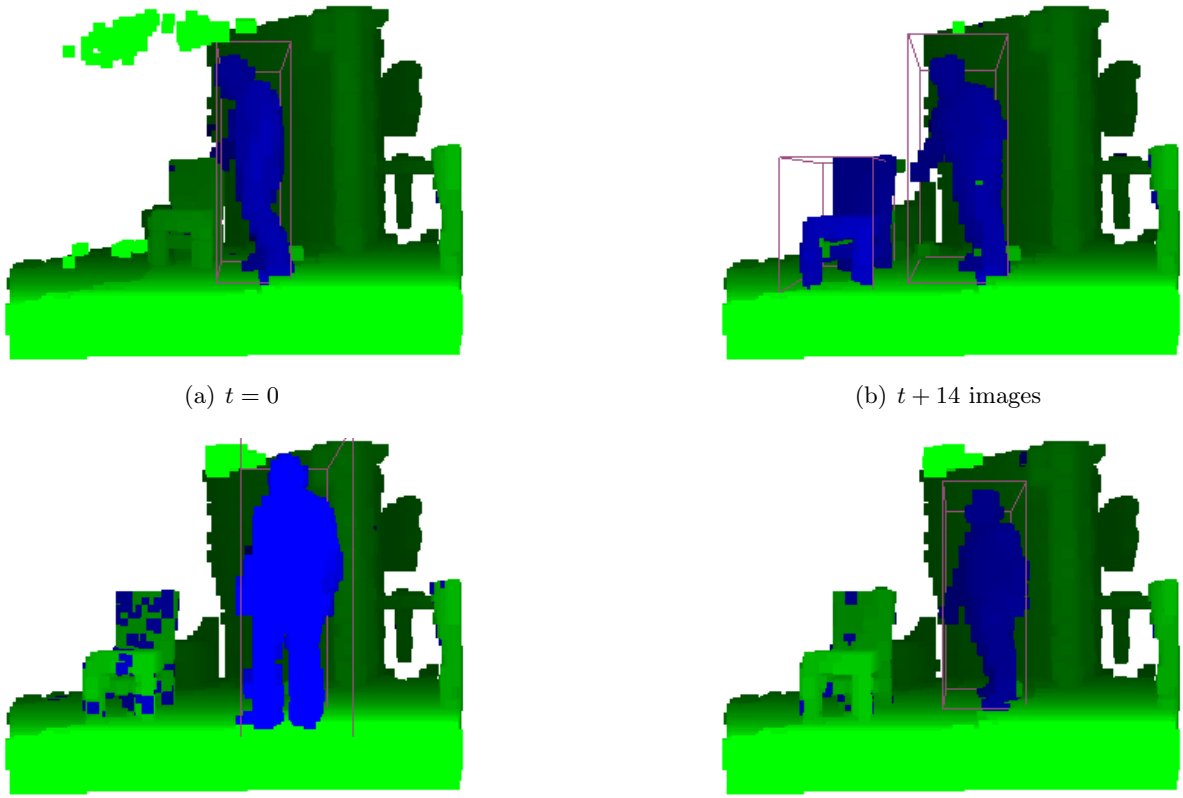


FIGURE 5.17 – Première colonne : Image RGB d'une Personne se déplaçant dans une scène. Deuxième colonne : Classification des cellules de la grille d'occupation par le HMM. La boîte englobante de la personne est affichée. Troisième colonne : La silhouette extraite de la personne.



(a) $t = 0$ (b) $t + 14$ images

(c) $t + 93$ images (d) $t + 117$ images

FIGURE 5.18 – Images montrant la capacité de notre méthode à apprendre en continu le fond de la scène.

Conclusion générale

Dans cette partie, le système de suivi du mouvement de la personne développé au cours de cette thèse a été présenté. Ce système est capable de reconstruire en 3D la posture de la personne filmée dans une scène dynamique contenant des obstacles. Dans la littérature, le suivi dans un environnement dynamique avec occlusions est peu traité. La plupart des approches existantes supposent que seule la personne se déplace dans une scène statique sans obstacles. Certains travaux traitent les occlusions en utilisant plusieurs caméras ou en étiquetant manuellement les zones susceptibles de générer des occlusions avant de lancer le suivi. Ces solutions ne passent pas à l'échelle, et ne régleront pas le problème d'occlusion. En effet, même avec plusieurs caméras, les occlusions existeront toujours, et à cause de la dynamique de la scène, un étiquetage manuel des zones occultées n'est pas suffisant.

La contribution majeure de cette partie est la proposition d'une méthode de suivi du mouvement capable d'identifier d'une façon automatique les parties invisibles du corps. Cette identification se fait en interrogeant la grille d'occupation pour déterminer la visibilité de chaque partie du corps. Ces parties du corps sont ensuite exclues du processus d'estimation de la pose. Nous avons montré par les expériences que nous avons menées, que notre approche fonctionne lorsque la silhouette de la personne est partiellement observable.

Une deuxième contribution de ce travail, est la proposition d'une méthode pour apprendre en ligne le fond de la scène et d'extraire la silhouette de la personne. En effet, dans des conditions réelles, la scène n'est jamais fixe et elle évolue en permanence. La difficulté est alors de pouvoir localiser la personne dans ces conditions et d'extraire sa silhouette. En effet, la majorité des méthodes existantes suppose que le fond est fixe et connu et procède à une soustraction pixel-à-pixel pour extraire la silhouette de la personne. Ces méthodes nécessitent aussi une phase d'initialisation où la personne doit être en dehors de la scène pour extraire le fond qui sera supposé fixe pendant tout le temps de suivi. Ces hypothèses ne sont pas compatibles avec nos hypothèses de travail, car le système de suivi développé va équiper un robot mobile dans un environnement qui évolue. Pour cela, nous avons proposé une méthode qui utilise une grille d'occupation et un HMM pour apprendre en ligne le fond et extraire la silhouette de la personne et ne nécessite pas une phase d'initialisation. L'utilisation d'une grille d'occupation a plusieurs avantages notamment parce qu'elle permet d'avoir une représentation unique de l'espace, aussi elle est capable d'intégrer naturellement plusieurs capteurs fixes ou mobiles, ce qui peut être intéressant, par exemple, si nous souhaitons intégrer les informations provenant d'autres types de capteurs présents dans l'environnement.

Une dernière contribution de cette partie est la proposition d'un *benchmark* disponible en ligne ([motion capture dataset, 2015]), composé d'un ensemble de séquences vidéo d'une personne se déplaçant dans différents types de scènes avec obstacles. La vérité terrain est obtenue par un système externe de capture du mouvement utilisant des marqueurs réfléchissants fixés sur la personne.

Troisième partie

Localisation d'une caméra mobile dans un environnement dynamique

Introduction

Le système de reconstruction de la posture développé dans la partie précédente a pour but d'être installé sur un robot d'assistance qui se déplace dans un environnement. Nous avons vu que ce système est capable d'extraire la silhouette d'une personne, en utilisant un nuage de points provenant de la caméra pour mettre à jour la grille d'occupation, et ainsi apprendre en continu le fond dynamique de la scène [7]. Dans notre système, le repère de la grille d'occupation est global, et le nuage de points provenant de la caméra doit y être projeté correctement.

Dans le cas où la caméra est montée sur un robot mobile, nous avons donc besoin de connaître la transformation entre la position de la caméra et le repère global, afin de mettre à jour correctement la grille d'occupation. Pour obtenir cette transformation, représentée par une rotation et une translation, nous avons besoin d'une méthode d'odométrie, qui estime le mouvement de la caméra. Ce problème est bien étudié dans la littérature pour le cas où la scène est statique. Par contre, dans notre cas, l'environnement est dynamique, et la localisation dans de telles conditions reste un défi scientifique ouvert. En effet, dans un milieu dynamique, le déplacement des objets présents dans la scène induit une difficulté supplémentaire pouvant nuire à l'estimation du mouvement de la caméra.

Dans cette partie, nous présentons la méthode d'odométrie visuelle que nous avons développée pour localiser une caméra mobile dans un environnement dynamique. Cette méthode est capable d'isoler les pixels de l'image appartenant aux objets mobiles de la scène, de façon à les enlever, et ainsi avoir une estimation précise du mouvement de la caméra.

La structure de la partie est la suivante. Dans le chapitre 6, nous décrivons les méthodes existantes d'odométrie visuelle. Ensuite, dans le chapitre 7, nous faisons un rappel sur les outils qui vont servir pour le développement de notre méthode de localisation visuelle. Dans le chapitre 8, nous présentons la méthode d'odométrie visuelle que nous avons développée. Finalement, dans le chapitre 9, nous évaluons les performances de la méthode.

7. Du point de vue de la caméra, le fond de la scène peut changer parce que les objets dans la scène se déplacent, mais aussi parce que la caméra se déplace ou change d'orientation.

Chapitre 6

Etat de l'art

Sommaire

6.1 Méthodes par extraction des points d'intérêt 111
6.2 Méthodes directes 115
6.2.1 Estimation du mouvement de la caméra par minimisation de l'erreur géométrique 115
6.2.2 Estimation du mouvement de la caméra par minimisation de l'erreur photométrique 116
6.3 Conclusion 118

L'odométrie visuelle (en anglais *visual odometry*) consiste à estimer le mouvement effectué par une caméra mobile à partir d'une séquence d'images. En effet, soit les deux images I_t et I_{t+1} de la figure 6.1 obtenues à deux instants successifs par la caméra, l'odométrie visuelle estime le déplacement que la caméra a effectué entre ces deux instants. Ce déplacement est représenté par une rotation R et une translation T. Ce processus d'estimation est répété à chaque réception d'une nouvelle image, et de cette façon la trajectoire effectuée par la caméra est obtenue.

L'odométrie visuelle est un sujet bien traité dans la littérature et avec l'apparition des caméras RGB-D, ce sujet est devenu de plus en plus actif ces dernières années. Actuellement, de nombreuses méthodes d'odométrie visuelle existent, mais toutes ces méthodes peuvent être classées dans deux grandes catégories : La première catégorie appelée creuse (en anglais *sparse*) et regroupe toutes les méthodes qui utilisent un sous ensemble de points (aussi appelés points d'intérêts) dans l'image pour estimer le mouvement de la caméra, et la deuxième catégorie appelée directe ou dense, regroupe toutes les méthodes qui utilisent la totalité des informations existantes dans l'image pour l'estimation.

Dans la suite nous décrivons dans un premier temps le principe de fonctionnement des méthodes creuses basées sur l'extraction des points d'intérêt de l'image et les méthodes denses qui utilisent tous les pixels dans l'image. Ensuite nous présentons les avantages et inconvénients de chacune de ces approches et dans la conclusion de ce chapitre, nous validons le choix de la méthode que nous avons choisie.

6.1 Méthodes par extraction des points d'intérêt

La localisation par extraction des points d'intérêt consiste d'abord à extraire un ensemble de primitives visuelles dite *Features* telles que les points de Harris ([Harris and Stephens, 1988]) comme le montre la figure 6.2. Ces points d'intérêt sont caractérisés par une intensité qui varie

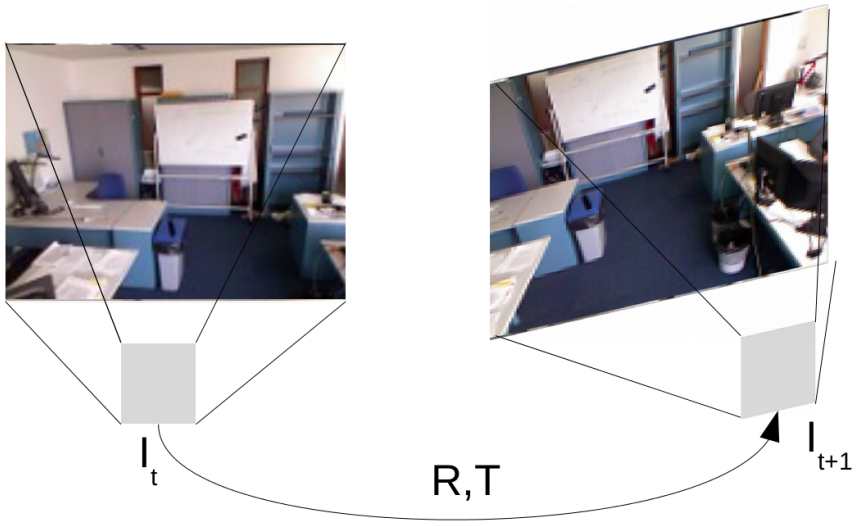


FIGURE 6.1 – L'odométrie visuelle consiste à estimer le mouvement de la caméra (représenté par une rotation R et une translation T) entre deux images consécutives I_t et I_{t+1}.

fortement dans une ou plusieurs directions. Ensuite, une étape de mise en correspondance ou appariement est appliquée qui consiste à trouver pour chaque pixel de l'image, son correspondant dans l'image suivante comme le montre la figure 6.3. Cette correspondance est faite en calculant d'abord un descripteur associé à chaque pixel et ensuite d'utiliser une fonction de mesure de similarité entre ces descripteurs. Plusieurs variantes existent permettant d'extraire des descripteurs locaux, invariants et robustes aux rotations et au changement d'échelle telles que FAST ([Rosten and Drummond, 2006]), SIFT ([Lowe, 1999]) et SURF ([Bay *et al.*, 2008]). Rublee *et al.* [Rublee *et al.*, 2011] proposent un nouveau descripteur appelé ORB comme alternative à SIFT et SURF qui est deux fois plus rapide que SIFT. La mise en correspondance est l'étape la plus délicate dans ces approches, et il arrive souvent d'avoir des mauvais appariements, c'est à dire des pixels de la première image qui ne sont pas associés correctement à leurs correspondants dans l'image suivant. Pour détecter et éliminer ces mauvais appariements, l'algorithme de RANSAC (*Random Sample Consensus* [Fischler and Bolles, 1981]) est souvent utilisé. Comme exemple des méthodes d'odométrie visuelle creuses utilisant RANSAC, nous citons la méthode de Torr et Zisserman [Torr and Zisserman, 2000a], Konolige *et al.* [Konolige *et al.*, 2006] et Nister *et al.* [Nister *et al.*, 2006].

A partir de ces points appariés, une première estimation de la transformation (rotation + translation) est calculée en utilisant la matrice fondamentale ou essentielle obtenue en utilisant la géométrie épipolaire ([Hartley and Zisserman, 2004]) qui décrit la relation géométrique entre les pixels en correspondance dans les deux images comme le montre la figure 6.4. En effet, pour un pixel x_L de l'image I_L, les points de l'espace 3D ayant comme image le point x_L se situent sur la ligne de vue L_2. Les correspondants potentiels de x_L dans l'image I_R se situent sur la droite l_2 qui correspond à la projection de L_2 dans l'image I_R. Ainsi pour chaque paire de points

FIGURE 6.2 – Les points de Harris extraits de l'image.

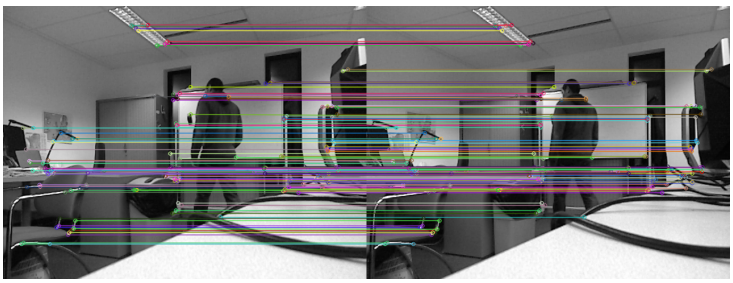


FIGURE 6.3 – Mise en correspondance des pixels entre deux images. Cette correspondance est représentée par les lignes joignant les pixels.

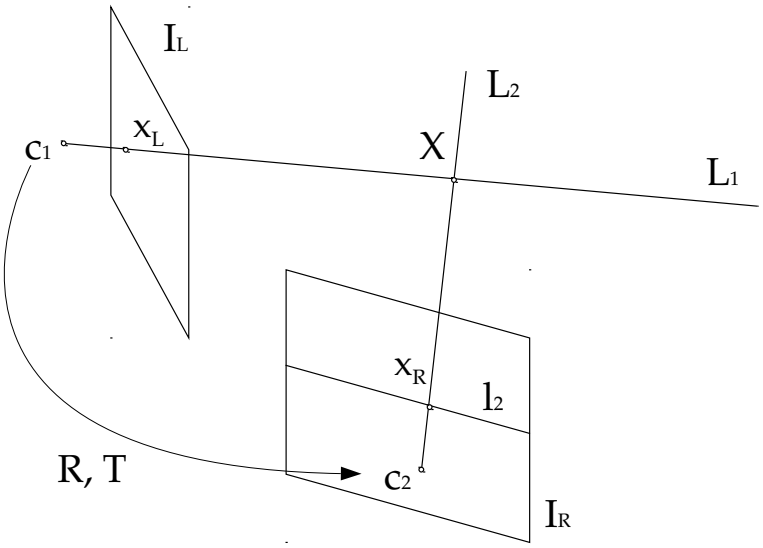


FIGURE 6.4 – Géométrie épipolaire

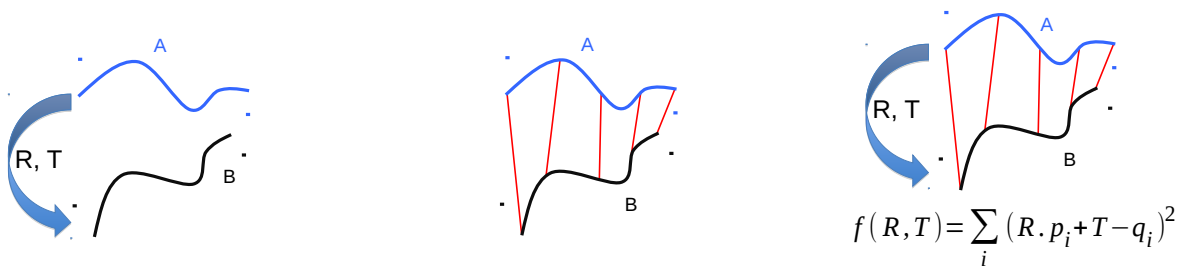
(appariés), la contrainte épipolaire s'écrit sous la forme suivante :

$$x_R{}^T \times F \times x_L = 0.$$

F la matrice fondamentale ([Faugeras, 1992]) de dimension 3×3 qui décrit la géométrie reliant I_R et I_L et encapsule la transformation (la rotation et la translation) rigide entre ces deux images. F est calculée ensuite à partir d'un ensemble de points (appariés) entre deux images. Plusieurs méthodes et variantes existent pour le calcul de F (voir [Longuet-Higgins, 1987], [Zhang and Kanade, 1998], [Torr and Zisserman, 2000b], [Torr, 2002]). L'obtention de la transformation reliant les deux images se fait par une décomposition en valeurs singulières de F. Cette transformation est utilisée comme un point de départ pour une procédure itérative de minimisation non linéaire de l'erreur de re-projection des points d'intérêts connue sous le nom du *Bundle Adjustment* ([Triggs *et al.*, 2000]). La fonction à minimiser s'écrit sous la forme suivante :

$$f = \sum_{i=1}^{L} \sum_{j=1}^{N} d(x_j{}^i, R_i X_j + T_i)^2,$$

avec L le nombre d'images (pour deux images successives L = 2) et N est le nombre de points 3D obtenue par une triangulation des pixels appariés. R_i et T_i représente respectivement la rotation et la translation. X_j est le point 3D obtenu par triangulation pour chaque paire de pixels appariés. Finalement, d est la distance géométrique entre le pixel projeté $\hat{x}_j^i = R_i X_j + T_i$ et le pixel correspondant mesuré $x_j{}^i$. La fonction f est minimisée par rapport à R_i, T_i et X_j. L'algorithme de Levenberg-Marquardt ([Ananth, 2004]) est souvent utilisé pour obtenir une solution itérative par moindres carrés. Une synthèse sur les principales méthodes d'odométrie visuelle existantes basées sur l'extraction des points d'intérêts a été effectuée par Scaramuzza et Fraundorfer [Scaramuzza and Fraundorfer, 2011] en 2011.



(a) (b) (c)

FIGURE 6.5 – L'algorithme ICP : a) Les deux ensembles de points b) L'étape d'association consiste à trouver pour chaque point d'un ensemble son voisin le plus proche dans le deuxième ensemble. c) La transformation représenté par R et T est calculée en minimisant la fonction quadratique $f(R,T)$.

6.2 Méthodes directes

Les méthodes directes utilisent toutes les informations disponibles dans l'image pour estimer le mouvement de la caméra. L'estimation de la position de la caméra ainsi que la mise en correspondance des pixels s'effectuent dans la même procédure d'optimisation. L'étape d'optimisation se fait soit en minimisant l'erreur des intensités communes aux deux images, soit en minimisant l'erreur géométrique entre les surfaces. Dans la suite nous décrivons le principe de fonctionnement de ces deux techniques.

6.2.1 Estimation du mouvement de la caméra par minimisation de l'erreur géométrique

Les méthodes qui minimisent l'erreur géométrique sont basées sur l'algorithme ICP (*Iterative Closest Point* [Besl and McKay, 1992]). Nous décrivons dans la suite le principe de fonctionnement de cet algorithme, ensuite nous expliquons comment cet algorithme est utilisé en odométrie visuelle.

Soit A et B, deux ensembles de points 3D, l'algorithme ICP cherche à trouver la transformation entre ces deux nuages de points qui minimise la distance géométrique entre eux comme le montre la figure 6.5(a). ICP procède d'une façon itérative, et à chaque itération, une étape d'association est appliquée consistant à trouver pour chaque point p_i de A , le point q_i dans B dont la distance géométrique est la plus proche (figure 6.5(b)). Cette étape est suivi d'une deuxième étape qui consiste à trouver la transformation entre les points associés en minimisant la fonction quadratique suivante (figure 6.5(c)) :

$$f(R,T) = \sum_{i=1}^{N} (R \cdot p_i + T - q_i)^2,$$

avec N le nombre des points associés, R et T représentent la transformation qui minimise la distance géométrique entre les points associés. La nouvelle transformation obtenue à la fin de chaque itération est appliquée sur l'ensemble de points B et le processus de minimisation itérative est répété sur le nouveau ensemble de points en calculant à nouveau une nouvelle association suivi d'une minimisation. L'algorithme s'arrête lorsqu'un certain nombre d'itérations est atteint ou si $f(R,T)$ est inférieure à un certain seuil prédéfinit.

L'étape d'association est critique dans l'algorithme ICP et elle est coûteuse en terme de temps de calcul et elle est sensible aux points aberrants. Ils existent plusieurs variantes de l'algorithme dont le but est d'accélérer la recherche du voisin le plus proche. Parmi ces méthodes, nous citons celle de Friedman *et al.* [Friedman *et al.*, 1977] qui utilise un *Kd-Tree* pour accélérer la recherche de voisin, et la méthode de Nutcher *et al.* [Nuchter *et al.*, 2007] qui emploie un cache pour accélérer la recherche dans le *Kd-Tree*. Pour rendre robuste l'étape d'association contre les points aberrants, Dorai *et al.* [Dorai *et al.*, 1997] utilisent une méthode de moindres carrés pondérés (en anglais *Weighted Least-Square*). D'autres techniques ont été développées pour éliminer les points aberrants (en anglais *outliers rejection*). Ces techniques sont décrites Dans les travaux de Pomerleau *et al.* [Pomerleau *et al.*, 2009]. Les auteurs proposent aussi une nouvelle méthode pour éliminer ces points. Dans les travaux de Rusinkiewicz et Levoy [Rusinkiewicz and Levoy, 2001] et Pomerleau *et al.* [Pomerleau *et al.*, 2015], les différentes variantes de l'algorithme d'ICP sont décrites.

L'application de l'algorithme ICP pour estimer le mouvement d'une caméra de type RGB-D consiste à re-projeter les pixels de profondeur dans l'espace pour former un nuage de points 3D. L'algorithme ICP est ensuite utilisé pour estimer le mouvement de la caméra en alignant les deux nuages de points obtenus à deux instants successifs. Comme exemple d'application, nous citons la méthode de reconstruction 3D d'une scène à partir d'une Kinect développée par Microsoft en 2011 ([Newcombe *et al.*, 2011a]) qui utilise une variante de l'algorithme d'ICP. Nous citons aussi les méthodes de Tykkala *et al.* [Tykkala *et al.*, 2011] et Pomerleau *et al.* [Pomerleau *et al.*, 2011] qui utilisent aussi l'algorithme ICP.

6.2.2 Estimation du mouvement de la caméra par minimisation de l'erreur photométrique

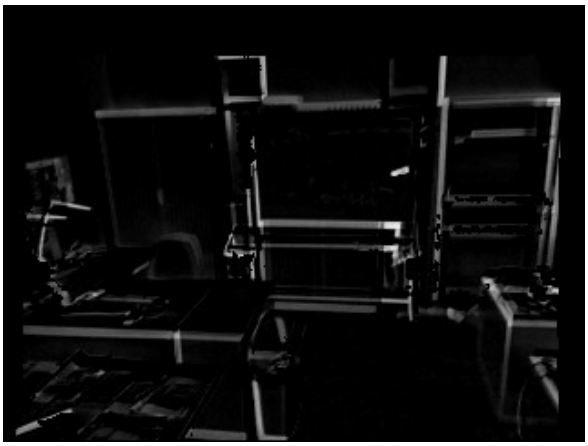
Ces méthodes consistent à estimer le mouvement de la caméra en minimisant l'erreur des intensités (aussi appelée erreur photométrique) de tous les pixels de l'image. Ces méthodes sont considérées comme une extension 3D des méthodes de calcul du flux optique (voir les travaux de Lucas-Kanade [Lucas and Kanade, 1981] et Baker et Matthews [Baker and Matthews, 2004]). Sur la figure 6.6, à partir de deux images reçues à deux instants successifs (figures 6.6(a) et 6.6(b)), une image appelée résidu est calculée par soustraction de ces deux images (figure 6.6(c)). Sur l'image résultante, la luminance de chaque pixel varie entre 0 pour le noir et 255 pour le blanc. L'erreur associé à chaque pixel est proportionnelle à la valeur de sa luminance. C'est à dire, plus le pixel est éclairé, plus l'erreur est élevée. L'objectif de ces approches est de trouver le déplacement ξ entre les deux images qui minimise les intensités de tous les pixels de l'image résidu. Le déplacement ξ est obtenu par une méthode de moindres carrés non linéaire en minimisant la fonction quadratique suivante :

$$E(\xi) = \sum_{i=1}^{n} \left(I_{t+1}(w(\xi, p_i)) - I_t(p_i)\right)^2, \tag{6.1}$$

qui représente la somme quadratique des erreurs des intensités de tous les pixels p_i, n étant le nombre des pixels dans l'image, I_t et I_{t+1} sont les images reçues à l'instant t et $t+1$. L'avantage de cette méthode par rapport à ICP est qu'elle ne nécessite pas une étape d'association qui est coûteuse en terme de temps de calcul. Par comparaison avec la méthode basée sur l'extraction des points d'intérêts, cette méthode utilise tous les pixels de l'image dans le processus d'estimation du mouvement de la caméra, permettant une estimation robuste et précise. Dans la suite, nous citons quelques méthodes d'odométrie minimisant l'erreur photométrique entre les images.

(a)

(b)



(c)

FIGURE 6.6 – Principe de l'odométrie visuelle dense par minimisation de l'erreur photométrique. a) image à l'instant t b) image à l'instant $t+1$ c) l'image résidu obtenue par soustraction des deux images. La luminance de chaque pixel est proportionnelle à l'erreur. Cette méthode cherche à trouver la transformation qui minimise l'erreur des intensités de tous les pixels dans l'image résidu.

Parmi les première méthodes d'odométrie visuelle dense minimisant l'erreur des intensités des pixels, nous citons les travaux de Comport *et al.* [Comport *et al.*, 2010]. Avec l'apparition des caméras 3D de type RGB-D (Microsoft Kinect et Asus Xtion), Audras *et al.* [Audras *et al.*, 2011] et Steinbruecker *et al.* [Steinbruecker *et al.*, 2011] proposent une méthode qui utilise une caméra RGB-D pour minimiser l'erreur photométrique entre deux images. pour rendre robuste cette approche dans un environnement dynamique, Audras *et al.* [Audras *et al.*, 2011] utilisent une méthode de moindres carrés pondérés basé sur M-estimateurs ([David, 1970]) pour réduire l'influence des points aberrants dans la procédure de l'estimation en affectant un poids d'importance à chaque pixel proportionnel à son erreur. Kerl *et al.* [Kerl *et al.*, 2013b] comparent plusieurs variantes des méthodes de moindres carrés pondérés basées sur les M-estimateurs. Whelan *et al.* [Whelan *et al.*, 2013b] proposent une méthode hybride qui minimise simultanément l'erreur géométrique et photométrique au sens des moindres carrées. Forster *et al.* [Forster *et al.*, 2014] utilisent une approche semi-dense pour l'estimation du mouvement de la caméra.

6.3 Conclusion

Dans ce chapitre, nous avons décrit le principe de l'odométrie visuelle qui consiste à estimer le mouvement d'une caméra mobile à partir d'une séquence d'images. Nous avons vu que les méthodes existantes peuvent être regroupées en deux familles, la première famille appelée creuse, qui utilise un ensemble de points d'intérêts de l'image pour estimer le déplacement de la caméra, et la deuxième famille est appelée dense qui, quant à elle, utilise toutes les informations de l'image pour estimer le mouvement de la caméra. Cette deuxième famille peut être divisée en deux sous catégories, la première catégorie minimise l'erreur géométrique entre deux nuages de points en utilisant l'algorithme ICP, et la deuxième catégorie minimise l'erreur photométrique entre les deux images en utilisant une extension de la méthode de Lucas-Kanade pour le calcul du flux optique. C'est cette dernière approche qui sera retenue dans cette thèse pour les raisons suivantes :

- Les méthodes denses fournissent une estimation robuste et plus précise que les méthodes basées sur l'extraction des points d'intérêts. Ce résultat a été démontré dans les travaux de Newcombe *et al.* [Newcombe *et al.*, 2011b] et Lovegrove *et al.* [Lovegrove *et al.*, 2011]. En effet, les méthodes par extraction des points d'intérêts ne sont pas précises puisqu'elles n'utilisent pas tous les pixels de l'image, et la qualité de l'estimation du mouvement de la caméra par ces méthodes dépend de l'étape d'appariement intermédiaire qui est souvent mal conditionnée et non robuste pouvant produire une fausse estimation du mouvement de la caméra.
- Par comparaison avec la méthode d'alignement utilisant l'ICP, cette dernière nécessite une étape d'association qui est coûteuse en temps de calcul. Tandis que l'approche dense par flux optique ne nécessite aucune étape d'association et c'est dans la procédure de minimisation que l'association est faite simultanément au moment de l'estimation du mouvement de la caméra.

L'approche dense par flux optique que nous avons choisi fournit une estimation très précise du mouvement de la caméra dans une scène statique. Par contre, cette approche dans son état actuel n'est pas capable d'estimer correctement le mouvement de la caméra dans un milieu dynamique. Rappelons, que pour nous, un milieu dynamique est un environnement dont l'état structurel peut être modifié à tout moment. Par exemple, des objets ou des meubles peuvent être déplacés d'un endroit à un autre, ou aussi des personnes peuvent se déplacer dans ce milieu. Ainsi, dans tel environnement, le mouvement des objets de la scène génère des pixels aberrants dans

l'image, ainsi une méthode dense qui utilise tous les pixels de l'image va certainement produire une estimation du mouvement de la caméra erronée si elle n'exclut pas ces points aberrants. Pour rendre robuste cette méthode dans un environnement dynamique, nous avons proposé une extension à cette méthode qui permet d'identifier les points aberrants de la scène et les retirer du processus de l'estimation du mouvement. La différence entre notre approche et celle qui utilise une méthode de moindre carrés pondérés (voir les travaux de Comport *et al.* [Comport *et al.*, 2010] et Kerl *et al.* [Kerl *et al.*, 2013b]), est que cette dernière réduit l'impact des pixels aberrants sur le processus de l'estimation mais sans les retirer complètement, alors que notre méthode exclut complètement ces points aberrants.

Dans le chapitre suivant, nous introduisons les outils théoriques nécessaires pour le développement de notre méthode d'odométrie visuelle. Dans un premier temps, nous introduisons une représentation minimaliste décrivant le mouvement d'un objet rigide dans l'espace en utilisant l'algèbre de Lie, ensuite le modèle simplifié de la caméra sera décrit. Un rappel sur les méthodes d'optimisation par moindres carrés sera détaillé. Finalement, nous décrivons l'algorithme de RANSAC que nous allons utilisé pour rendre robuste la méthode de localisation visuelle dense dans une scène dynamique.

Chapitre 7

Contexte méthodologique

Sommaire

7.1 Représentation minimaliste par l'algèbre de Lie du mouvement d'un objet rigide dans l'espace. 121
7.2 Modèle de la caméra 122
7.3 Méthode de moindres carrés 123
7.4 Ransac 125
7.5 Conclusion 127

Dans ce chapitre, nous faisons un rappel sur les outils théoriques nécessaires pour le développement de la méthode d'odométrie visuelle. Nous décrivons d'abord comment le mouvement d'un objet rigide peut être représenté sous une forme minimaliste en utilisant l'algèbre de Lie. Nous présentons ensuite le modèle simplifié de la caméra permettant de projeter un point 3D dans le plan image et vice versa. Un rappel sur les méthodes d'optimisation par moindres carrés sera présenté dans ce chapitre. Finalement, nous présentons l'algorithme RANSAC que nous allons utilisé pour rendre robuste la méthode d'odométrie visuelle dense.

7.1 Représentation minimaliste par l'algèbre de Lie du mouvement d'un objet rigide dans l'espace.

Le mouvement d'un objet rigide dans l'espace 3D est parfaitement décrit par sa position (ou translation) et sa rotation par rapport à un repère fixe à tous les instants. Le mouvement d'un corps rigide possède six degrés de liberté en total, trois degrés pour la rotation et trois pour la translation. Pour la composante rotationnelle du mouvement, une représentation sous forme d'une matrice de rotation orthogonale $R \in \mathbb{R}^{3\times3}$ est souvent utilisée (une autre représentation utilisant les quaternions est possible). La translation est représentée par un vecteur $T \in \mathbb{R}^3$. R et T peuvent être regroupés en une seule matrice de transformation rigide g telle que :

$$g = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{7.1}$$

avec :

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

et :

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}.$$

Cependant, la représentation d'une rotation par une matrice R n'est pas canonique puisque cette dernière possède 9 paramètres, alors qu'une rotation est décrite avec 3 degrés de liberté seulement. Ainsi les autres paramètres de la matrice R ne sont pas indépendants. Il existe une représentation minimaliste décrivant le mouvement d'un objet rigide en utilisant l'algèbre de Lie ([Ma *et al.*, 2003]). Cette représentation minimaliste est utile lorsqu'on cherche, par exemple, à déterminer les paramètres en utilisant une méthode d'optimisation numérique. En effet, chaque matrice de transformation décrivant le mouvement d'un objet rigide possède une représentation dans l'algèbre de Lie par le vecteur ξ de dimension 6×1 :

$$\xi = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}, \tag{7.2}$$

avec $v = (v_1, v_2, v_3)$ la vitesse linéaire et $w = (w_1, w_2, w_3)$ vitesse angulaire. La transformation g peut être calculée à partir de ξ en utilisant l'application exponentielle de l'algèbre de Lie :

$$g = e^{\hat{\xi}} = \begin{bmatrix} e^{\hat{w}} & (I - e^{\hat{w}}) \cdot \hat{w} \cdot v + w \cdot w^T \cdot v \\ 0 & 1 \end{bmatrix}, \tag{7.3}$$

avec $\hat{\xi}$ est la matrice anti-symétrique égale à :

$$\hat{\xi} = \begin{bmatrix} 0 & -w_3 & w_2 & v_1 \\ w_3 & 0 & -w_1 & v_2 \\ -w_2 & w_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{7.4}$$

7.2 Modèle de la caméra

Le modèle mathématique d'une caméra le plus utilisé est celui de *Pinhole* qui consiste à modéliser la caméra par un axe optique, un plan perpendiculaire à cet axe appelé plan focal ou plan d'image et un trou infiniment petit noté c situé sur le centre optique qui se trouve à l'intersection entre le plan et l'axe. La distance focale f est la distance entre le centre optique et le plan image. Ce modèle est illustré sur la figure 7.1. Tout rayon lumineux venant du monde 3D se projette sur le plan image en passant par le centre optique. Cette projection s'appelle une projection perspective et permet de passer d'un point 3D en un pixel dans l'image. Soit P cette transformation. Chaque point 3D de coordonnées (X, Y, Z) dans l'espace est lié à son pixel 2D (u, v) correspondant par l'équation suivante :

$$\begin{aligned} P : \mathbb{R}^3 \to \mathbb{R}^2; \quad (X, Y, Z) \to (u, v) \\ u = \frac{X \times f_x}{Z} + c_x \\ v = \frac{Y \times f_y}{Z} + c_y, \end{aligned} \tag{7.5}$$

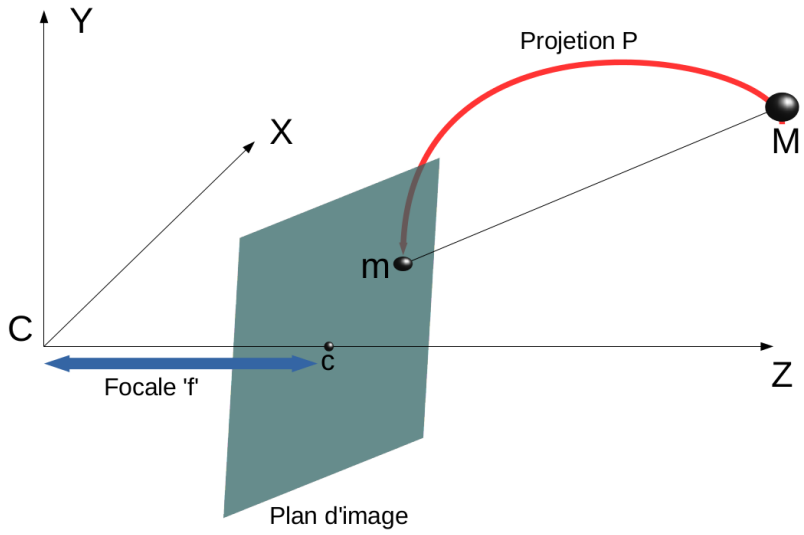


FIGURE 7.1 – Modèle simplifié de la caméra. Un point M dans l'espace 3D est projeté par la transformation P en un pixel m dans le plan image.

avec f_x, f_y, c_x et c_y sont respectivement la focale et le centre optique de la caméra. Cette projection n'étant pas bijective, c'est à dire, que connaissant un point 2D sur le plan image, il n'est pas possible de retrouver le point 3D correspondant avec une seule caméra. Si cependant, une caméra de type RGB-D est utilisée, il est possible de reconstruire le point 3D correspondant à un pixel grâce à la profondeur d encodée dans l'image de profondeur renvoyée par la caméra. Soit P^{-1} cette transformation qui permet de reconstruire le point 3D d'un pixel donné telle que :

$$
\begin{aligned}
P^{-1} : \mathbb{R}^2 \to \mathbb{R}^3; \quad (u, v, d) \to (X, Y, Z) \\
X &= \frac{u - c_x}{f_x} \times d \\
Y &= \frac{v - c_y}{f_y} \times d \\
Z &= d.
\end{aligned}
\tag{7.6}
$$

7.3 Méthode de moindres carrés

La méthode de moindres carrés ([Bjorck, 1996]) cherche à estimer les paramètres d'un modèle mathématique $f(x;\theta)$ avec $\theta = (\theta_1, \cdots, \theta_m)$ les paramètres inconnus et x les variables. En d'autres termes, ayant un ensemble d'observations $(y_i)_{i=1\cdots,n}$ bruitées, la méthode de moindres carrés cherche à trouver les meilleurs paramètres θ du modèle mathématique $f(x;\theta)$ qui décrit ces observations y_i. Pour trouver ces paramètres, elle minimise la somme quadratique des écarts entre la fonction mathématique et les observations :

$$E(\theta) = \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2 = \sum_{i=1}^{n} (r_i(\theta))^2. \tag{7.7}$$

Le minimum de cette fonction est obtenu lorsque la dérivée pa rapport à θ est égale à zéro :

$$\frac{\partial E(\theta)}{\partial \theta} = 2 \times \sum_{i=1}^{n} \frac{\partial r_i(\theta)}{\partial \theta} \cdot r_i(\theta) = 0. \tag{7.8}$$

Cas Linéaire

Lorsque la fonction $f(x_i, \theta)$ est linéaire en paramètres θ, la résolution est directe en utilisant le pseudo-inverse. En effet, $f(x_i, \theta)$ elle peut être écrite de la façon suivante :

$$f(x_i, \theta) = A(x_i) \cdot \theta,$$

avec $A(x_i)$ une matrice qui dépend uniquement de x_i. Dans ce cas là, le terme $\frac{\partial r_i(\theta)}{\partial \theta}$ sera égal à :

$$\frac{\partial r_i(\theta)}{\partial \theta} = A(x_i)^T.$$

En substituant la valeur de $\frac{\partial r_i(\theta)}{\partial \theta}$ dans l'équation 7.8, nous obtenons :

$$\sum_{i=1}^{n} A(x_i)^T \cdot (y_i - A(x_i)^T \cdot \theta) = 0.$$

En regroupant les termes de cette équation :

$$\sum_{i=1}^{n} A(x_i)^T \cdot A(x_i) \cdot \theta = \sum_{i=1}^{n} A(x_i)^T \cdot y_i.$$

En utilisant une représentation matricielle, nous obtenons :

$$A^T \cdot A \cdot \theta = A^T \cdot y.$$

La solution à cette équation est obtenue en multipliant par $(A^T \cdot A)^{-1}$ des deux côtés :

$$\boldsymbol{\theta} = (A^T \cdot A)^{-1} \cdot \ A^T \cdot y.$$

Cas Non Linéaire

Lorsque la fonction $f(x_i, \theta)$ est non linéaire en paramètres θ, une solution directe n'est pas possible. Il existe plusieurs méthodes pour résoudre le cas non linéaire. Dans la suite nous décrivons la méthode de Gauss-Newton. Cette méthode permet d'obtenir une solution itérative en linéarisant la fonction $r_i(\theta)$ dans le but d'obtenir une dépendance linéaire avec θ. Ainsi, à chaque itération k, $r_i(\theta_k)$ est linéarisée au voisinage de la solution de l'itération précédente $\theta = \theta_{k-1}$ en utilisant la série de Taylor de premier ordre telle que :

$$r_i(\theta_k)|_{\theta=\theta_{k-1}} \approx r_i(\theta_{k-1}) + \frac{\partial r_i(\theta)}{\partial \theta}|_{\theta=\theta_{k-1}} \cdot (\theta_k - \theta_{k-1}). \tag{7.9}$$

Avec $\frac{\partial r_i(\theta)}{\partial \theta}|_{\theta=\theta_{k-1}} = J_i(\theta_{k-1})$ est la matrice jacobienne et $\theta_k - \theta_{k-1}$ représente l'incrément noté $\Delta\theta$. Ainsi, l'équation 7.9 peut être écrite sous la forme suivante :

$$r_i(\theta_k)|_{\theta=\theta_{k-1}} = r_i(\theta_{k-1}) + J_i(\theta_{k-1}) \cdot \Delta\theta$$

En substituant la valeur de $r_i(\theta_k)$ et de $\frac{\partial r_i(\theta)}{\partial \theta}$ dans l'équation 7.8, nous obtenons :

$$\sum_{i=1}^{n} (J_i(\theta_{k-1})^T \cdot (r_i(\theta_{k-1}) + J_i(\theta_{k-1}) \cdot \Delta\theta)) = 0.$$

En développant et réarrangeant les termes, nous obtenons :

$$\sum_{i=1}^{n} J_i(\theta_{k-1})^T \cdot (J_i(\theta_{k-1}) \cdot \Delta\theta = -\sum_{i=1}^{n} J_i(\theta_{k-1})^T \cdot r_i(\theta_{k-1}). \tag{7.10}$$

Sous forme matricielle l'équation 7.10 s'écrit :

$$J(\theta_{k-1})^T \cdot J(\theta_{k-1}) \cdot \Delta\theta = -J(\theta_{k-1})\ r(\theta_{k-1}). \tag{7.11}$$

La solution à cette équation est obtenue en multipliant par le terme $(J(\theta_{k-1})^T \cdot J(\theta_{k-1}))^{-1}$ des deux côtés :

$$\boldsymbol{\Delta\theta} = -(J(\theta_{k-1})^T \cdot J(\theta_{k-1}))^{-1} \cdot J(\theta_{k-1})^T \cdot r(\theta_{k-1}), \tag{7.12}$$

avec $J(\theta_k)$ la matrice jacobienne de taille $n \times m$. Donc à chaque itération k, l'incrément $\boldsymbol{\Delta\theta}$ est d'abord calculé en utilisant l'équation 7.12, et la solution à l'itération k est obtenue en rajoutant $\boldsymbol{\Delta\theta}$ à la valeur de θ obtenue à l'itération $k-1$:

$$\theta_k = \theta_{k-1} + \boldsymbol{\Delta\theta}.$$

Ce processus itératif est répété un certains nombre de fois jusqu'à atteindre un seuil prédéfinit tel que : $E(\theta) \leq seuil$. La méthode de Gauss-Newton décrite ci-dessus converge vers le minimum local le plus proche et peut même diverger dans certain cas. Le point de départ θ_0 joue un rôle décisif dans ce processus itératif et doit être proche du minimum global pour que la méthode converge.

7.4 Ransac

En général, les méthodes de moindres carrés sont très sensible au bruit. En effet, si le bruit présent dans les données n'est pas gaussien, l'estimation des paramètres du modèle est mauvaise. Pour illustrer l'impact du bruit sur l'estimation finale, prenons l'exemple d'une régression linéaire simple comme sur la figure 7.2. Les points rouges correspondent à des points bruités appelés aussi points aberrants (en anglais *outlier*. Les points verts sont les points qui appartiennent au modèle (appelés *inliers* en anglais) . Si nous appliquons la méthode de moindre carrés pour trouver les paramètres de la droite en prenant les points vert et rouge, nous obtenons une mauvaise estimation représentée par la droite en bleu sur la figure, alors que la vrai droite qui représente le modèle est celle en noire. Donc les points aberrants ont une forte influence sur l'estimation finale et une méthode de moindres carrés est connue pour être très sensible aux bruits. Pour rendre robuste

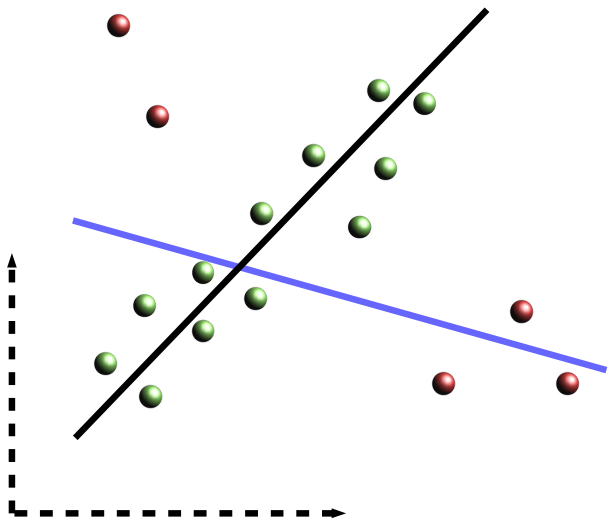


FIGURE 7.2 – Exemple d'une régression linéaire simple avec des données bruitées montrant la sensibilité de la méthode des moindres carrés aux points aberrants (en rouge). A cause de ces points, l'estimation finale est celle est en bleu, alors que la vraie estimée attendue est celle en noire.

cette méthode contre les *outliers*, l'algorithme RANSAC (*RANdom SAmple Consensus* [Fischler and Bolles, 1981]) est souvent utilisé pour éliminer d'une façon efficace ces points aberrants.

RANSAC est une méthode itérative utilisée pour estimer les paramètres d'un modèle mathématique à partir des données bruitées. Cette algorithme est non-déterministe, dans le sens où il garantie un résultat correct avec une certaine probabilité. L'algorithme prend en entrée l'ensemble des points, et retourne les paramètres du modèle mathématique et aussi l'ensemble des points aberrants retrouvés. L'algorithme consiste en deux étapes :

- Sélection : Dans cette étape, l'algorithme sélectionne d'une façon aléatoire n points, et génère une hypothèse sur le modèle à partir de ces n points. Cette hypothèse n'est qu'une estimation du paramètre du modèle.
- Évaluation : Dans cette étape, chaque hypothèse générée est évaluée en calculant le nombre de points qui s'ajustent à l'hypothèse dans le sens où la distance par rapport à l'hypothèse du modèle est inférieure à un seuil prédéfini.

C'est deux étapes sont répétées un certain nombre de fois et, à la fin, l'hypothèse qui possède le nombre le plus élevé de points qui s'ajustent à elle est retournée par l'algorithme. Cette meilleure hypothèse est à nouveau raffinée en recalculant les paramètres du modèle avec les points qui s'ajustent à cette hypothèse. Si dans l'étape de sélection, un ensemble ne contenant que des points pertinents a été sélectionné, l'hypothèse générée à partir de cet ensemble sera la meilleure. Donc dès que l'algorithme réussit à sélectionner un ensemble de points pertinents, il est sûr qu'il va converger. Maintenant la question qui se pose est : combien il faut d'itérations pour garantir que l'algorithme sélectionne un ensemble de points pertinents ? RANSAC permet de déterminer (d'une façon théorique) le nombre des itération K nécessaires pour obtenir au

moins un ensemble de n points pertinents. En effet, soit p la probabilité que l'algorithme choisit à une certaine itération que des points pertinents et w la probabilité pour qu'un point soit non pertinent. Alors $(1-w)^n$ est la probabilité que tous les n points sélectionnés sont pertinents. De la même façon, $1-(1-w)^n$ est la probabilité qu'au moins un des points sélectionné est aberrant. La probabilité que l'algorithme ne sélectionne que des points aberrants pour toutes les K itérations est : $(1-(1-w)^n)^K$ qui est égale à $1-p$. Nous avons alors :

$$1-p = (1-(1-w)^n)^K.$$

Ce qui permet au final de retrouver le nombre d'itérations nécessaires pour que l'algorithme converge :

$$K = \frac{\log(1-p)}{\log(1-(1-w)^n)}. \tag{7.13}$$

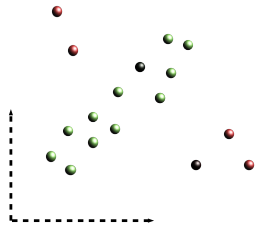
En pratique, w est inconnue à l'avance et une valeur approximative est souvent utilisée suivant la nature du problème. p détermine la probabilité que l'algorithme produit un résultat correct. Elle est souvent fixé à 0.99 ce qui se signifie que l'algorithme garantit à 99% qu'il sélectionne un ensemble de points pertinents. Pour le choix de n, généralement il est égal au nombre minimal de points nécessaire pour obtenir une hypothèse (pour le cas d'une régression linéaire $n = 2$), cependant Rosten *et al.* [Rosten *et al.*, 2010] ont démontré que dans certains cas où le bruit présent dans les données est important, les performances de l'algorithme (en terme de qualité d'estimation finale) peuvent être améliorées en choisissant un nombre plus grand que le nombre minimal nécessaire.

Un exemple de fonctionnement de l'algorithme RANSAC sur un problème de régression linéaire est illustré sur la figure 7.3 montrant les différentes étapes de l'algorithme et comment il est capable d'estimer correctement les paramètres du modèle et d'éliminer les points aberrants (en rouge). Sur cette figure, seulement deux itérations de l'algorithme sont présentées.

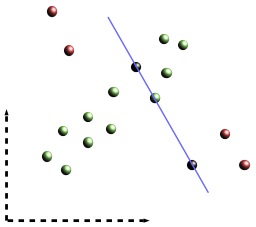
7.5 Conclusion

Dans ce chapitre, nous avons introduit les outils théoriques qui vont servir dans la suite pour développer la méthode de localisation visuelle à partir d'une caméra RGB-D dans un environnement dynamique. Dans un premier temps, Nous avons montré qu'il est possible de représenter le mouvement d'un objet rigide dans l'espace d'une façon minimaliste en utilisant l'algèbre de Lie. Cette représentation est utile pour le problème d'optimisation par moindre carrés et elle est très souvent utilisée dans la résolution de ce type de problème. Dans un deuxième temps, nous avons fait un rappel sur le modèle simplifié d'une caméra en décrivant les équations de projections perspectives. La méthode de moindres carrés a été aussi présentée, et nous avons montré que cette méthode est très sensible aux données bruitées et souvent l'algorithme RANSAC est utilisé pour rendre robuste cette méthode.

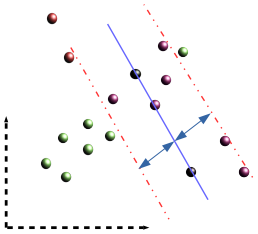
Dans le chapitre suivant, nous décrivons la méthode d'odométrie visuelle que nous avons développée.



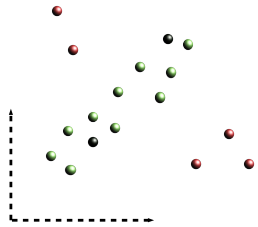
(a) Itération 1 : L'algorithme selectionne 2 points aléatoirement (en noire)



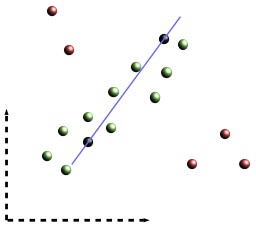
(b) Itération 1 : Une hypothèse sur le modèle est générée à partir de ces deux points (droite en bleu).



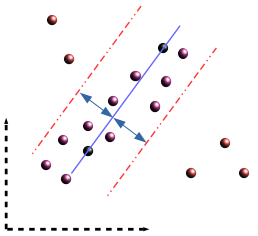
(c) Itération 1 : L'hypothèse est évaluée en comptant le nombre de points pertinents dont la distance par rapport à cette hypothèse est inférieure à un seuil prédéfinit (ici ce nombre est égale à 6).



(d) Itération 2 : L'algorithme selectionne à nouveau 2 points aléatoirement (en noire).



(e) Itération 2 : Une deuxième hypothèse est générée à partir de ces deux points (droite en bleu).



(f) Itération 2 : La nouvelle hypothèse est évaluée en comptant le nombre de points pertinents dont la distance par rapport à cette hypothèse est inférieure à un seuil prédéfinit (ici ce nombre est égale à 10).

FIGURE 7.3 – Exemple de fonctionnement de l'algorithme RANSAC sur un problème de régression linéaire qui montre la capacité de l'algorithme à estimer correctement les paramètres de la droite en éliminant les points aberrants (en rouge). Ici deux itérations de l'algorithme sont présentées sur chaque ligne. Après l'exécution des k itérations, l'algorithme renvoie l'hypothèse dont le nombre de points pertinents est le plus élevé.

Chapitre 8

Estimation robuste du mouvement de la caméra dans un environnement dynamique.

Sommaire

8.1 Formulation mathématique 130
8.1.1 Calcul de la Jacobienne 132
8.2 Pyramide multi-résolution 133
8.3 Estimation robuste du mouvement de la caméra 134
8.3.1 Principe de fonctionnement 135
8.3.2 Application de l'algorithme RANSAC 136
8.3.3 Nettoyage de la matrice jacobienne 137
8.3.4 Choix de nombre des pixels 139
8.3.5 Algorithme d'estimation robuste du mouvement de la caméra . . . 141
8.4 Conclusion 143

La méthode d'odométrie visuelle dense que nous avons présentée dans la section 6.2.2 cherche à estimer le mouvement de la caméra en minimisant l'erreur des intensités de tous les pixels de l'image. Cette minimisation est faite avec une méthode des moindres carrés. Or, nous avons vu dans le chapitre 7 que la méthode des moindres carrés est très sensible aux pixels aberrants. La cause principale de ces pixels est la dynamique de la scène, par exemple, une personne se déplaçant devant la caméra. Notre objectif étant d'estimer le mouvement de la caméra dans un environnement dynamique, il est alors nécessaire d'éliminer les pixels aberrants pour obtenir une estimation précise du mouvement de la caméra.

Dans ce chapitre, nous faisons d'abord un rappel sur le principe de fonctionnement de la méthode d'odométrie visuelle dense (sections 8.1 et 8.2). Ensuite, nous présentons, dans la section 8.3, les modifications que nous avons apportées à cette méthode pour la rendre robuste aux pixels aberrants présents dans l'image. La première originalité de notre méthode est d'abord, l'application de l'algorithme de RANSAC pour éliminer les pixels aberrants présents dans l'image. La deuxième originalité de la méthode proposée est l'introduction d'une étape de nettoyage de la matrice jacobienne afin d'améliorer sa précision.

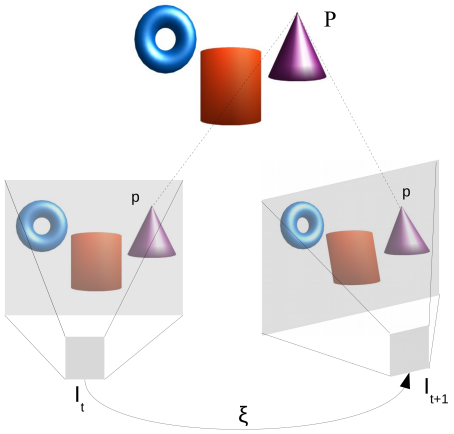


FIGURE 8.1 – L'odométrie visuelle dense repose sur le principe qu'un point P de l'espace observé par deux positions différentes de la caméra possède la même intensité lumineuse dans les deux images.

8.1 Formulation mathématique

L'odométrie visuelle dense consiste à estimer le mouvement d'une caméra mobile à partir d'une séquence d'images en utilisant tous les pixels de l'image. Elle repose sur l'hypothèse Lambertienne (en anglais *photo-consistency assumption*) illustrée sur la figure 8.1 qui est la suivante : un point 3D de l'espace observé par deux positions différentes (ou plusieurs) de la caméra possède la même intensité lumineuse dans les deux images. En d'autres termes, connaissant le mouvement ξ de la caméra entre les deux images, l'intensité du pixel p transformé par ξ dans l'image I_{t+1} est la même que celle dans l'image I_t. Cette hypothèse se traduit par l'équation suivante :

$$I_{t+1}(w(\xi, p)) = I_t(p), \tag{8.1}$$

où $I_t(p)$ et $I_{t+1}(w(\xi, p))$ sont les intensités d'un pixel p observées sur l'image reçue à l'instant t et à $t+1$ respectivement, et $w(\xi, p)$ étant la fonction de *warp* qui dépend du mouvement ξ de la caméra entre les deux images et qui permet de projeter chaque pixel d'une image à l'autre. Donc, théoriquement si la mouvement ξ de la caméra entre les deux images est parfaitement connue, l'erreur des intensités sur tous les pixels est nulle :

$$I_{t+1}(w(\xi, p)) - I_t(p) = 0.$$

Mais en réalité, cette erreur n'est jamais nulle à cause du bruit du capteur et les objets dynamiques présents dans la scène etc.... Cependant, cette erreur reste minimale connaissant la vrai transformation entre les deux images. En appliquant ce principe, les paramètres inconnues de ξ peuvent être calculés en minimisant l'erreur des intensités entre les deux images. Cela peut être formulé de la façon suivante :

$$\xi^* = \arg\min_{\xi} E(\xi) = \arg\min_{\xi} \left(I_{t+1}(w(\xi, p)) - I_t(p)\right)^2, \tag{8.2}$$

avec $E(\xi)$ est égale à :

$$E(\xi) = \sum_{i=1}^{N} \left(I_{t+1}(w(\xi, p_i)) - I_t(p_i)\right)^2 = \sum_{i=1}^{n} r_i(\xi)^2, \tag{8.3}$$

qui est la somme quadratique des erreurs des intensités de tous les N pixels p_i de l'image, $r_i(\xi)$ étant le résidu pour chaque pixel de l'image. Le modèle mathématique décrit ci-dessus nécessite de connaître l'intensité et la profondeur pour chaque pixel de l'image. Pour cela, un dispositif à deux caméras calibrées (calibration intrinsèque et extrinsèque) est nécessaire.

Nous avons vu dans la section 7.1 que pour un objet se déplaçant dans l'espace 3D (ici la caméra) il est possible de représenter son mouvement sous un forme minimaliste en utilisant l'algèbre de Lie. Cette représentation est utile pour résoudre le problème d'optimisation par moindres carrés puisqu'elle permet de minimiser la fonction 8.3 par rapport à un vecteur de 6 dimensions représentant la vitesse linéaire et angulaire de la caméra, et ensuite en utilisant l'application exponentielle (voir équation 7.3), il est possible de retrouver la transformation rigide (rotation + translation) décrivant le mouvement de la caméra entre les deux images. Ainsi la fonction 8.3 est minimisée par rapport à $\xi = \begin{bmatrix} v_1 & v_2 & v_3 & w_1 & w_2 & w_3 \end{bmatrix}$, avec (v_1, v_2, v_3) et (w_1, w_2, w_3) représentent la vitesse linéaire et angulaire de la caméra respectivement.

Linéarisation La fonction $r_i(\xi)$ de l'équation 8.3 n'est pas linéaire et amène ainsi à un problème de moindres carrés non linéaire. Nous avons vu dans la section 7.3 que ce problème peut être résolu en linéarisant le résidu de chaque pixel au voisinage de celui de l'itération précédente, ainsi la solution est obtenue d'une façon itérative en calculant à chaque fois un incrément $\Delta\xi$ qui sera cumulé à ξ_k tel que :

$$\xi_k = \xi_{k-1} + \Delta\xi,$$

et le résidu de chaque pixel à chaque itération k sera égale à :

$$r_i(\xi_k) = r_i(\xi_{k-1}) + J_i(\xi_k) \cdot \Delta\xi,$$

avec $J_i(\xi_k)$ étant la matrice jacobienne qui contient les dérivées partielles par rapport à ξ.

Construction de la fonction de *warp* La fonction de *warp* $w(\xi, p)$ permet de transformer chaque pixel d'une image à l'autre. Elle est composée d'un ensemble de transformations comme le montre la figure 8.2. Un pixel p de coordonnées (u, v, d) de l'image I_t est projeté en un point M en 3D de coordonnées (X, Y, Z) par la transformation P^{-1} de l'équation 7.6. M est ensuite transformé du repère attaché à I_t en un point M' de coordonnées (X', Y', Z') dans le repère de I_{t+1} par la transformation $g(\xi)$ de l'équation 7.3 . Finalement, M' est projeté dans le plan image de I_{t+1} par la transformation P de l'équation 7.5. Ainsi la fonction de *warp* s'écrit sous la forme suivante :

$$w(\xi, p) = P(g(\xi)(P^{-1}(p))). \tag{8.4}$$

Connaissant la forme de $w(\xi, p))$ il est maintenant possible de construire la jacobienne en se basant sur la série de transformations appliquées à chaque pixel comme décrit ci-dessus.

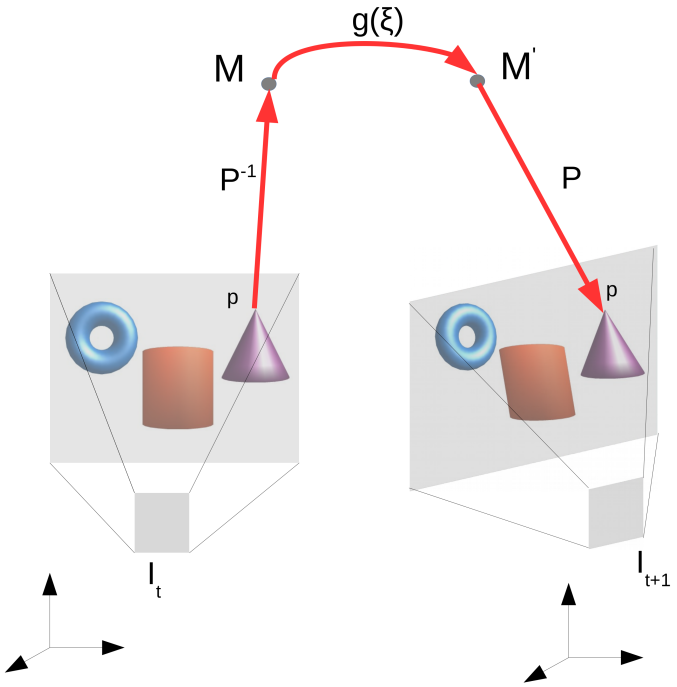


FIGURE 8.2 – La fonction de *warp* est composée d'un ensemble de transformations permettant de projeter chaque pixel de l'image I_t dans I_{t+1}.

8.1.1 Calcul de la Jacobienne

La matrice Jacobienne $J(\xi_k)$ est une matrice de taille $n \times 6$ de la forme suivante :

$$J(\xi_k) = \begin{bmatrix} \frac{\partial r_1}{\partial v_1} & \frac{\partial r_1}{\partial v_2} & \frac{\partial r_1}{\partial v_3} & \frac{\partial r_1}{\partial w_1} & \frac{\partial r_1}{\partial w_2} & \frac{\partial r_1}{\partial w_3} \\ \frac{\partial r_2}{\partial v_1} & \frac{\partial r_2}{\partial v_2} & \frac{\partial r_2}{\partial v_3} & \frac{\partial r_2}{\partial w_1} & \frac{\partial r_2}{\partial w_2} & \frac{\partial r_2}{\partial w_3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_i}{\partial v_1} & \frac{\partial r_i}{\partial v_2} & \frac{\partial r_i}{\partial v_3} & \frac{\partial r_i}{\partial w_1} & \frac{\partial r_i}{\partial w_2} & \frac{\partial r_i}{\partial w_3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_N}{\partial v_1} & \frac{\partial r_N}{\partial v_2} & \frac{\partial r_N}{\partial v_3} & \frac{\partial r_N}{\partial w_1} & \frac{\partial r_N}{\partial w_2} & \frac{\partial r_N}{\partial w_3} \end{bmatrix} = \begin{bmatrix} J_1(\xi_k) \\ J_2(\xi_k) \\ \vdots \\ J_i(\xi_k) \\ \vdots \\ J_N(\xi_k) \end{bmatrix},$$

avec N est le nombre de pixels dans l'image et 6 est le nombre de degrés de liberté (vitesse linéaire (v_1, v_2, v_3) et angulaire (w_1, w_2, w_3). Chaque ligne $J_i(\xi_k)$ de cette matrice représente la dérivée du résidu r_i du pixel p_i par rapport à ξ pour l'itération k. Rappelons que r_i est égal à l'erreur de l'intensité d'un pixel dans les deux images :

$$r_i(\xi) = I_{t+1}(w(\xi, p_i)) - I_t(p_i).$$

Ainsi, nous avons :

$$J_i(\xi_k) = \frac{\partial I_{t+1}}{\partial \xi}|_{\xi=\xi_k}.$$

En utilisant le théorème de dérivation des fonctions composées, nous pouvons écrire $J_i(\xi_k)$ sous la forme suivante :

$$J_i(\xi_k) = \nabla I_{t+1}(w(\xi_k, p)) \cdot J_w(\xi_k),$$

avec ∇I_{t+1} une matrice de 1×2 représentant le gradient d'un pixel dans l'image dans la direction x et y :

$$\nabla I_{t+1} = \left[\nabla I_{t+1,x}, \nabla I_{t+1,y}\right] \tag{8.5}$$

Le calcul détaillé de J_w se trouve dans l'annexe B. Au final, la matrice jacobienne finale est la suivante :

$$J_i(\xi_k) = \begin{bmatrix}\nabla I_{t+1,x} & \nabla I_{t+1,y}\end{bmatrix} \cdot \begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x \cdot X'}{Z'^2} & -\frac{f_x \cdot X' \cdot Y'}{Z'^2} & f_x + \frac{fx \cdot X'^2}{Z'^2} & -\frac{f_x \cdot Y'}{Z'} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y \cdot Y'}{Z'^2} & -f_y - \frac{fy \cdot Y'^2}{Z'^2} & \frac{f_y \cdot X' \cdot Y'}{Z'^2} & \frac{f_y \cdot X'}{Z'} \end{bmatrix}. \tag{8.6}$$

$J_i(\xi_k)$ dépend de la position du pixel transformé X', Y', Z' et ainsi doit être calculée à chaque itération. Backer et Matthews [Baker and Matthews, 2004] ont proposé une méthode appelée Inverse compositionnelle permettant de pré-calculer la jacobienne sur l'image de référence I_t qui sera constante tout au long de la minimisation, ce qui accélère considérablement la vitesse d'exécution de l'algorithme puisque cela évite de recalculer à chaque itération la matrice jacobienne de grande taille $n \times 6$. Ainsi en utilisant la méthode Inverse compositionnelle, la nouvelle jacobienne que nous notons J_i dans la suite devient :

$$J_i = \begin{bmatrix}\nabla I_{t,x} & \nabla I_{t,y}\end{bmatrix} \cdot \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x \cdot X}{Z^2} & -\frac{f_x \cdot X \cdot Y}{Z^2} & f_x + \frac{fx \cdot X^2}{Z^2} & -\frac{f_x \cdot Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y \cdot Y}{Z^2} & -f_y - \frac{fy \cdot Y^2}{Z^2} & \frac{f_y \cdot X \cdot Y}{Z^2} & \frac{f_y \cdot X}{Z} \end{bmatrix}. \tag{8.7}$$

Nous verrons dans la suite que le fait d'avoir une matrice jacobienne constante a un autre intérêt pour notre approche.

8.2 Pyramide multi-résolution

L'inconvénient des méthodes de minimisation itérative par moindres carrés est qu'elles ne sont pas capables de gérer les grands déplacements de la caméra. Pour pallier ce handicap, une approche multi-résolution, introduite par Burt et Adelson [Burt and Adelson, 1983], et utilisé par Comport *et al.* [Comport *et al.*, 2010] dans leur méthode, est souvent utilisée. Elle consiste à construire une pyramide composée d'un ensemble de M images à partir de l'image originale où chaque image de la pyramide est lissée et sous-échantillonnée par un facteur de deux. En effet, soit I_t^0 l'image au niveau 0 de la pyramide correspondant à l'image originale, l'image I_t^1 dans le niveau 1 de la pyramide est obtenue en sous-échantillonnant par un facteur de deux I_t^0 et en lissant l'image par un noyau gaussien. Le reste de la pyramide est construit de la même façon jusqu'au niveau $M-1$. Après construction de la pyramide, l'étape de minimisation par moindres carrés est appliquée en cascade sur les images de la pyramide en partant de l'image la plus petite I_t^{M-1}, l'estimation $\xi^{*(M-1)}$ trouvée au niveau $M-1$ est utilisée comme initialisation pour l'image au niveau suivant où le processus d'alignement est à nouveau appliqué et ainsi de suite jusqu'à atteindre l'image originale I_t^0. Ce nouveau processus d'estimation multi-résolution est illustré sur la figure 8.3.

L'avantage de l'utilisation d'une pyramide multi-résolution et que ça permet de lisser les grands mouvements effectués par la caméra. En effet, chaque image sous-échantillonnée est obtenue par un filtre passe bas, ainsi dans les images de petite résolution, très peu de détails existent et une estimation grossière est alors obtenue, en remontant dans la pyramide, les images contiennent plus de détails et l'estimation devient plus précise. De cette façon les grands mouvements de la caméra sont gérés progressivement. (Ce processus est appelé en anglais *Coarse to Fine*).

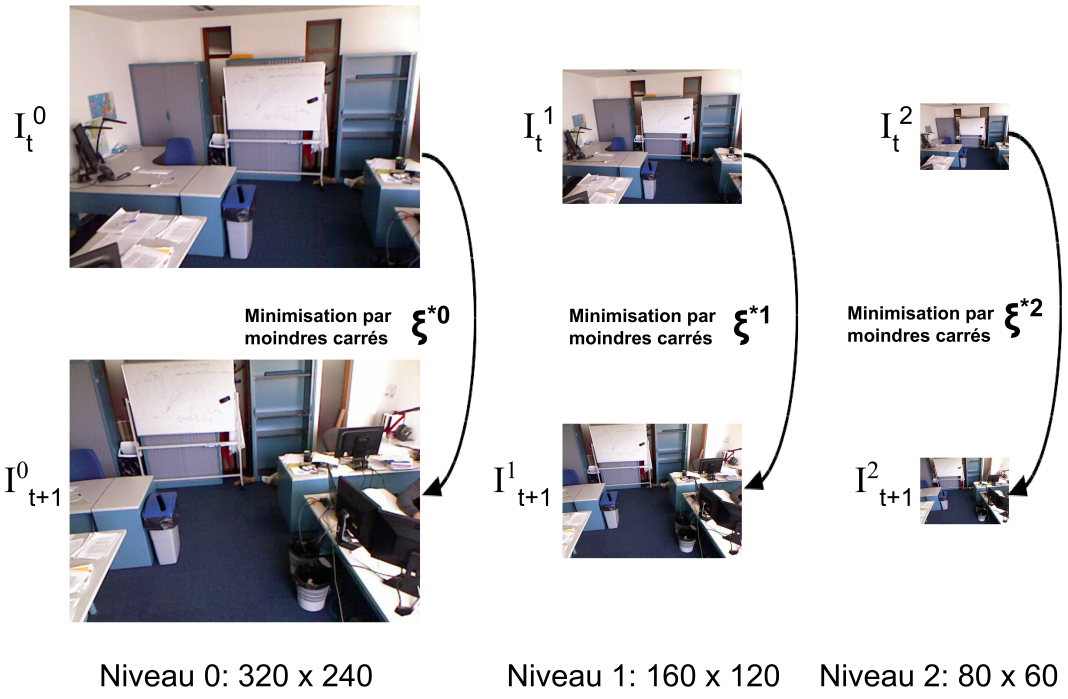


FIGURE 8.3 – Processus itérative d'alignement des images en utilisant une pyramide d'images multi-résolution.

8.3 Estimation robuste du mouvement de la caméra

Nous avons introduit le principe de fonctionnement de la méthode classique d'odométrie visuelle dense. Dans cette section nous montrons par une expérience son incapacité à estimer correctement le mouvement de la caméra lorsque des pixels aberrants sont présents dans l'image. En effet, la méthode de minimisation par moindres carrés décrite ci-dessus est sensible aux pixels aberrants pouvant influencer l'estimation finale. La cause principale de ces pixels est la dynamique de la scène (par exemple, une personne se déplaçant devant la caméra). Pour évaluer l'influence de ces pixels sur l'estimation finale, nous avons réalisé une expérience qui consiste à estimer le mouvement d'une caméra fixe dans une scène dynamique contenant une personne se déplaçant devant la caméra. Comme la caméra est fixe, sa position est toujours la même à chaque instant et correspond au $(0, 0, 0)$. Nous pouvons ainsi calculer, à chaque instant, l'erreur sur la position de la caméra obtenue par la méthode d'odométrie visuelle dense que nous avons décrite précédemment. La figure 8.4 montre la courbe d'erreur obtenue sur toute la séquence. Nous remarquons que l'erreur est très élevée et dépasse dans certains cas les 20 cm.

L'interprétation de ce résultat est la suivante : la méthode d'odométrie visuelle dense expliquée ci-dessus est une extension de la méthode de calcul de flux optique qui consiste à estimer la vitesse de déplacement des pixels dans l'image. Si tous les pixels de l'image appartiennent à des objets fixes de la scène, ainsi la vitesse estimée par la méthode d'odométrie visuelle correspond à celle de la caméra. Si cependant, certains pixels dans l'image correspondent à des objets mobiles de la scène, ces pixels lorsqu'ils sont pris en compte dans le processus d'estimation du déplacement de la caméra vont induire un mouvement supplémentaire dans l'estimée finale, ce qui explique que l'estimation du mouvement de la caméra dans l'expérience ci-dessus n'est pas

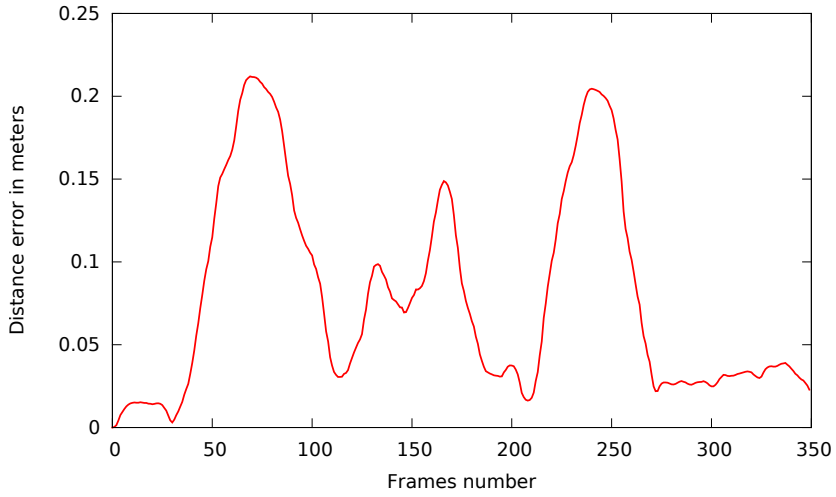


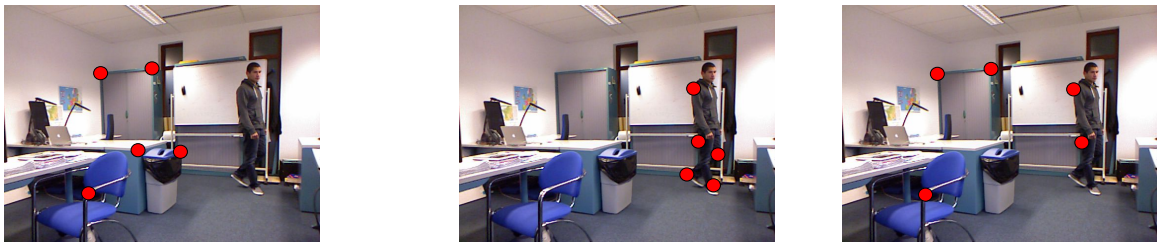
FIGURE 8.4 – Courbe d'erreur sur la position de la caméra.

correcte.

Cette expérience que nous avons réalisée montre que la présence des pixels aberrants dégrade la qualité de l'estimation finale puisqu'ils induisent un mouvement supplémentaire qui ne doit pas être pris en compte dans le processus d'estimation. Notre objectif, est d'estimer le déplacement d'une caméra mobile dans un environnement dynamique. Dans tel environnement, les objets mobiles de la scène produisent des pixels aberrants. Il est alors nécessaire de trouver un moyen pour réduire l'influence de ces pixels ou de les éliminer. Une méthode existante actuellement consiste à utiliser un processus de minimisation par moindres carrés pondérés qui assigne un poids $\in [0, 1]$ à chaque pixel. Ce poids représente la confiance donnée à chaque pixel. De cette façon, l'influence des pixels aberrants est réduite mais pas complètement. Idéalement, il sera plus intéressant d'éliminer complètement ces pixels. Pour cela, nous proposons par la suite une extension à la méthode d'odométrie visuelle dense capable de détecter et éliminer les pixels aberrants du processus de l'estimation du mouvement de la caméra.

8.3.1 Principe de fonctionnement

Une estimation correcte du mouvement de la caméra revient à calculer la vitesse des pixels non bruités appartenant à des objets fixes de la scène (ces pixels sont appelés pertinents). En d'autres termes, le mouvement des pixels non bruités appartenant à des objets fixes correspond en réalité au mouvement de la caméra. Donc, notre objectif qui est d'éliminer les pixels aberrants revient à sélectionner que les pixels pertinents pour l'estimation du mouvement de la caméra. Si nous choisissons un ensemble de pixels de l'image et nous calculons leur vitesse par la méthode d'odométrie visuelle que nous avons décrite ci-dessus, il existe 3 cas de figure (figure 8.5) : le premier cas (figure 8.5(a)), si tous les pixels choisis sont pertinents (appartiennent à des objets fixes de la scène et non bruités), alors la vitesse calculée à partir de ces pixels correspond à celle de la caméra. Le deuxième cas (figure 8.5(b)) lorsque tous les pixels choisis sont aberrants (appartiennent à la personne se déplaçant devant la caméra ou pixels bruités), alors la vitesse calculée à partir de ces pixels ne correspond pas à celle de la caméra. Le dernier cas (figure 8.5(c)) lorsque des pixels pertinents et aberrants sont sélectionnés, la vitesse calculée est difficile à prédire puisqu'elle dépend de la vitesse et du sens du déplacement des pixels aberrants et aussi de leur



(a) (b) (c)

FIGURE 8.5 – Suite à une sélection d'un ensemble de pixels de l'image, trois cas de figures se présentent : le premier lorsque tous les pixels sélectionnés sont pertinents, ainsi la vitesse de ces pixels correspond à celle de la caméra. Le deuxième et troisième cas est lorsque les pixels sélectionnés contiennent des pixels aberrants et dans ce cas là, il est très probable que la vitesse de ces pixels ne correspond pas à celle de la caméra.

nombre, mais il est très peu probable que cette vitesse calculée correspond à celle de la caméra. Ainsi, une estimation correcte du mouvement de la caméra revient à sélectionner un ensemble de pixels pertinents et de calculer leurs vitesse qui sera celle de la caméra. La question qui se pose est comment choisir de l'image que les pixels pertinents. Nous avons présenté dans la section 7.4 l'algorithme RANSAC qui permet d'estimer les paramètres d'un modèle mathématique à partir des données bruitées. Cet algorithme procède d'une façon itérative sur l'ensemble des données et à chaque itération il sélectionne aléatoirement un ensemble de points et calcule ensuite une hypothèse sur le modèle qui sera évaluée en calculant le nombre de points dont la distance par rapport à l'hypothèse est inférieure à un seuil prédéfini (voir la figure 7.3). Ce processus est répété un certain nombre de fois, et l'hypothèse retenue est celle qui possède le nombre le plus élevé de points qui s'ajustent correctement à elle. Il est possible avec RANSAC de déterminer le nombre des itérations K nécessaire (voir l'équation 7.13) pour garantir que l'algorithme sélectionne au moins un ensemble de points pertinents. Ainsi, pour rendre robuste la méthode d'odométrie visuelle contre les points aberrants, notre idée consiste à appliquer l'algorithme de RANSAC puisqu'il permet de faire exactement ce que nous souhaitons.

8.3.2 Application de l'algorithme RANSAC

Dans cette partie nous décrivons le nouvelle méthode que nous avons conçue qui utilise l'algorithme RANSAC pour estimer d'une façon robuste le mouvement de la caméra. Le fonctionnement est le suivant : Soit K le nombre des itérations nécessaires pour que l'algorithme RANSAC converge (nous détaillons dans la suite comment calculer K). A chaque itération j ($j \leq K$), un ensemble noté O_j de n pixels est choisi d'une façon aléatoire de l'image. A partir de cet ensemble, une fonction de coût $E_j(\xi)$ est minimisée pour obtenir une hypothèse $\hat{\xi}_j^*$ sur le déplacement de la caméra. Cette fonction est similaire à celle de l'équation 8.3 avec la seule différence qu'elle est appliquée sur les n pixels et non pas sur tous les pixels N de l'image (avec $n < N$) :

$$E_j(\xi) = \sum_{i=1}^{n} (I_{t+1}(w(\xi, p_i)) - I_t(p_i))^2 = \sum_{i=1}^{n} r_i(\xi)^2. \tag{8.8}$$

$\hat{\xi}_j^*$ est obtenue telle que :

$$\hat{\xi}_j^* = \arg\min_{\xi} E_n(\xi). \tag{8.9}$$

L'hypothèse $\hat{\xi}_j^*$ sur le déplacement de la caméra est évaluée en alignant d'abord les deux images I_t et I_{t+1} par $\hat{\xi}_j^*$, et ensuite à calculer l'image résultante R_j de la soustraction des deux images alignées. Finalement, pour évaluer cette hypothèse, nous devons connaître le nombre de pixels qui ont été alignés correctement par cette hypothèse. La valeur absolue de chaque pixel r_i avec $i \in [1 \cdots N]$ de l'image R_j est comparée à un seuil t prédéfini par l'algorithme, et si $R_j(r_i) \leq t$, le pixel sera ajouté à l'ensemble S_j associé à cette hypothèse (S_j est appelé *consensus set* en anglais). Ce processus est répété K fois, en calculant et évaluant à chaque itération, une hypothèse générée à partir d'un ensemble de n pixels sélectionnés aléatoirement de l'image. A la fin, l'hypothèse $\hat{\xi}_{best}^*$ retenue par l'algorithme est celle telle que :

$$Card(S_{best}) > Card(S_j), \text{ pour tout} j \in [1 \cdots K]\,.$$

A partir de $\hat{\xi}_{best}^*$ et de son ensemble de pixels alignés correctement S_{best}, l'estimation $\hat{\xi}_{best}^*$ est raffinée en recalculant une nouvelle estimation ξ_{best}^* du mouvement de la caméra à partir de l'ensemble de points dans S_{best} en minimisant la fonction $E(\xi)$ pour tous les points appartenant à S_{best}.

8.3.3 Nettoyage de la matrice jacobienne

Dans cette section, nous décrivons la méthode que nous avons développée pour améliorer l'étape de sélection des n pixels. En effet, il existe des endroits dans l'image dont les pixels ne rapportent pas des informations sur le mouvement de la caméra. Ainsi l'algorithme décrit ci-dessus risque de générer des hypothèses non pertinentes si les pixels sélectionnés appartiennent à ces endroits. Dans cette partie nous présentons la méthode que nous avons développé pour améliorer l'étape de sélection des pixels basée sur une analyse directe de la jacobienne.

Reprenons la jacobienne de l'équation 8.7 calculée pour chaque pixel d'indice i de l'image. Nous avons vu que grâce à la méthode d'inverse compositionnelle (section 8.1.1), cette matrice est pré-calculée sur l'image de référence et elle est constante tout au long de la minimisation. Cette matrice que nous notons par la suite J_i est le produit des deux matrices J_p et J_G tel que :

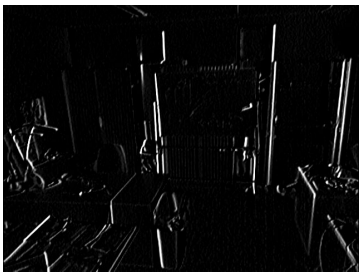
$$J_i = J_p \cdot J_G, \tag{8.10}$$

avec J_i une matrice de 1×6 tel que :

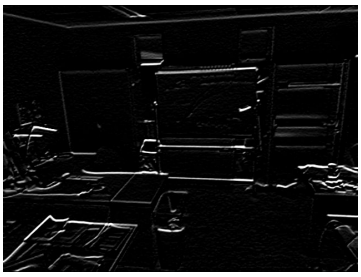
$$J_i = \begin{bmatrix} J_i^1 & J_i^2 & J_i^3 & J_i^4 & J_i^5 & J_i^6 \end{bmatrix},$$

et J_p le gradient photométrique qui est la matrice à gauche de l'équation 8.7 et J_G représentant le gradient géométrique à droite de l'équation 8.7. Sur la figure 8.6, le gradient sur x et y pour une image est affichée. Certains pixels de cette image possèdent un gradient faible ou proche de zéro (représenté par une intensité en noire sur les figures 8.6(b) et 8.6(c)). Ces pixels corresponds généralement à des régions uniformes dans l'image très peu texturées (comme un mur par exemple). Ces pixels ne sont pas utiles pour l'estimation du mouvement de la caméra puisque la jacobienne J_i associée à chacun de ces pixels est nulle. Ainsi dans notre algorithme, il faut éviter de sélectionner des pixels de ces régions sinon les hypothèses produites risquent d'être non pertinentes. Une façon simple de le faire, est qu'à chaque fois un pixel est choisi aléatoirement

(a)



(b)



(c)

FIGURE 8.6 – a) Image originale b) Gradient sur X. b) Gradient sur Y. Les endroits uniformes dans la scène possède un faible gradient proche de zéros (pixels en noirs). Les pixels appartenant à ces endroits ne sont pas utiles pour l'estimation du mouvement de la caméra puisque la jacobienne correspondante est nulle (voir équation 8.7).

FIGURE 8.7 – Un exemple illustrant l'importance de l'information géométrique représentée par la matrice J_G (voir équation 8.10). Bien que les pixels lointains possèdent un gradient photométrique élevé (en rouge), ils décrivent moins précisément la translation de la caméra que les pixels proches (en bleu).

parmi les n pixels, nous pouvons le rejeter si son gradient est faible pour en sélectionner un nouveau. L'utilisation du gradient seul comme critère pour valider ou non un pixel n'est pas suffisant puisque il ne prend pas en compte l'information géométrique encodée dans la matrice J_G qui représente le mouvement d'un pixel par rapport aux 6 degrés de liberté, et chaque élément de J_G correspond à un degré de liberté de ξ :

$$J_G = \begin{bmatrix} J_G^1 & J_G^2 & J_G^3 & J_G^4 & J_G^5 & J_G^6 \end{bmatrix}.$$

Pour bien illustrer l'importance de l'information géométrique représentée par la matrice J_G, prenons l'exemple de la figure 8.7 où la zone, marquée en bleue, correspond à des pixels (appartenant au sol) peu texturé et proche de la caméra et la zone en rouge correspond à des objets fortement texturés mais loin de la caméra. Le fait de n'utiliser que le gradient photométrique comme critère pour la sélection des pixels, va amener l'algorithme à favoriser ceux qui appartiennent à la région loin de la caméra. Ainsi, l'estimation du mouvement de la caméra à partir des pixels éloignés est moins précise sur la translation puisque ces pixels sont peu sensibles au mouvement translationnel et décrivent moins précisément la translation que les pixels proches (voir la figure 8.8 pour un exemple d'illustration). En pratique, pour une caméra de profondeur d'une portée de quelques mètres, ce phénomène d'insensibilité des pixels à un certain type de mouvement est réduit mais il est toujours présent, alors que pour une caméra de grande portée, ce phénomène est amplifié. Ainsi, avant de sélectionner les pixels de l'image, nous introduisons une étape de nettoyage de la jacobienne J consistant à retirer tous les lignes J_i dont la valeur d'au moins une colonne est inférieure à un seuil noté s. De cette façon seuls les pixels qui possèdent une bonne observabilité de chaque degré de liberté sont gardés et la sélection des pixels est faite à partir de cette nouvelle matrice J nettoyée. Notons que cette étape de nettoyage est faite une seule fois sur la matrice jacobienne pré-calculée sur l'image de référence.

8.3.4 Choix de nombre des pixels

Le nombre de pixels à sélectionner joue un rôle important dans la détermination du nombre des itérations K (voir équation 7.13) nécessaire pour que l'algorithme RANSAC converge. Dans cette partie nous décrivons comment nous avons choisi ce paramètre.

La méthode d'odométrie visuelle dense est une extension de la méthode de calcul de flux optique (aussi appelée *Lucas-Kanade* [Lucas and Kanade, 1981]), ainsi elle hérite d'elle une

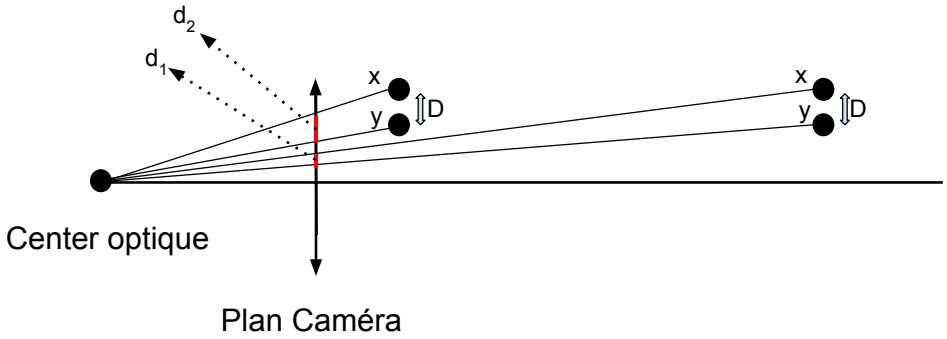


FIGURE 8.8 – Un exemple 1D illustrant comment les pixels éloignés ne décrivent pas précisément le mouvement de translation de la caméra. Pour une même déplacement D d'un objet d'un point x vers y, la zone occupée par l'image lorsque l'objet est loin (d_1) est inférieure que celle lorsqu'il est proche (d_2). Si l'objet est à l'infini, son déplacement D n'est pas observée sur le plan caméra.

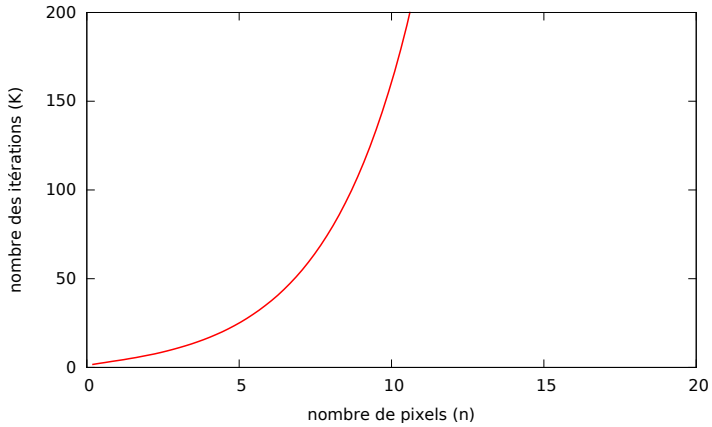


FIGURE 8.9 – Le nombre des itérations K nécessaire pour que l'algorithme RANSAC converge varie d'une façon exponentielle en fonction de n (voir équation 7.13).

hypothèse que nous allons discuter dans la suite. En effet dans la méthode de *Lucas-Kanade*, pour calculer la vitesse de déplacement d'un pixel, elle suppose que le flux est constant dans un voisinage local du pixel pour pouvoir résoudre l'équation du flux optique. Pour illustrer pourquoi les auteurs avaient besoin de cette hypothèse, prenons l'exemple de calcul de la vitesse de déplacement d'un pixel en 2D. D'après l'hypothèse Lambertienne nous avons :

$$I(x, y, t) = I(x + dx, y + dy, t + dt), \tag{8.11}$$

Avec (x, y) les coordonnées d'un pixel dans l'image. En supposant que le déplacement est relativement petit entre les deux images, $I(x + dx, y + dy, t + dt)$ peut être développée en série de Taylor :

$$I(x + dx, y + dy, t) = I(x, y, t) + \frac{\partial I}{\partial x} \cdot dx + \frac{\partial I}{\partial y} \cdot dy + \frac{\partial I}{\partial t} \cdot dt, \tag{8.12}$$

En remplaçant l'équation 8.12 dans 8.11 et en divisant par dt, nous obtenons :

$$\begin{aligned} \frac{\partial I}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I}{\partial t} &= 0 \\ \Rightarrow \nabla I_x \cdot v_x + \nabla I_y \cdot v_y + \nabla I_t &= 0, \end{aligned} \tag{8.13}$$

Avec $\nabla I_x = \frac{\partial I}{\partial x}$ et $\nabla I_y = \frac{\partial I}{\partial y}$ sont respectivement les gradients de l'image dans la directions x et y. v_x et v_y sont les deux inconnues qui représentent la vitesse de déplacement du pixel. Ainsi il n'est pas possible de calculer cette vitesse avec une seule équation. Pour résoudre cette équation, les auteurs supposent que les pixels voisins de p dans une fenêtre de 3×3 possèdent la même vitesse. C'est grâce à cette hypothèse que le système devient sur-dimensionné et ainsi il est possible d'estimer v_x et v_y par moindres carrés.

Dans notre problème, lorsque nous tirons aléatoirement n pixels avec les voisins pour estimer leur vitesse, nous obtenons $n \times 9$ pixels au final. Par contre le choix d'un nombre très élevé de pixels pour la génération des hypothèses nécessite plus d'itérations par RANSAC. En effet, sur la figure 8.9 est affiché la courbe de variation de K en fonction de n à partir de l'équation 7.13 pour des valeurs de $p = 0.99$ et $w = 0.3$. Nous voyons sur cette courbe que K varie d'une façon exponentielle avec le nombre n de pixels, ce qui est normal, puisque lorsque n devient grand, l'algorithme a besoin de plus d'itérations pour trouver un ensemble de pixels pertinents. Ainsi, lorsque nous tirons par exemple 5 pixels de l'image, nous obtenons, en prenant en compte les voisins, $5 \times 9 = 45$ pixels. Or pour $n = 45$ nous avons un nombre d'itérations qui vaut 43036194, ce qui n'est pas applicable en réalité. L'hypothèse que la vitesse des voisins d'un pixel est la même, se traduit par le fait que si un pixel est pertinent (ou non), ses voisins le sont aussi. Ainsi, l'algorithme n'a pas besoin de ce nombre élevé d'itérations puisque au final la vitesse de ces 45 pixels n'est que la vitesse des 5 pixels situés au centre. Donc, dans l'exemple ci-dessus, n vaut en réalité 5 et non pas 45 et K dans ce cas-là vaut 25 itérations. Ainsi, grâce à l'hypothèse qui suppose que le flux dans un voisinage d'un pixel est constant, nous allons pouvoir tirer un nombre $n \times 9$ de pixels de l'image tout en obtenant un nombre d'itérations raisonnable.

8.3.5 Algorithme d'estimation robuste du mouvement de la caméra

Nous pouvons maintenant décrire l'algorithme complet que nous avons conçu pour estimer d'une façon robuste le mouvement de la caméra. La figure 8.10 représente les différentes étapes de l'algorithme. A partir de l'image I_t, la matrice Jacobienne J est calculée sur cette image, ensuite pour chaque ligne J_i de cette matrice, si au moins la valeur d'une colonne est inférieure à un

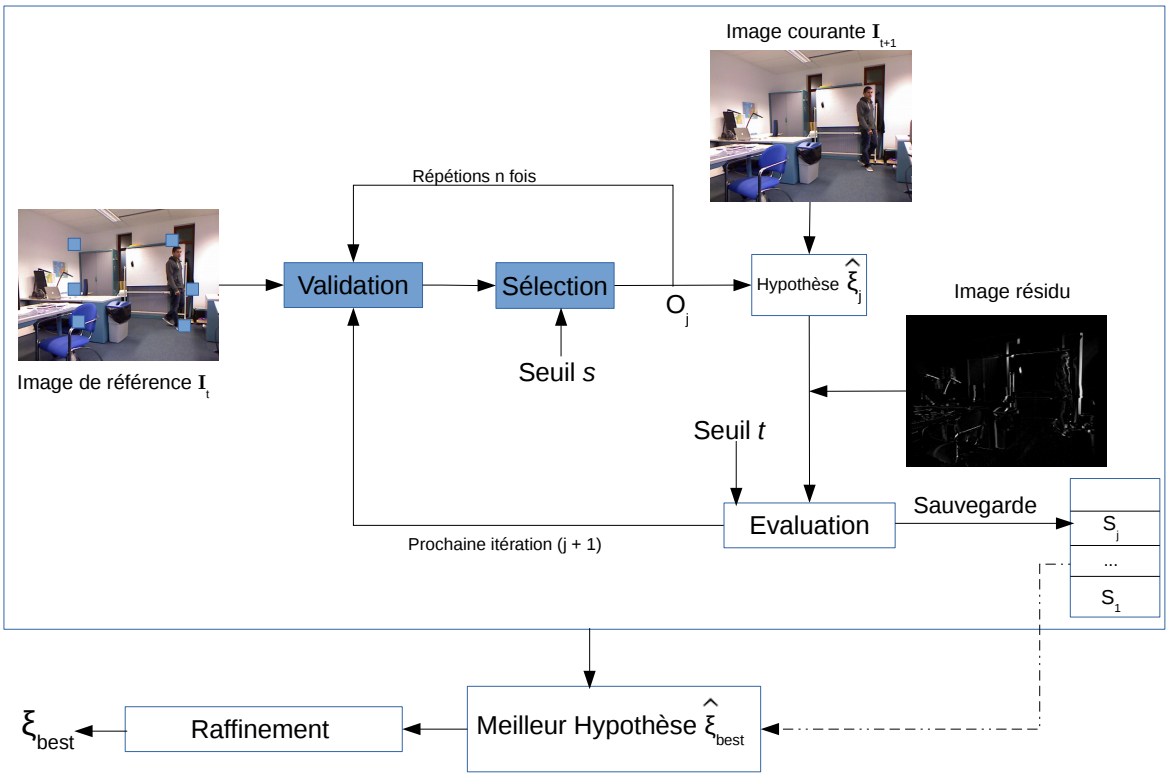


FIGURE 8.10 – Processus de minimisation robuste par RANSAC.

certain s, cette ligne sera retirée de la matrice J, de cette façon seuls les pixels qui conditionnent bien les degrés de liberté sont choisis. Ensuite l'algorithme procède d'une façon itérative en sélectionnant à chaque itération j un ensemble de n pixels pour construire l'ensemble O_j. Une hypothèse $\hat{\xi}_j$ sur le déplacement de la caméra est ensuite obtenue, cette hypothèse sera évaluée en alignant les deux images I_t et I_{t+1} par $\hat{\xi}_j$ et en calculant l'image résidu R_j correspondante. Pour chaque pixel dans R_j si sa valeur est inférieure à un seuil t, il sera ajouté dans l'ensemble S_j. A la fin des K itérations. La meilleure hypothèse $\hat{\xi}_{best}$ dont le cardinal est le plus élevé est renvoyée par l'algorithme. Finalement, cette hypothèse est raffinée en calculant à nouveau l'estimation finale de la caméra en utilisant les pixels dans S_{best}.

Il arrive dans certaines situations, lorsque les données sont très peu bruitées, qu'une hypothèse donnée possède dans son ensemble S_j un nombre très élevés de pixels, ce qui signifie que la majorité des pixels présents dans l'image ont bien été alignés par cette hypothèse. Dans ce cas là, il n'est pas utile de continuer les itérations puisque l'algorithme a réussit à trouver une bonne hypothèse. Ainsi, si au cours des itérations j, une hypothèse dont le rapport des pixels de son ensemble S_j par rapport au nombre total de pixels N est supérieure à $(1 - w)$ (avec w est la probabilité qu'un pixel soit aberrant), l'algorithme s'arrête et renvoie cette hypothèse comme la meilleure retrouvée. Une dernière remarque est que le processus décrit sur la figure 8.10 est répété pour chaque image de la pyramide.

8.4 Conclusion

Dans ce chapitre, nous avons d'abord présenté la méthode d'odométrie visuelle dense classique qui utilise une méthode de minimisation par moindres carrés pour minimiser l'erreur des intensités de tous les pixels dans l'image. Nous avons montré avec une expérience que cette méthode est très sensible aux pixels aberrants présents dans l'image. Ensuite, nous avons décrit les modifications que nous avons apportées à cette méthode pour la rendre robuste à ces pixels aberrants. La première modification consiste à utiliser l'algorithme itératif RANSAC pour éliminer les pixels aberrants. Cet algorithme génère des hypothèses sur le mouvement de la caméra à partir d'un ensemble de pixels sélectionnés aléatoirement de l'image, et choisit celle qui décrit le mieux le déplacement de la caméra. Dans le but d'améliorer la qualité des hypothèses générées par RANSAC, nous avons introduit une deuxième modification qui consiste à travailler directement sur la matrice jacobienne et de la nettoyer en retirant les pixels qui ne rapportent pas d'information sur le mouvement de la caméra.

Dans le chapitre suivant, nous procédons à une série d'expérimentation pour évaluer notre approche.

Chapitre 9

Evaluation

Sommaire

9.1 Evaluation de la capacité de notre méthode à éliminer les pixels aberrants 145
9.1.1 Caméra fixe dans un environnement dynamique 145
9.1.2 Caméra montée sur un motor pan-tilt 148
9.1.3 Résultats qualitatifs 148
9.2 Évaluation avec les données du Benchmark 149
9.3 Influence des paramètres sur les performances 151
9.3.1 Seuil pour le nettoyage de la jacobienne 151
9.3.2 Choix du seuil 151
9.3.3 Nombre de pixels 152
9.4 Vitesse d'exécution 152
9.5 Conclusion 152

Dans ce chapitre, nous montrons, dans un premier temps, avec une série d'expériences que nous avons réalisée, la capacité de notre méthode à éliminer d'une façon efficace les pixels aberrants présents dans l'image. Nous montrons aussi, que notre méthode améliore la qualité de l'estimation du mouvement de la caméra par rapport à la méthode classique d'odométrie visuelle dense dans des scènes dynamiques.

Dans un deuxième temps, nous utilisons un *benchmark* en ligne pour évaluer les performances de notre méthode et pour la comparer à une autre méthode de la littérature.

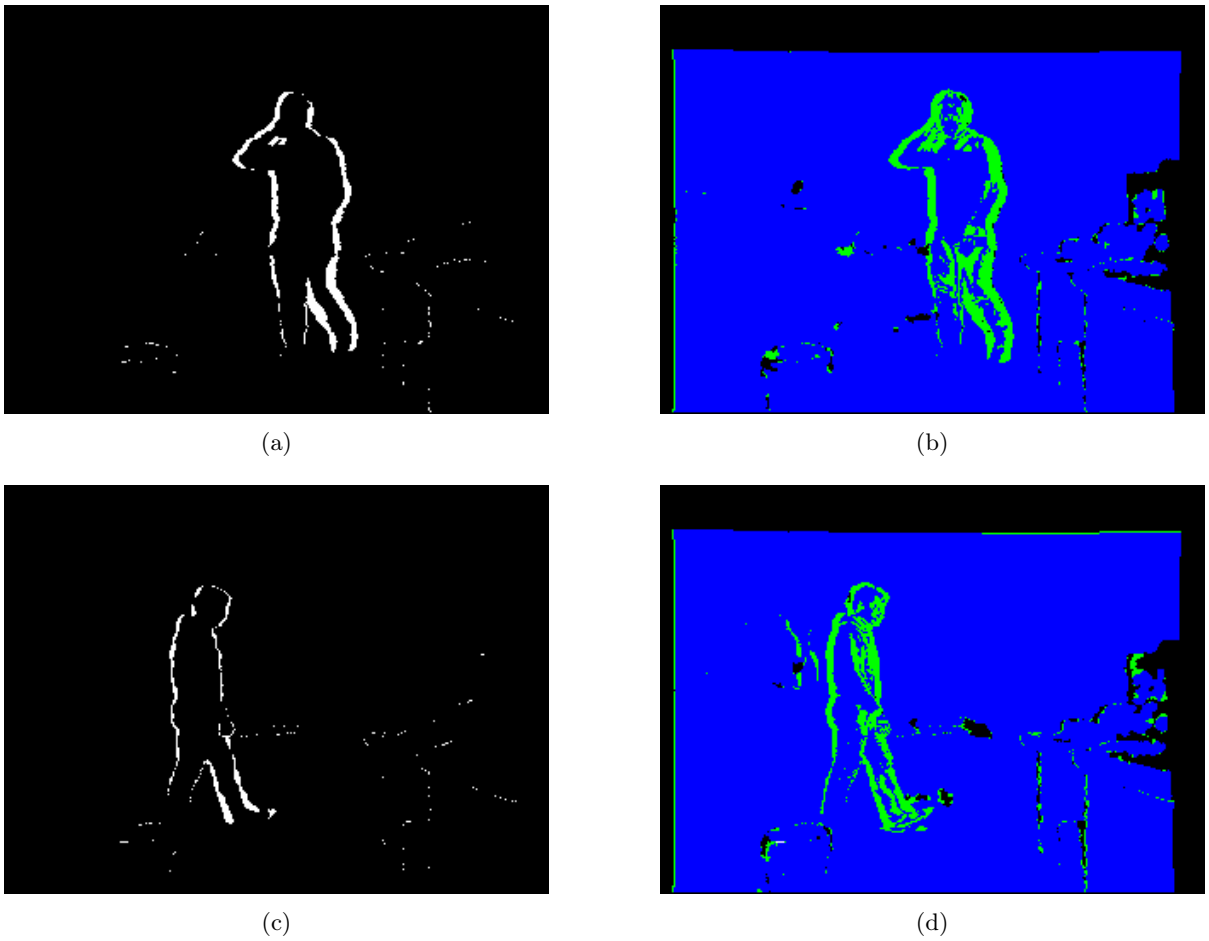
Finalement, nous évaluons l'influence des différents paramètres sur les performances de notre méthode.

9.1 Evaluation de la capacité de notre méthode à éliminer les pixels aberrants

L'objectif de cette partie est d'abord d'évaluer la capacité de notre méthode à éliminer les pixels aberrants de l'image, et aussi d'évaluer de combien notre approche améliore la qualité de l'estimation par rapport à la méthode classique d'odométrie visuelle dense.

9.1.1 Caméra fixe dans un environnement dynamique

Nous avons réalisé une première expérience consistant à filmer avec une Kinect fixe une scène dynamique contenant une personne se déplaçant devant la caméra. Comme la caméra est fixe, les

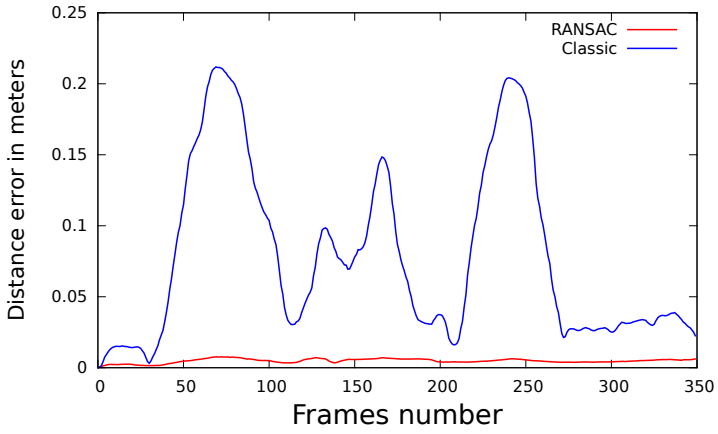


(a) (b)

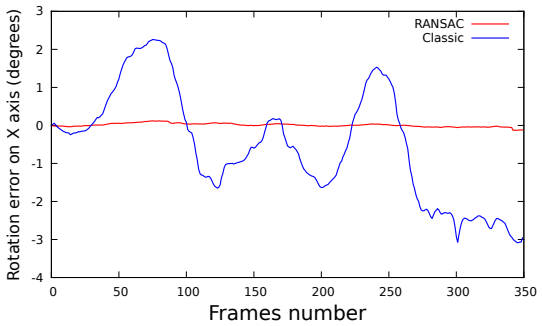
(c) (d)

FIGURE 9.1 – a) et c) : Le résultat de soustraction de deux images consécutives. b) et d) : Les pixels pertinents (en bleu) et aberrants (en vert) retournés par notre méthode.

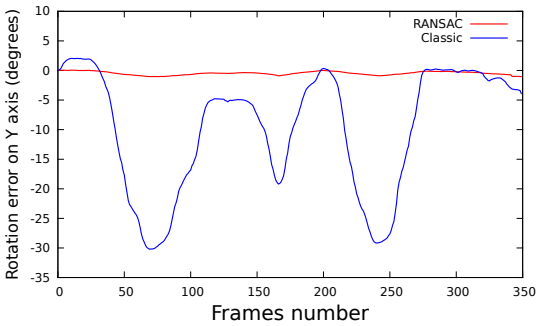
pixels correspondant au mouvement de la personne sont obtenus par une simple soustraction des deux images consécutives (les pixels en blanc sur les figures 9.1(a) et 9.1(c)). Nous avons ensuite lancé notre méthode sur cette séquence et les figures 9.1(b) et 9.1(d) montrent les pixels pertinents (bleu) et aberrants (vert) détectés par notre méthode. Ces figures montrent que notre méthode réussit à distinguer correctement entre les pixels aberrants (qui correspond au mouvement de la personne) et les pixels pertinents. Sur la figure 9.2(a) est affichée la courbe d'erreur sur la position de la caméra obtenue par notre méthode et celle obtenue par la méthode d'odométrie visuelle classique (La position de la caméra correspond au $(0, 0, 0)$ puisque elle est fixe). Sur cette figure, nous voyons que notre approche a une dérive très faible par rapport à la méthode originale qui à une erreur très élevée et une dérive qui dépasse les 20 cm. Les figures 9.2(b) et 9.2(c) montrent la courbe d'erreur sur l'orientation de la caméra pour les axes X et Y. L'erreur d'orientation par la méthode classique est très élevée et dépasse parfois les 30°, alors que notre approche a une faible erreur qui reste inférieure à 1° tout au long de la séquence.



(a) Courbe d'erreur sur la position de la caméra.

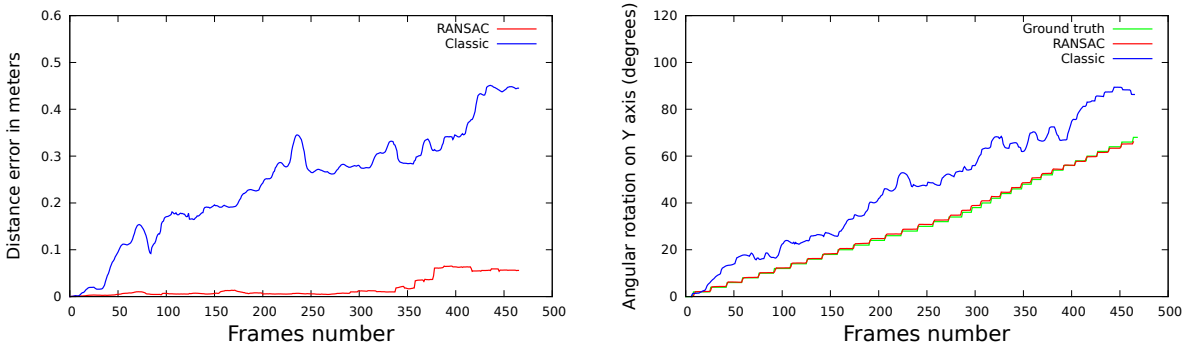


(b) Courbe d'erreur sur l'orientation de la camera sur l'axe horizontal x.



(c) Courbe d'erreur sur l'orientation de la camera sur l'axe vertical y.

FIGURE 9.2 – Courbe d'erreur sur la position et l'orientation de la caméra obtenue par notre méthode (en rouge) et la méthode d'odométrie visuelle classique (en bleu) pour une caméra fixe filmant une personne se déplaçant dans la scène.



(a) (b)

FIGURE 9.3 – Courbe d'erreur sur la position et l'orientation d'une caméra montée sur un pan-tilt effectuant une rotation sur l'axe vertical. a) Courbe d'erreur sur la position de la caméra obtenue par notre méthode (en rouge) et la méthode d'odométrie visuelle classique (en bleu). b) L'estimation de l'orientation de la caméra sur l'axe vertical par les deux méthodes.

9.1.2 Caméra montée sur un motor pan-tilt

Une autre expérience que nous avons réalisée consiste à monter une Kinect sur un moteur pan-tilt de type Biclops, fabriqué par la société Traclabs, qui effectue une rotation de 90 degrés (avec un pas de 2 degrés chaque seconde) sur l'axe vertical dans une scène avec une personne se déplaçant dedans. La figure 9.3(a) compare l'erreur sur la position de la caméra obtenue par notre approche (en rouge) et celle obtenue par la méthode classique (en bleu). A cause des pixels aberrants générés par le mouvement de la personne devant la caméra, La dérive de la méthode originale continue a augmenter et dépasse à la fin de la séquence 40 cm, tandis que notre méthode réussit à maintenir une dérive faible. Sur la figure 9.3(b) est affiché en vert l'orientation du Biclops sur l'axe vertical obtenue par l'odométrie interne très précise du Biclops. L'orientation de la caméra estimée par notre méthode (en rouge) est très proche de la courbe en vert, alors que celle estimée par l'approche originale (en bleu) possède un écart très important.

9.1.3 Résultats qualitatifs

L'objectif de cette expérience est de valider qualitativement la capacité de notre approche à éliminer les pixels aberrants du processus de l'estimation de mouvement. Pour cela, l'expérience consiste à faire bouger la caméra à la main dans une scène qui contient une personne se déplaçant et effectuant des mouvements rapides devant la caméra. Les images sur la première colonne de la figure 9.4 (page 154) montrent la personne qui se déplace, saute et qui lance une balle. Le deuxième colonne de la figure 9.4 montre les pixels détectés comme aberrants avec notre méthode qui corresponds au mouvement de la personne entre deux images.

Nous avons réalisé une deuxième expérience qui consiste dans un premier temps à faire bouger la Kinect dans une scène statique et de construire une carte 3D de l'environnement avec notre méthode. Dans un deuxième temps, nous avons refait la même séquence mais cette fois-ci avec une personne se déplaçant devant la caméra. Nous avons comparé visuellement la qualité de la carte 3D de l'environnement obtenue par les deux méthodes. Sur la figure 9.5 (page 155) est affichée la carte 3D obtenue dans une scène statique avec notre méthode, et sur la figure 9.6 (page 155) est affichée la carte 3D obtenue pour la scène dynamique. La comparaison visuelle de ces deux cartes montre que notre approche réussit à estimer correctement le mouvement de la

Kinect. Ce résultat est déduit de la qualité de reconstruction obtenue dans les deux cas.

9.2 Évaluation avec les données du Benchmark

Nous avons utilisé le *benchmark* de TUM RGB-D disponible en ligne développé par Sturm *et al.* [Sturm *et al.*, 2012a] pour comparer les performances de notre méthode avec une autre approche de la littérature. Ce *Benchmark* est composé d'un ensemble de séquences vidéo obtenues à partir d'une caméra mobile du type RGB-D dans différents types de scènes. Un système de capture de mouvement externe est utilisé pour obtenir la position précise de la caméra à chaque instant dans un référentiel commun. La performance d'une méthode d'odométrie visuelle est mesurée en calculant la dérive de la trajectoire estimée par rapport à la trajectoire de référence. Nous utilisons le critère *rpe* (*Relative Pose Error* [Sturm *et al.*, 2012a]) souvent utilisé pour comparer les méthodes d'odométrie visuelles entre elles. Ce critère mesure la précision d'une méthode d'odométrie visuelle sur un intervalle de temps δ fixe (en général $\delta = 1$ seconde). En effet, soit $P_1 \cdots P_n$ une séquence de n poses (une pose est une matrice rotation et un vecteur de translation) obtenues par une méthode d'odométrie visuelle et $Q_1 \cdots Q_n$ la séquence de n poses obtenues de la vérité terrain (Les deux séquences sont supposées synchronisées et associées). A chaque pas de temps i, l'erreur de pose relative est calculée de la façon suivante :

$$E_i = (Q_i^{-1} \cdot Q_{i+\delta})^{-1} \cdot (P_i^{-1} \cdot P_{i+\delta}).$$

Pour une séquence de n poses, nous obtenons ainsi $m = n - \delta$ valeurs de E_i sur toute la séquence. Pour évaluer les performances de la méthode sur toute la séquence le critère *RMSE* (en anglais *Relative Root Mean Square Error*) est calculée de la façon suivante :

$$RMSE(E_{i:n}, \delta) = \frac{1}{m} \cdot \sum_{i=1}^{m} ||trans(E_i)||,$$

avec $trans(E_i)$ est la partie translationnelle de E_i.

Les méthodes comparées sont les suivantes :

1. RANSAC : Notre méthode qui utilise RANSAC pour l'élimination des pixels aberrants mais sans utilisation de l'étape de nettoyage de la jacobienne (voir section 8.3.3).
2. RANSAC+SEL : Une deuxième variante de notre algorithme qui utilise RANSAC avec l'étape de nettoyage de la jacobienne.
3. Classic (LS) : La méthode originale qui utilise une méthode de moindres carrés classique.
4. WLS(Huber) : Une méthode robuste de la littérature développée par Audras *et al.* [Audras *et al.*, 2011] qui utilise un processus de minimisation par moindres carrés pondérés. Cette méthode calcule un poids représentant la confiance associée à chaque pixel en utilisant la fonction de Huber ([Huber, 1981]).

Dans la suite, nous comparons ces différentes méthodes sur des séquences vidéo du *benchmark* prises à partir d'une caméra mobile dans des scènes dynamiques contenant deux personnes se déplaçant devant la caméra.

La table 9.1 montre l'erreur moyenne de la dérive (*RMSE*) obtenue pour chaque méthode sur des séquences du *benchmark*. Les séquences de la table 9.1 peuvent être séparées en deux catégories :

— La première correspond à des mouvement très lents de deux personnes toujours assises sur des chaises et bougeant les mains et les bras générant ainsi très peu de pixels aberrants (les 3 premières séquences de la table. 9.1).

Dataset	Classic(LS)	WLS(Huber)	RANSAC	RANSAC+SEL
fr3/sitting static	0.016766m	0.015468m	0.015955m	**0.015354m**
fr3/sitting rpy	**0.046596m**	0.046707m	0.053552m	0.050798m
fr3/sitting xyz	0.131580m	**0.130301m**	0.132564m	0.131361m
fr3/walking static	0.188958m	0.175445m	0.170895m	**0.168123m**
fr3/walking rpy	0.321251m	0.324633m	0.317083m	**0.300369m**

TABLE 9.1 – Erreur moyenne de la dérive en mètre par seconde (RMSE) pour les différentes méthodes dans différentes scènes dynamiques du *benchmark*.

— La deuxième catégorie (les deux dernières séquences de la table 9.1) correspond à des mouvements rapides de deux personnes se déplaçant très prés de la caméra masquant parfois le champ de la caméra. Cette catégorie est classée difficile d'après les auteurs. Une raison supplémentaire pour laquelle ces séquences sont difficiles est liée à la nature de la scène. En effet, ces séquences ont été réalisées dans un hall ouvert comme le montre la figure 9.7(a) (page 156) où la majorité des pixels de l'image sont invalides ou se situent à plus que 4 mètres de la portée de la caméra. D'après l'étude de Khoshelham et Elberink [Khoshelham and Elberink, 2012], la précision de la Kinect se dégrade avec la distance, et les auteurs recommandent que la portée soit réduite à $3 \sim 4$ mètres pour les applications de localisation et cartographie. Ainsi, dans toutes les expériences cette portée a été réduite à 4 mètres ce qui réduit le pourcentage des pixels exploitables à la moitié (51.77%) sur ces séquences (voir figure 9.7(b) et 9.7(c)) et c'est principalement à cause du faible pourcentage de pixels exploitables que ces séquences sont très difficiles.

Les résultats de la table 9.1 amène à plusieurs conclusions. En effet, pour la première catégorie de séquences (mouvements très lents des deux personnes), toutes les méthodes possèdent des erreurs proches, avec une légère supériorité de notre méthode (RANSAC + SEL) et celle de WLS sur LS pour les séquences *sitting static* et *sitting xyz*. Pour la séquence *sitting rpy*, c'est la méthode classique qui possède l'erreur la plus faible mais qui reste proche de l'erreur des autres méthodes. Sur cette catégorie de séquences, le fait que les personnes effectuent très peu de mouvement, ce qui se traduit par un nombre de pixels aberrants très faible, explique pourquoi la méthode classique possède des résultats comparables aux autres méthodes.

Pour la deuxième catégorie de séquences difficiles (mouvements rapides de deux personnes se déplaçant très prés de la caméra), les erreurs de toutes les méthodes sont très élevées. Par contre, ce qui est intéressant à remarquer est que notre méthode (avec ses deux variantes) possède l'erreur la plus faible par rapport aux autres méthodes. Plus précisément, pour la séquence *walking static*, notre méthode (RANSAC+SEL) possède une erreur égale à 16.8 cm qui est inférieure à celle de WLS qui vaut 17.5 cm et de LS qui est égale à 18.8 cm. Pour la séquence *walking rpy*, l'erreur de RANSAC+SEL est égale à 30.0 cm qui est en dessous de l'erreur obtenue avec WLS qui vaut 32.4 cm et celle de LS qui est égale à 32.1 cm. Ces résultats montrent que notre méthode améliore les performances de la méthode classique et aussi donne des meilleurs résultats que la méthode WLS sur des séquences difficiles.

Une dernière conclusion tirée est que la variante de notre méthode RANSAC+SEL possède toujours une erreur légèrement inférieure à la version RANSAC de notre algorithme. Ce qui signifie que l'étape de nettoyage de la jacobienne améliore les performances de la méthode RANSAC.

9.3 Influence des paramètres sur les performances

Dans cette partie nous étudions l'influence de la variation des paramètres sur les performances de la méthode.

9.3.1 Seuil pour le nettoyage de la jacobienne

Dans le chapitre 8, nous avons décrit l'étape de nettoyage de la matrice jacobienne qui consiste à retirer les pixels qui ne rapportent pas d'information sur la position de la caméra afin de ne garder que ceux qui possèdent une bonne observabilité de chaque degré de liberté. Les résultats de la table 9.1 montrent que la version qui utilise cette étape de nettoyage (RANSAC+SEL) améliore légèrement les performances de la méthode qui n'utilise pas cette étape (RANSAC). Cette amélioration est importante sur certaines séquences que sur d'autres et dépend de la nature de la scène et surtout si elle est bien texturé ou non. Le coût de cette étape en temps de calcul est négligeable puisque le nettoyage est fait une seule fois au moment de la construction de la jacobienne (voir section 8.3.3).

Pour le choix de la valeur de s, nous avons réalisé une expérience consistant à faire varier s et observer l'image résultante après élimination des pixels inutiles. La figure 9.8 (page 157) montre l'image résultante après nettoyage de la jacobienne avec différentes valeurs de s. Nous remarquons que les pixels inutiles (peu texturés ou loin) sont filtrés d'une façon efficace lorsque le seuil est élevé. Sur toutes les séquences du *benchmark*, le meilleur filtrage est obtenu pour des valeurs de s proche de 50.

9.3.2 Choix du seuil

Un paramètre important qui a une influence directe sur les performances de la méthode est le seuil t que nous avons introduit dans la section 8.3.2 qui permet de considérer si un pixel a été aligné correctement ou non par une hypothèse. En d'autres termes, ce seuil est utilisé pour comparer la valeur absolue de chaque pixel r_i de l'image résidu R_i (résultante de la soustraction des deux images alignés par une hypothèse donnée) si elle est inférieure à t ou non, si c'est ce le cas, le pixel est rajouté à l'ensemble S_j de l'hypothèse. Théoriquement, ce seuil devrait être égal à zéro, puisque quand les deux images sont parfaitement alignées par une hypothèse, r_i vaut zéro pour tous les pixels de R_i. Mais en réalité, ce seuil n'est jamais nul. Étudions ce qui se passe lorsque nous varions t. Nous savons que l'intervalle de variation de t est entre $[0, 255]$. Prenons le cas extrême lorsque t est égale 255, dans ce cas là, pour n'importe quelle hypothèse générée par RANSAC, tous les pixels de l'image vont être considérés comme alignés par cette hypothèse. l'application de RANSAC n'a aucune importance dans ce cas là.

Sur la figure 9.9 (page 158) est affichée l'erreur moyenne (en bleu) sur la position de la caméra obtenue par notre méthode pour différentes valeurs de t sur une séquence prise avec une caméra fixe filmant une personne se déplaçant dans la scène. Nous remarquons que la meilleure précision est obtenue pour t entre 5 et 10. En terme de vitesse d'exécution (en rose sur la figure 9.9), lorsque t devient de plus en plus grande, la vitesse d'exécution de l'algorithme augmente. Cette augmentation de vitesse est expliquée par le fait que lorsque la valeur de t est grande, RANSAC n'a pas besoin d'exécuter toutes les K itérations puisqu'il va trouver à une certaine itération une hypothèse dont le rapport des pixels alignés par rapport au nombre total des pixels est supérieur à $1 - w$ (avec $w = 0.3$) et il va s'arrêter. Par contre cette hypothèse trouvée ne sera pas la meilleure estimation ce qui explique que la précision de l'algorithme est moins bonne pour les grandes valeurs de t.

9.3.3 Nombre de pixels

Le nombre de pixels utilisés pour la génération des hypothèses à une influence directe sur le nombre des itérations K. En effet, nous avons vu dans la section 8.3.4, que K croit exponentiellement avec ce nombre. Pour que l'algorithme reste applicable, le choix de n doit être limité dans un intervalle raisonnable. La courbe en rouge de la figure 9.10 (page 158) montre la variation de la vitesse d'exécution moyenne (en nombre d'images par seconde) pour les différentes valeurs de n entre 2 et 10, et la courbe en vert correspond au nombre des itérations pour chaque valeurs de n. Nous remarquons que les valeurs de n supérieures à 6, la vitesse d'exécution chute en dessous de 1 images par secondes. En terme de précision, la figure 9.11 montre l'erreur moyenne obtenue pour les différentes valeurs de n sur une séquence prise à partir d'une caméra fixe filmant une personne se déplaçant dans une scène. La valeur $n = 5$ permet d'avoir le meilleur rapport entre vitesse d'exécution et précision. Rappelons que pour ces 5 pixels, leurs voisins dans une fenêtre de 3×3 sont pris ce qui fait au total un nombre de 45 pixels (voir section 8.3.4).

9.4 Vitesse d'exécution

Bien que notre méthode améliore la précision de la méthode d'odométrie visuelle originale, elle rajoute une couche de complexité qui consiste à faire des itérations supplémentaires dans le but de pouvoir identifier et éliminer les pixels aberrants. Cela se traduit par un ralentissement de la vitesse d'exécution. En effet, notre implémentation actuelle (pour $n = 5$) tourne à une vitesse moyenne égale à 2 images par seconde tandis que la méthode originale tourne à 25 images par secondes. Les deux implémentations sont faites sur un processeur de dernière génération de type Intel i7.

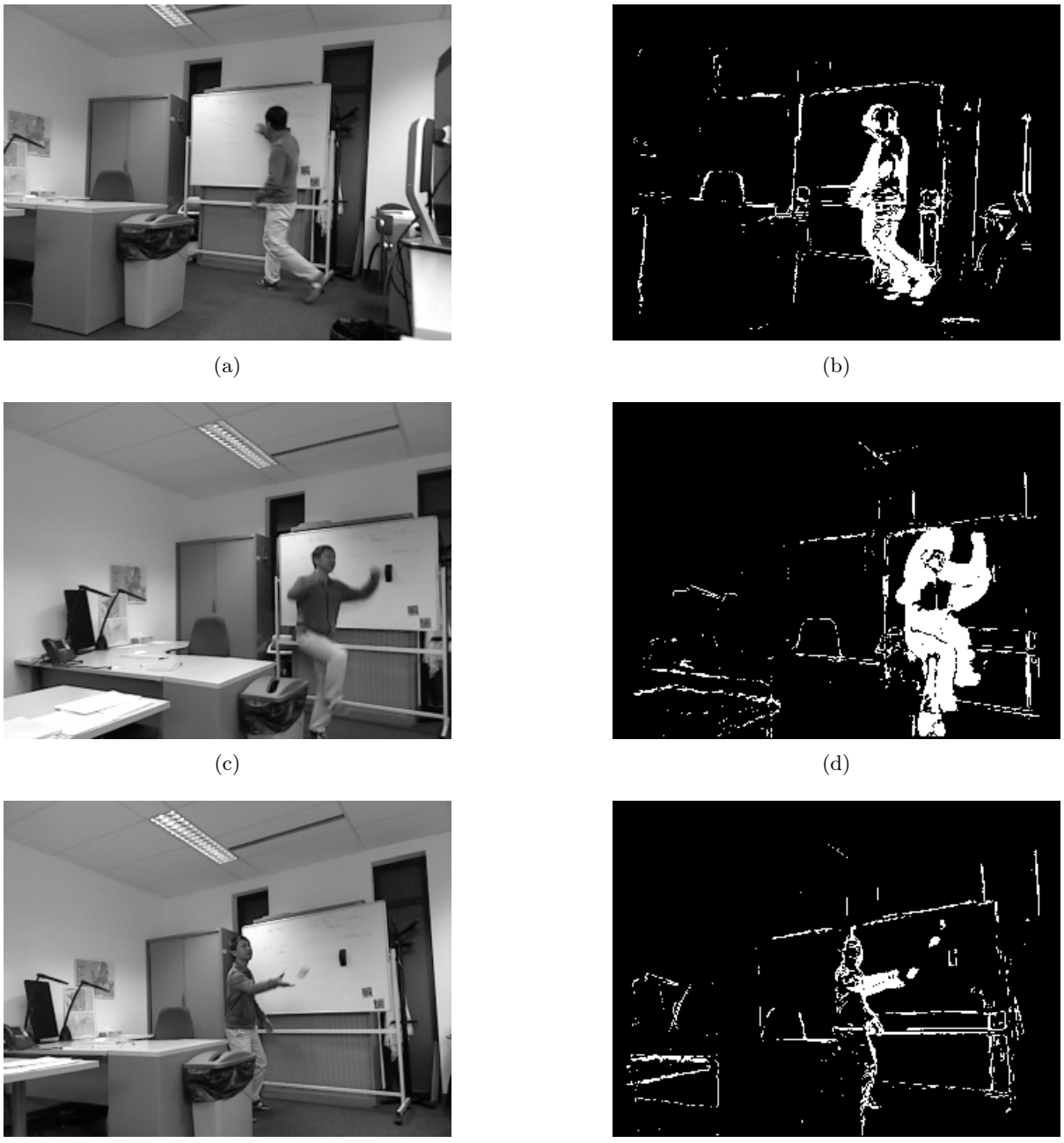
Pour une application temps-réels, il est important d'optimiser la version actuelle de notre algorithme. Une solution potentielle intéressante serait de paralléliser la génération et l'évaluation des différentes hypothèses produite par RANSAC. Surtout que ces hypothèses sont indépendantes les unes des autres. Ainsi, il est possible d'utiliser les différents cœurs présents dans un processeur moderne pour paralléliser l'algorithme.

9.5 Conclusion

Dans ce chapitre, nous avons évalué, dans un premier temps, notre méthode sur un ensemble de séquences que nous avons réalisées. Les résultats obtenus montrent la capacité de notre méthode à détecter et éliminer les pixels aberrants présents dans l'image. Ces résultats montrent aussi que notre méthode améliore considérablement la qualité de l'estimation du mouvement de la caméra par rapport à la méthode classique. Dans un deuxième temps, nous avons évalué notre méthode sur des séquences prises d'un *benchmark* en ligne, utilisé souvent pour comparer les méthodes d'odométrie visuelle entre elles. Les résultats obtenus montrent que notre méthode améliore la précision de la méthode d'odométrie visuelle classique dans des scènes dynamiques. Ces résultats montrent aussi que notre méthode possède des meilleures performances qu'une méthode existante (Huber) de la littérature dans les situations difficiles (deux personnes se déplaçant très prés de la caméra). Finalement, nous avons étudié l'influence de changement des paramètres de l'algorithme sur ses performances.

L'inconvénient majeure de notre méthode est son temps d'exécution puisque l'application de RANSAC rajoute de la complexité à la méthode d'odométrie visuelle classique. Une piste intéressante à explorer pour accélérer la version actuelle, serait de paralléliser la génération et

l'évaluation des différentes hypothèses produites par l'algorithme sur les différents cœurs présents dans un processeur moderne.



(a) (b)

(c) (d)

(e) (f)

FIGURE 9.4 – Premier colonne : Des images prises d'une séquence qui correspond à une personne effectuant plusieurs types de mouvement rapides devant une caméra mobile. Deuxième colonne : les pixels détectés comme aberrants par notre méthode sont affichés en blanc. Ces pixels correspondent au mouvement de la personne entre deux images consécutives.

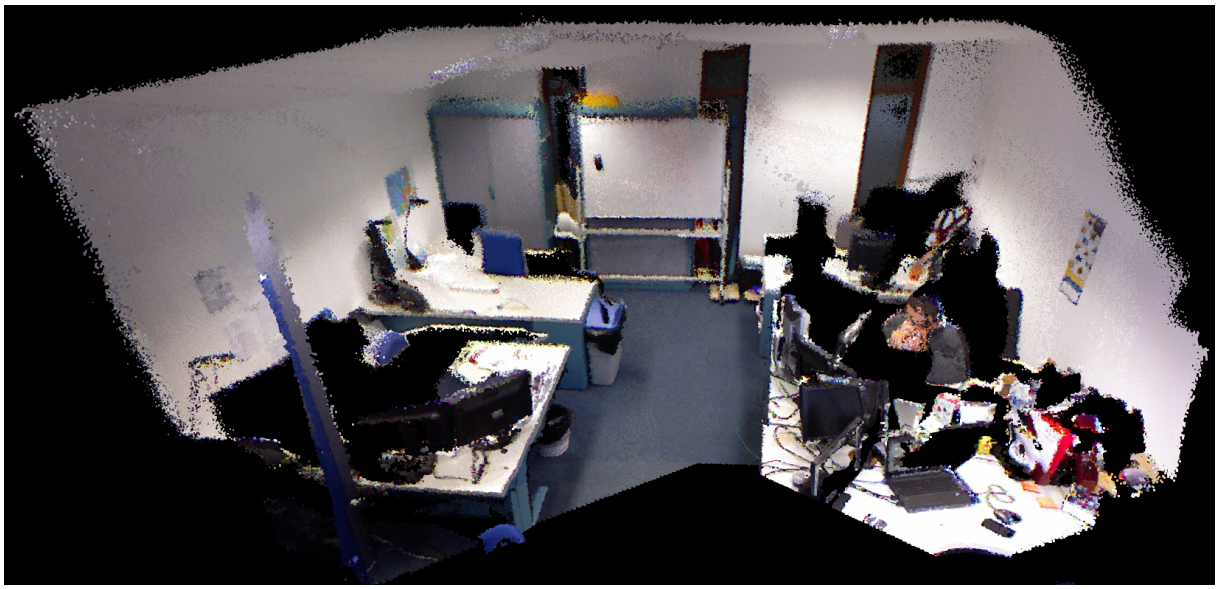


FIGURE 9.5 – Carte 3D obtenue obtenue par notre méthode dans une scène statique.

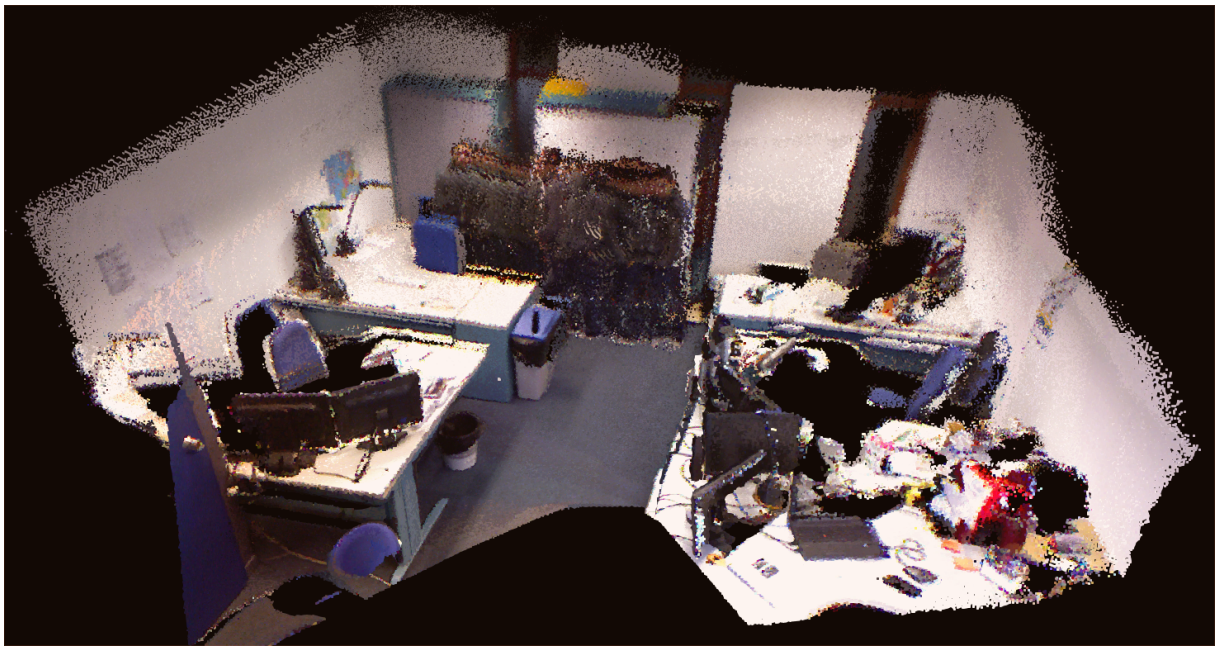
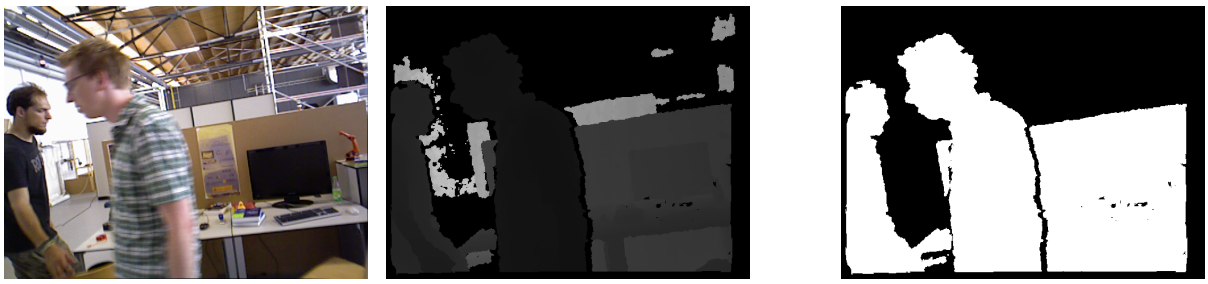


FIGURE 9.6 – Carte 3D obtenue par notre méthode dans une scène contenant une personne qui se déplace devant la caméra. Par comparaison visuelle avec le carte 3D faite lorsque la scène est statique (figure 9.5), nous remarquons que dans les deux cas la qualité de reconstruction est bonne, ce qui montre la capacité de notre méthode à estimer correctement le mouvement de la caméra dans une scène dynamique.



(a) (b) (c)

FIGURE 9.7 – a) Image de couleur de la séquence *freiburg3 walking xyz* du *benchmark* b) Image de profondeur correspondante. c)Représentation binaire de l'image avec les pixels invalides et non exploitables en noir. Le pourcentage des ces pixels représente 49.23% de l'image.

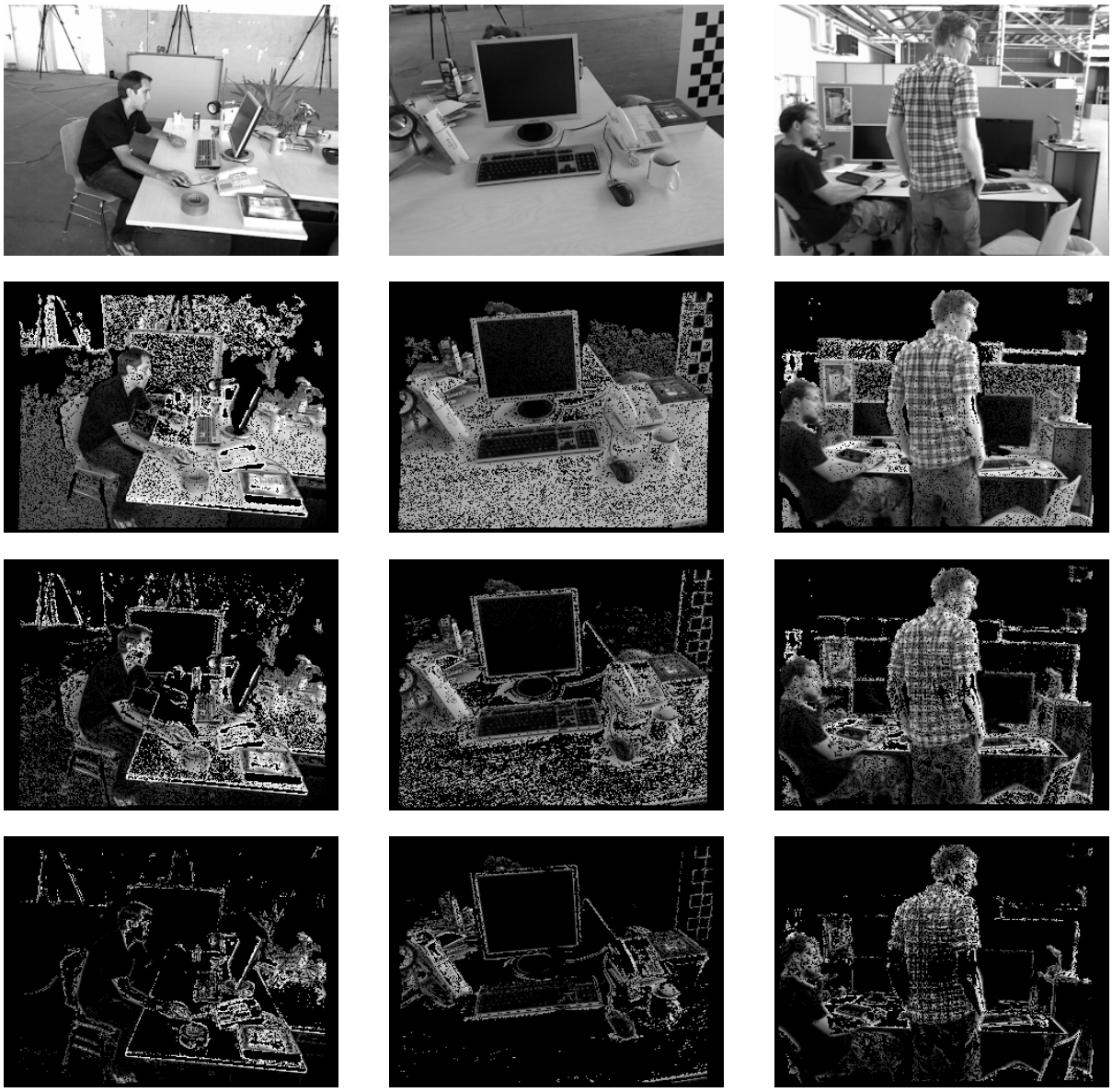


FIGURE 9.8 – Image résultante pour différente valeur du seuil s. Première ligne : image d'entrée pour différente types de scène du *benchmark*. Deuxième ligne : Image résultante après nettoyage de la jacobienne avec $s = 10$. Troisième ligne : Image résultante après nettoyage de la jacobienne avec $s = 50$. Quatrième ligne : Image résultante après nettoyage de la jacobienne avec $s = 200$.

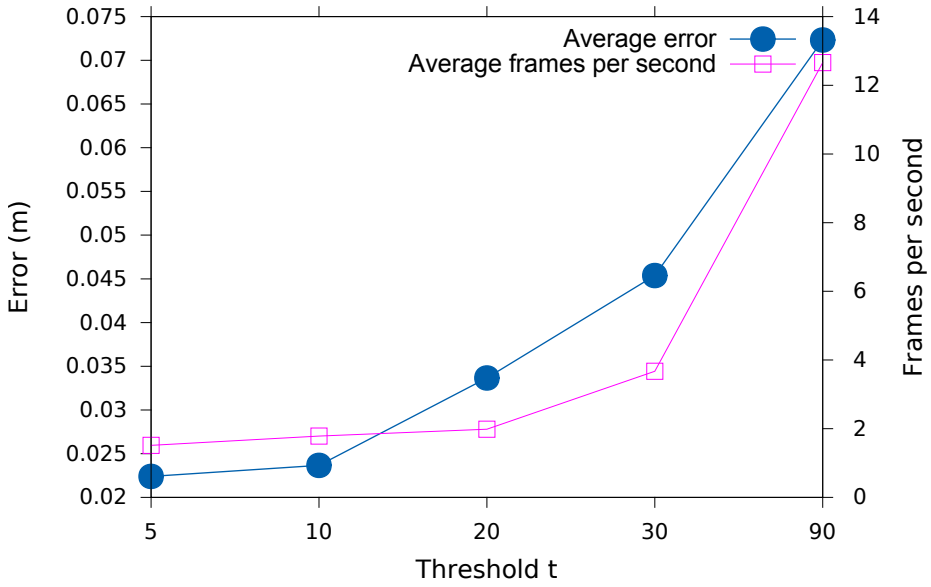


FIGURE 9.9 – En bleu : l'erreur moyenne sur la position de la caméra obtenue par notre méthode pour différentes valeurs de t sur une séquence prise avec une caméra fixe filmant une personne se déplaçant dans la scène. En rose : variation de la vitesse d'exécution en fonction de t.

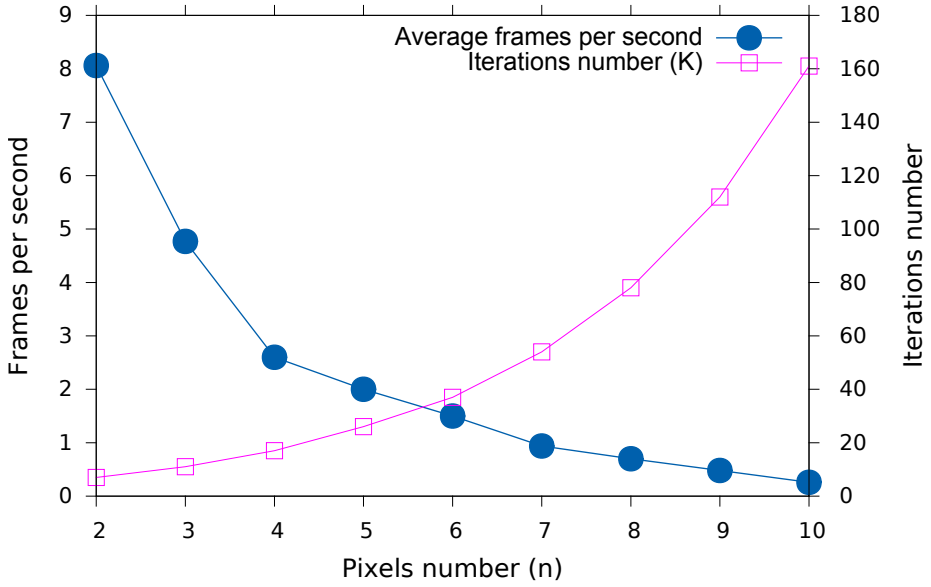


FIGURE 9.10 – Variation de la vitesse d'exécution et de nombre des itérations en fonction de nombre des pixels choisis pour la génération des hypothèses.

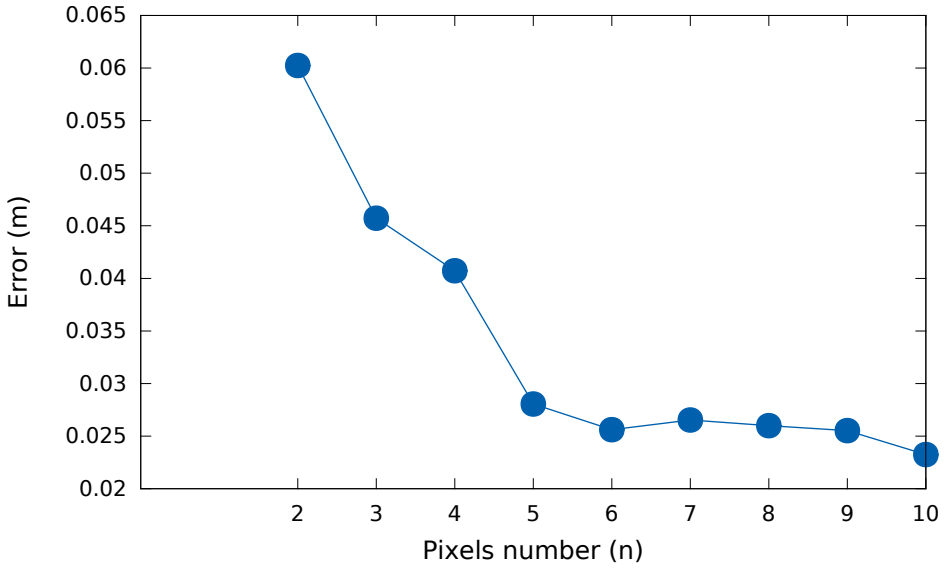


FIGURE 9.11 – Erreur moyenne sur la position de la caméra pour différentes valeurs de n sur une séquence prise avec une caméra fixe filmant une personne se déplaçant dans la scène.

Conclusion générale

Dans cette partie, nous avons traité le problème de la localisation d'une caméra mobile dans un environnement dynamique. Pour cela, nous avons proposé une nouvelle méthode de localisation visuelle, qui étend l'approche originale basée sur le calcul de flux optique, pour la rendre robuste dans un environnement dynamique. La première contribution de cette partie est l'utilisation de l'algorithme RANSAC pour détecter et éliminer les pixels aberrants du processus d'estimation du mouvement de la caméra. La seconde contribution est le développement d'une technique permettant d'améliorer la qualité des hypothèses générées par RANSAC. Cette technique consiste à nettoyer la matrice Jacobienne en retirant les pixels qui ne rapportent pas d'information sur le mouvement de la caméra. Nous avons montré avec les multiples expériences réalisées, que notre méthode améliore la qualité de l'estimation du mouvement de la caméra par rapport à la méthode originale dans des scènes dynamiques. L'inconvénient majeur de notre méthode est son temps d'exécution. En effet, la version actuelle tourne à 2 images par seconde. Pour une application temps-réel, il serait important d'optimiser cette version.

Notre objectif principal dans cette thèse est de doter un robot mobile d'un système de perception de la posture humaine à bas coût. Le défi dans cette thèse était d'utiliser uniquement une caméra du type RGB-D *low cost* pour le développement des différents algorithmes. Cette caméra représente une alternative à d'autres types de capteurs comme le télémètre laser. Ce dernier capteur fournit une estimation précise et robuste même dans un environnement dynamique. Par contre, un inconvénient de ce capteur est son prix coûteux qui vaut entre 1000 et 5000 euros pour un produit de bonne qualité tel que le Hokuyo.

Pour la mise en place du système de perception visuelle de la posture humaine sur un robot mobile, rappelons que la méthode de capture de mouvement, développée dans la partie précédente, utilise l'image de profondeur, reçue de la caméra, pour mettre à jour l'état de chaque cellule de la grille d'occupation. Ce qui permet ensuite d'apprendre et d'adapter en continu le fond de la scène, et d'extraire la silhouette de la personne. Pour avoir une mise à jour correcte de la grille d'occupation lorsque la caméra est mobile, nous allons utiliser la transformation (rotation+translation) renvoyée par la méthode d'odométrie visuelle, que nous avons développée dans cette partie. Ainsi, dans la partie suivante, nous décrivons, dans un premier temps, la mise en place du système complet de perception visuelle de la posture humaine. Ce système intègre les différentes composantes développées dans cette thèse, à savoir, l'odométrie visuelle, l'extraction de la silhouette et le suivi du mouvement 3D de la personne. Ensuite, nous présentons les premiers résultats que nous avons obtenus, les difficultés et problèmes rencontrés.

Quatrième partie

Vers une plate-forme robotique d'analyse de mouvement de la personne dans un environnement intérieur et encombré

Chapitre 10

Système de perception visuelle de la posture humaine

Ce chapitre décrit l'intégration des travaux réalisés tout au long de cette thèse pour développer une application de suivi du mouvement humain à partir d'une caméra montée sur un robot d'assistance mobile. Cette application est un système logiciel qui intègre les différentes méthodes développées :

- l'odométrie visuelle, qui a pour rôle d'estimer le mouvement de la caméra mobile.
- l'extraction de la silhouette, qui utilise une grille d'occupation 3D pour modéliser l'environnement dynamique, identifier les parties mobiles dans la scène et extraire la silhouette de la personne.
- le suivi du mouvement, qui utilise un algorithme de filtrage particulaire pour suivre, dans l'espace et le temps, la posture de la personne.

Dans un premier temps, nous décrivons l'architecture de ce système qui connecte les différentes méthodes citées ci-dessus. Dans un deuxième temps, nous discutons les premiers résultats que nous avons obtenus avec les expériences que nous avons réalisées dans l'appartement expérimental du LORIA. Ces expériences nous ont permis d'identifier les limites du système. Finalement, nous présentons les pistes possibles pour améliorer les performances du système.

10.1 Architecture

Sur la figure 10.1 est affichée l'architecture du système de perception visuelle de la posture humaine qui intègre les différents résultats que nous avons obtenus tout au long de cette thèse. L'odométrie visuelle reçoit une image de profondeur et de couleur à l'instant t, estime le déplacement de la caméra mobile entre l'instant $t-1$ et t. Ce déplacement représenté par la matrice de transformation homogène $\hat{\xi}_t$ de taille 4×4, composée d'un vecteur de translation et d'une matrice de rotation, est cumulée pour construire la transformation globale ξ_t au cours du temps :

$$\xi_t = \xi_{t-1} \cdot \hat{\xi}_t, \tag{10.1}$$

ξ_t représente la transformation entre la position à l'instant t de la caméra et le repère global G attaché à la grille. ξ_t est ensuite utilisée pour mettre à jour la grille d'occupation avec le nuage de points reçu à l'instant t. Ainsi, l'algorithme 5 pour la mise à jour de la grille d'occupation est remplacé par le nouvel algorithme 8. Les modifications par rapport à l'algorithme précédent sont montrées en gras. Ce nouvel algorithme parcourt toutes les cellules c_i de la grille, projette les coordonnées (X_i, Y_i, Z_i) de chaque cellule dans le repère caméra par l'inverse de ξ_t. Le nouveau

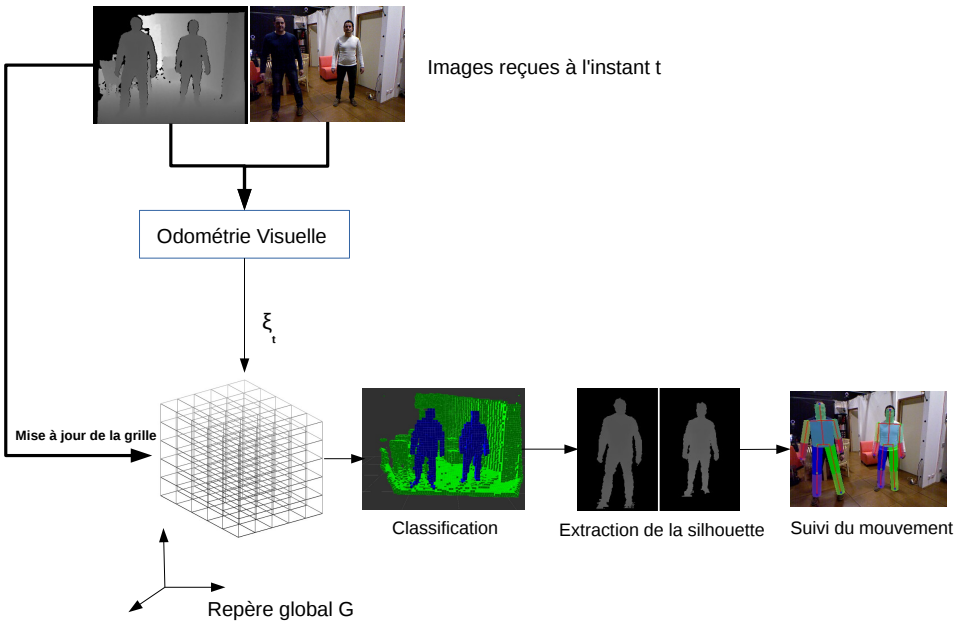


FIGURE 10.1 – L'Architecture du système de perception visuelle de la posture humaine composé de trois parties : l'odométrie visuelle, l'extraction de la silhouette (grille d'occupation + HMM), et le suivi du mouvement de la personne par filtrage particulaire.

point résultant $(X\prime_i, Y\prime_i, Z\prime_i)$ est ensuite projeté dans le plan de la caméra et le HMM est appliqué pour identifier si la cellule est occupée par un objet fixe ou mobile de la scène. Après la mise à jour de la grille d'occupation, l'algorithme d'étiquetage est appliqué pour regrouper les cellules mobiles en étiquettes.

Pour extraire la silhouette d'une étiquette, les points M_{min} et M_{max} de sa boite englobante sont d'abord projetés dans le repère caméra en utilisant la transformation ξ_t. Ensuite ils sont projetés dans le plan image de la caméra en utilisant les équations de projection perspective. Chaque pixel contenu dans la zone 2D, qui correspond à la projection de M_{min} et M_{max} dans le plan image, est projeté d'abord en 3D et ensuite dans la grille en utilisant la transformation inverse de ξ_t pour tester s'il appartient à la fois à la boite englobante de l'étiquette et à une cellule mobile. Si c'est le cas, le pixel appartient à la silhouette (la section 3.4.2 décrit le fonctionnement de la méthode d'extraction de la silhouette dans le cas où la caméra est fixe). La silhouette obtenue est utilisée par l'algorithme de filtrage particulaire pour extraire la posture.

Pour la méthode de gestion des occlusions. Les sphères constituant chaque cylindre sont projetées dans le repère de la grille en utilisant l'inverse de ξ_t pour tester si chaque sphère appartient à une cellule visible ou non (la section 4.5.3 décrit le fonctionnement la méthode de gestion des occlusions).

10.2 Résultats préliminaires

Nous avons réalisé quelques séquences de test vidéos à partir d'une caméra montée sur un robot mobile filmant une personne se déplaçant dans l'appartement expérimental du LORIA (voir

Algorithm 8: Algorithme de mise à jour de la grille d'occupation par un HMM pour une caméra mobile.

entrée : grille d'occupation, image de profondeur I_t, $\boldsymbol{\xi_t}$

pour *chaque cellule* c_i *dans la grille de coordonnées* (X_i, Y_i, Z_i) **faire**

- /* Projection dans le repère caméra */
- $\boldsymbol{(X_i', Y_i', Z_i') = \xi_t^{-1} \cdot (X_i, Y_i, Z_i)}$
- /* projection perspective*/
- $\boldsymbol{u_i = \frac{X_i' \times f_x}{Z_i'} + c_x}$
- $\boldsymbol{v_i = \frac{Y_i' \times f_y}{Z_i'} + c_y}$
- **si** (u_i, v_i)*dans le plan de la caméra* **alors**
  - $d_i = I_t(u_i, v_i)$
  - $\varepsilon_i = d_i - Z\prime_i$
  - $obs = f(\varepsilon_i)$
  - /* Mise à jour de l'état de chaque cellule par le HMM*/
  - $P(Q_t^i = L \mid e_{Y_{1:t}^i}) = (1 - obs) \cdot [P(Q_{t-1}^i = F) \cdot \alpha + P(Q_{t-1}^i = M) \cdot \gamma + P(Q_{t-1}^i = L) \cdot (1 - \beta)]$
  - $P(Q_t^i = M \mid e_{Y_{1:t}^i}) = obs \cdot [P(Q_{t-1}^i = L) \cdot \beta + P(Q_{t-1}^i = M) \cdot (1 - \gamma - \psi)]$
  - $P(Q_t^i = F \mid e_{Y_{1:t}^i}) = obs \cdot [P(Q_{t-1}^i = F) \cdot (1 - \alpha) + P(Q_{t-1}^i = M) \cdot \psi]$
  - /* Normalisation*/
  - $somme = P(Q_t^i = L \mid e_{Y_{1:t}^i}) + P(Q_t^i = M \mid e_{Y_{1:t}^i}) + P(Q_t^i = F \mid e_{Y_{1:t}^i})$
  - $P(Q_t^i = L \mid e_{Y_{1:t}^i}) = \frac{P(Q_t^i = L \mid e_{Y_{1:t}^i})}{somme}$
  - $P(Q_t^i = M \mid e_{Y_{1:t}^i}) = \frac{P(Q_t^i = M \mid e_{Y_{1:t}^i})}{somme}$
  - $P(Q_t^i = F \mid e_{Y_{1:t}^i}) = \frac{P(Q_t^i = F \mid e_{Y_{1:t}^i})}{somme}$

FIGURE 10.2 – La personne filmée par une caméra montée sur un robot mobile.

figure 10.2). La trajectoire du robot a été programmée manuellement [8]. Deux types de trajectoires ont été effectuées par le robot. La première trajectoire programmée consiste effectuer une rotation sur place en observant toujours la scène. La seconde consiste à effectuer des aller retour tout en observant la scène.

Les résultats qualitatifs de suivi pris à différents instants sont illustrés sur la figure 10.3. Ces résultats montrent que le système réussit à suivre le mouvement de la personne à partir d'une caméra mobile et dans un environnement encombré. Ces résultats montrent que notre méthode de gestion d'occlusions réussit à identifier correctement les parties du corps invisibles, et à les retirer du processus de l'estimation du mouvement de la personne. Ce qui produit au final un suivi précis et robuste.

Les résultats obtenus sont intéressants puisqu'ils montrent la faisabilité d'un tel système malgré les difficultés que nous avons rencontrées durant ces expériences et que nous présentons dans la suite.

10.2.1 Limites constatées

En effet, sur toutes les séquences que nous avons réalisées (rotation sur place et translation), le système n'était pas capable d'analyser chaque séquence en entier à cause des dérives de la méthode d'odométrie visuelle amenant à une fausse mise à jour de la grille d'occupation et par la suite une fausse extraction de la silhouette ce qui traduit au final par une perte de suivi complète. En effet, le bon fonctionnement du système est conditionné par une estimation correcte du mouvement de la caméra par l'odométrie visuelle. Toutes les méthodes d'odométrie visuelle soufrent d'un problème de dérive dans le temps à cause de l'accumulation des petites erreurs dues à l'estimation du mouvement entre deux images consécutives (voir équation 10.1). Ces erreurs sont d'autant plus importantes dans une scène dynamique. La dérive de l'odométrie visuelle se traduit par une mauvaise transformation ξ_t ne décrivant pas exactement le mouvement effectué par la caméra. Nous avons vu que ξ_t est utilisé pour mettre à jour l'état de chaque cellule de la grille d'occupation avec le HMM. Si cependant, ξ_t est fausse, la classification par le HMM des cellules n'est pas correcte. Ce qui se traduit, dans certains cas, par une partie du décor qui est identifiée comme mobile par exemple. Cette mauvaise classification amène forcément à une mauvaise extraction de la silhouette de profondeur. La méthode de suivi de mouvement dans ce cas là échoue puisque la silhouette utilisée pour la construction de la fonction de vraisemblance contient une partie du décors. Un exemple montrant l'impact de la dérive de l'odométrie visuelle

8. Dans cette thèse, nous ne traitons pas le problème de la navigation du robot

sur le fonctionnement du système est affichée sur la figure 10.4. Sur cette figure, Nous remarquons qu'une mauvaise estimation du mouvement de la caméra amène à une mise à jour incorrecte de la grille d'occupation se traduisant par une partie du mur identifiée comme mobile (figure 10.4(a)). Par la suite, l'application de l'algorithme d'étiquetage génère une silhouette contenant une partie du mur (figure 10.4(b)). Ce qui amène le suivi à l'échec (jambe droite attachée au mur sur la figure 10.4(c)).

Un autre problème, relevé pendant cette expérience, est lié au fait que la trajectoire du robot est programmé manuellement, et au champ de vision réduit de la Kinect 1 (57 degrés vertical et 43 degrés horizontal). Ce qui fait que le mouvement du robot n'était pas synchronisé avec celle de la personne. Ce qui amène, dans certains cas, à la personne qui sort complètement du champ de vision de la caméra. En effet, ce problème relève une question très importante qui n'a pas été traitée dans cette thèse et il serait intéressant de la traiter dans les travaux futurs. La question est la suivante : comment positionner le robot pour mieux observer la personne et pour qu'elle soit toujours dans le champs de la caméra.

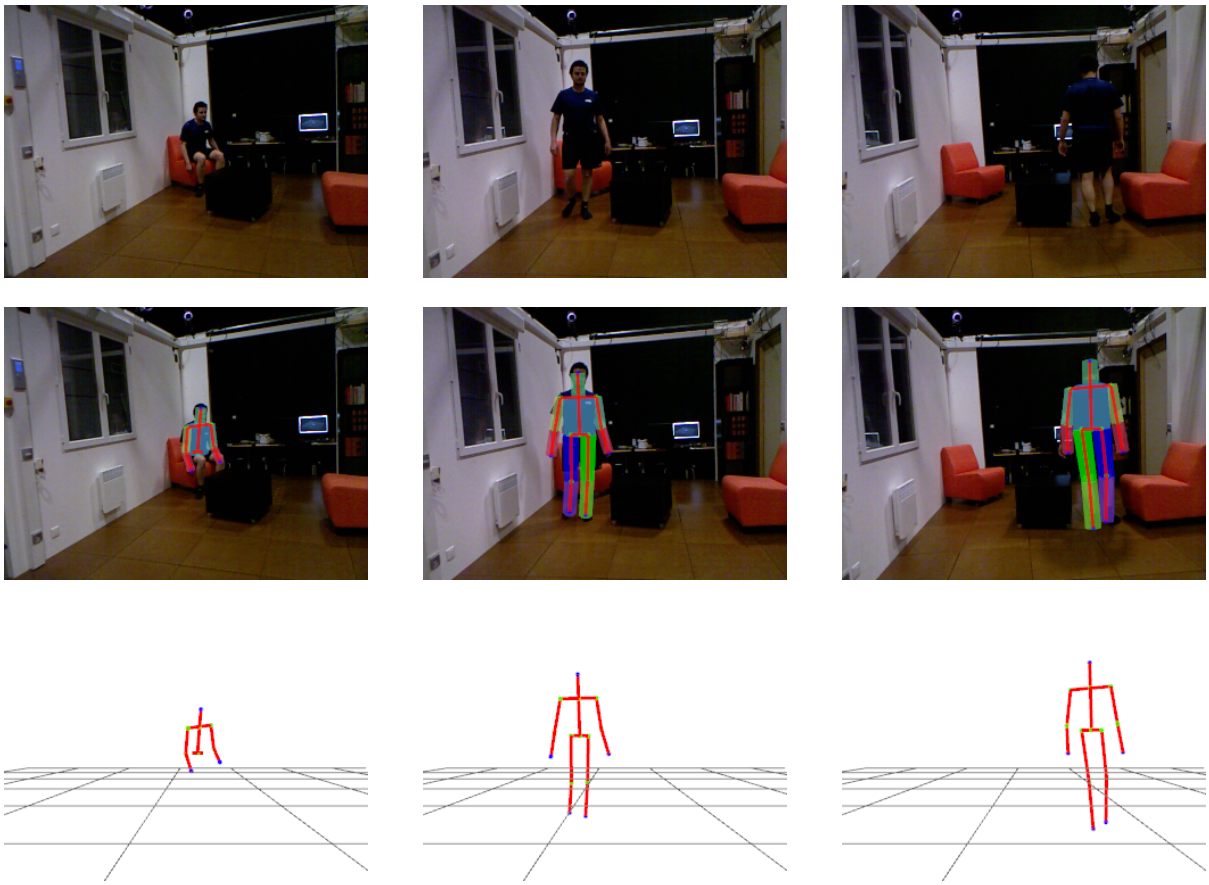


FIGURE 10.3 – Résultats obtenus par notre système montrant sa capacité à suivre correctement le mouvement de la personne à partir d'une caméra mobile dans un environnement en présence d'occlusions (première colonne : t, deuxième colonne : $t + 4$ sec, troisième colonne : $t + 7$ sec).

(a) (b) (c)

FIGURE 10.4 – Impact de la dérive de l'odométrie sur le fonctionnement du système : Une mauvaise estimation du mouvement de la caméra amène à une mise à jour incorrecte de la grille d'occupation se traduisant dans cet exemple par une partie du mur identifiée comme mobile (figure a). Par la suite, l'application de l'algorithme d'étiquetage génère une silhouette contenant une partie du mur (figure b) ce qui amène le suivi à l'échec (jambe droite attachée au mur sur la figure c).

10.2.2 Pistes de travail

Nous avons vu qu'une fausse estimation du mouvement de la caméra a un impact direct sur la classification des cellules de la grille d'occupation. Ce qui se traduit par des objets du décor qui sont identifiés comme mobiles comme le montre la figure 10.4(c). Une première solution pour pallier ce problème serait d'adapter les probabilités de transitions entre les différents états du HMM pour réduire l'impact des dérives de l'odométrie visuelle. Par exemple, augmenter les probabilités de transition de l'état mobile à l'état fixe et libre (γ et ψ de la figure 3.12). De cette manière, les objets du décor identifiés comme mobiles vont passer à nouveau à l'état fixe après un temps relativement court qui dépend des valeurs de γ et ψ. En revanche, l'augmentation des probabilités de transitions (γ et ψ) peut poser, dans certaines situations, un problème pour l'extraction de la silhouette de la personne. En effet, pour les grandes valeurs de γ et ψ, si la personne effectue des mouvements lents, les cellules occupées par la personne, qui sont identifiées comme mobiles, peuvent devenir fixes et par conséquent la silhouette de la personne sera perdu. Une deuxième solution pour pallier ce problème serait d'utiliser une grille d'occupation avec une résolution très basse, dans le but de réduire l'impact des dérives de l'odométrie visuelle. En effet, à titre d'exemple, une grille d'occupation de résolution 10 cm est moins sensible aux dérives qu'une grille de 5 cm de résolution.

Une autre solution serait d'intégrer d'autres types de capteurs embarqués sur le robot [9]. En effet, notre objectif de départ était développer un système de suivi de mouvement de la personne et de localisation du robot à partir d'un capteur bas coût de type Kinect. Si nous relâchons cette contrainte en intégrant les informations provenant des capteurs souvent embarqués sur un robot, comme un accéléromètre et/ou un gyroscope, nous pouvons alors fusionner les informations provenant d'autres types de capteurs avec l'odométrie visuelle. Ce qui pourrait réduire considérablement les dérives dans le temps.

Une solution plus intéressante pour améliorer la méthode d'odométrie visuelle, serait de remplacer l'alignement des images consécutives par un alignement par rapport à une image clef (en anglais *keyframe*) suffisamment proche de l'image courante. En effet, l'utilisation du *keyframe* permet de réduire considérablement la dérive de l'odométrie visuelle surtout quand la caméra

9. Ou des capteurs présents dans l'environnement.

effectue des mouvements lents ([Huang *et al.*, 2011a]). A partir de là, il serait important d'étendre la méthode d'odométrie visuelle actuelle vers une méthode de SLAM (*Simultaneous Localization And Mapping* [Leonard and Durrant-Whyte, 1991]).

10.3 Discussion

Le suivi du mouvement d'une personne à partir d'une caméra mobile dans un environnement dynamique et encombré est un sujet de recherche difficile et très peu traité dans la littérature. La difficulté est liée à plusieurs facteurs, principalement, la dynamique de la scène qui se traduit par un environnement qui évolue en continu, les occlusions générées par les objets présents dans la scène masquant certaines parties du corps et le mouvement de la caméra [10]...

Dans cette thèse, nous avons identifié et traité certains problèmes dans le but de faire avancer ce sujet de recherche. Les résultats obtenus montrent que la réalisation d'un système de perception de la posture humaine à partir d'une caméra mobile est possible. A partir des résultats obtenus dans cette thèse, il serait intéressant dans un premier temps d'exploiter les pistes d'améliorations que nous avons citées dans la section précédente pour améliorer le système actuel. Puis dans un deuxième temps de traiter les autres difficultés comme la navigation et le positionnement du robot.

La question de positionnement du robot de façon à mieux observer la personne est particulièrement intéressante. Ce sujet devra nécessairement être traité pour la réalisation d'un robot compagnon d'assistance, capable de suivre efficacement les mouvements d'une personne. La stratégie de navigation du robot pourrait être construite de manière à optimiser le champ de vision de la caméra mais aussi en essayant de réduire les occlusions.

Vers un système bas coût, de perception de la posture, localisation, cartographie et navigation

Le système que nous avons conçu dans cette thèse constitue un point de départ pour le développement d'un système plus complexe, capable à partir d'une seule caméra, d'estimer la posture de la personne, localiser le robot et aussi assurer la navigation et le positionnement du robot. En effet, un point fort de notre système actuel est qu'il repose entièrement sur une grille d'occupation pour modéliser l'environnement, extraire la silhouette et gérer les occlusions. Un avantage de l'utilisation de la grille d'occupation est qu'elle permet d'avoir une représentation unique de l'espace. Et par conséquent, elle est capable d'intégrer naturellement les informations provenant d'autres types de capteurs, pouvant être présents dans l'environnement. Comme exemple de ces capteurs, nous pouvons citer, des caméras fixes, ou un système des dalles intelligentes composé d'un réseau de capteurs de pression au sol (nous disposons d'un prototype dans l'appartement expérimental du LORIA ([Pepin *et al.*, 2009])). Dans la littérature, les grilles d'occupation sont aussi utilisées pour traiter le problème du SLAM et la navigation du robot. L'avantage que notre grille a par rapport à ces grilles est qu'elle permet de modéliser un environnement dynamique et non pas juste statique. Il est alors possible, dans des travaux futurs, d'étendre notre système pour traiter le problème SLAM et la navigation du robot pour l'évitement des obstacles et aussi pour mieux positionner le robot dans le but d'avoir une meilleure observation sur l'état de la personne. Un avantage d'un tel système, serait sa capacité à traiter, à partir d'un seul capteur bas coût,

10. Il existe d'autres difficultés qui ne sont pas traitées dans cette thèse comme la navigation du robot, l'aspect multi-personnes, interaction de la personne avec l'environnement, positionnement du robot (pour mieux observer la personne) etc...

le problème de suivi du mouvement de la personne, et à assurer la localisation, navigation et le positionnement du robot dans un environnement réel (dynamique et encombré).

En conclusion, une problématique importante à étudier dans la suite de cette thèse serait de pouvoir traiter simultanément la question de localisation/navigation et positionnement du robot, cartographie d'un environnement dynamique et le suivi du mouvement de la personne.

Conclusion et perspective

Le vieillissement de la population est un enjeu majeur auquel les sociétés modernes vont devoir faire face dans les années à venir. D'ici 2050, la proportion de personnes âgées de plus de 60 ans va atteindre les 30% dans les pays développés et notamment la France. L'un des enjeux, par sa dimension sociale et économique porte sur le maintien à domicile. Pour les personnes ayant gardé leur autonomie, la problématique est de conserver cet état le plus longtemps possible, tout en limitant les risques, notamment les accidents domestiques ou les malaises. Motivé par cet enjeu, l'objectif du projet L.A.R. (Living Assistant Robot) financé dans le cadre des investissements d'avenir, au sein duquel cette thèse s'est déroulée, est de développer une plate-forme robotique d'assistance et un environnement domotique dans le but de surveiller la personne chez elle, détecter des situations à risques et aider la personne à réaliser certaines tâches du quotidien.

L'objectif défini pour cette thèse était le développement d'un système, pouvant être embarqué sur un robot mobile, capable de localiser et suivre la personne en mouvement et d'en estimer la posture y compris lorsque la personne suivie est partiellement occultée par des objets. L'extraction de la posture est un élément important pour les applications de surveillance, d'assistance et d'interaction. Elle permet de mieux maîtriser certaines situations qui nécessitent de comprendre l'intention de la personne avec laquelle le robot interagit, ou encore de détecter des situations à risque, comme les chutes ou encore d'analyser les capacités motrices de la personne.

Pour répondre à l'objectif défini pour cette thèse dans le cadre du projet L.A.R, nous avons proposé un système de perception visuelle, à partir d'une caméra RGB-D bas coût, de la posture humaine dans un environnement dynamique et encombré pouvant être embarqué sur un robot mobile.

Au moment de la conception du système, nous avons identifié et adressé les problèmes suivants :

— Extraction de la silhouette dans un environnement dynamique.
— Suivi de la posture humaine dans un environnement encombré.
— Localisation du robot dans un environnement dynamique.

Pour le premier problème, du fait que l'arrière plan peut évoluer à cause de la dynamique de la scène et du mouvement de la caméra, les méthodes traditionnelles d'extraction de la silhouette par soustraction de l'arrière plan ne sont pas applicables dans notre cas. Pour cela, nous avons proposé une nouvelle méthode d'extraction de la silhouette, capable en même temps d'apprendre en ligne le fond de la scène. La méthode développée utilise une grille d'occupation coupler à un HMM pour identifier si chaque cellule de la grille est occupée par un objet fixe ou mobile de la scène. La méthode proposée possède plusieurs avantages :

— Elle tourne en temps réel.
— Elle extrait la silhouette de la personne à la résolution native du capteur, tout en ayant une grille d'occupation de faible résolution.
— La méthode est capable d'extraire la silhouette de plusieurs personnes présentes dans la scène même si l'aspect multi-personnes n'a pas été traité dans cette thèse.

— La méthode ne nécessite aucune connaissance sur la nature et le type de scène.
— La grille d'occupation permet d'avoir une représentation unique de l'espace. Elle permet aussi d'intégrer naturellement plusieurs capteurs fixes ou mobiles. Ce qui peut être intéressant pour l'intégration des informations provenant d'autres types de capteurs présents dans l'environnement.

Pour le second problème qui est le suivi du mouvement humain dans un environnement en présence d'occlusions. Nous avons vu que ce sujet est peu traité dans la littérature et reste un défi scientifique ouvert. Nous avons aussi vu que la majorité des méthodes existantes sont testées dans des environnements contrôlés et supposent que la personne soit toujours visible en entier. Pour cela, nous avons proposé une nouvelle méthode de suivi du mouvement humain capable, et avec une seule caméra, de gérer correctement les situations d'occlusions et de fournir un suivi robuste dans ces conditions. La méthode proposée modifie une méthode existante de la littérature basée sur le filtrage particulaire pour la rendre robuste aux occlusions. Cette modification consiste principalement à interroger la grille d'occupation pour identifier quelle partie du corps n'est pas visible pour la retirer du processus d'estimation du mouvement. Nous avons vu dans la partie expérimentale, que le fait de retirer ces parties du corps rend le suivi robuste et améliore considérablement la qualité de suivi. Un des avantages de la méthode de gestion d'occlusions que nous avons développée est qu'elle est simple à mettre en œuvre et elle ne nécessite aucune connaissance sur le type de la scène et l'emplacement des obstacles.

Cependant la méthode proposée possède certaines limitations qui ont été soulignées dans la conclusion du chapitre 5. Nous citons principalement, le temps de calcul qui un inconvénient majeur de la méthode actuelle. Un travail d'optimisation, en utilisant au maximum la puissance du GPU, est en cours de réalisation pour accélérer la version actuelle de l'algorithme et les premiers résultats obtenus sont très concluants. Une autre limitation de la méthode proposée est qu'elle utilise un ensemble de cylindre pour le modèle 3D qui reste une approximation et ne décrit pas parfaitement la forme des parties du corps de la personne. Additionnellement, ce modèle est générique et n'est pas adapté à la forme de chaque personne. L'utilisation d'un modèle 3D plus fin, qui caractérise plus précisément la posture de la personne, peut améliorer considérablement la qualité du squelette extrait. Par l'adaptation automatique du modèle 3D aux différents types de posture des personnes, une solution potentielle serait de paramétrer le vecteur d'état avec les dimensions des segments, et ensuite d'apprendre ces paramètres. De cette façon les paramètres intrinsèques de chaque partie du corps seraient appris pendant le suivi.

Par ailleurs, nous avons construit un *benchmark* composé d'un ensemble de séquences vidéo enregistrées avec une caméra RGB-D filmant une personne se déplaçant dans différents types de scènes contenant des obstacles masquant certaines parties du corps. La vérité terrain est obtenue par un système externe de capture de mouvement utilisant des marqueurs réfléchissants fixés sur la personne. Ce *benchmark* est disponible en ligne sur el lien suivant [motion capture dataset, 2015], et il est mis à disposition de la communauté.

Pour le dernier problème, le fait que l'environnement soit dynamique rend la tâche de localisation difficile. En effet, le déplacement des objets présents dans la scène induit une difficulté supplémentaire pouvant nuire à l'estimation du mouvement de la caméra. Pour cela, nous avons proposé une amélioration de la méthode d'odométrie visuelle dense en utilisant l'algorithme RANSAC pour identifier et éliminer les pixels aberrants de l'image. Dans les expériences que nous avons réalisées, nous avons montré la capacité de notre méthode à éliminer les pixels aberrants présents dans la scène. Nous avons aussi montré que la nouvelle méthode améliore les performances de la méthode d'odométrie visuelle classique. Cependant, l'inconvénient majeur de notre méthode est son temps d'exécution puisque l'application de RANSAC rajoute de la complexité à la méthode d'odométrie visuelle classique. Un travail d'optimisation serait important

pour une application temps-réel. Un autre inconvénient est lié à la nature de la méthode d'odométrie visuelle qui génère des dérives dans le temps à cause de l'accumulation des petites erreurs dues à l'estimation du mouvement entre deux images consécutives. Une solution intéressante pour réduire ces dérives serait de remplacer la procédure d'alignement des images consécutives par un alignement par rapport à une image clef (en anglais *keyframe*) suffisamment proche de l'image courante. Cela permettrait de réduire considérablement les dérives surtout quand la caméra effectue des mouvements lents. Il serait aussi important d'étendre la méthode d'odométrie visuelle actuelle vers une méthode de SLAM complète.

Dans la dernière partie de cette thèse, nous avons décrit l'intégration du système complet de la perception de la posture humaine. Ce système regroupe les trois méthodes développées dans cette thèse, à savoir, l'extraction de la silhouette, le suivi du mouvement de la personne et l'odométrie visuelle. La connexion entre les différentes parties du système a été décrite et consiste principalement à utiliser la transformation décrivant le mouvement de la caméra entre deux instants d'acquisition, pour mettre à jour correctement la grille d'occupation avec le nuage de points. Les premiers résultats obtenus montrent que la réalisation d'un tel système est possible malgré les limites du système actuel. Ces limites sont principalement dues aux dérives dans le temps de la méthode d'odométrie visuelle. En effet, le bon fonctionnement du système est conditionné par une bonne estimation du mouvement de la caméra. Les dérives de la méthode de localisation amènent à une fausse mise à jour de la grille d'occupation et par la suite une mauvaise extraction de la silhouette de la personne. Une solution intéressante à ce problème serait d'étendre la méthode d'odométrie visuelle vers une méthode de SLAM.

En conclusion, le problème du suivi de mouvement d'une personne à partir d'une caméra mobile dans un environnement dynamique et encombré est peu traité dans la littérature et reste un sujet de recherche difficile. Dans cette thèse, nous avons traité certains problèmes techniques dans le but de faire avancer ce sujet. Ces problèmes sont liés principalement à la dynamique de la scène, les occlusions inévitables dans un environnement encombré et la localisation de la caméra. Nous avons montré qu'un tel système est faisable. Il existe d'autres problèmes qui n'ont pas été abordés dans cette thèse. Nous citons principalement, la navigation autonome du robot dans l'environnement, l'interaction de la personne avec des objets de la scène et finalement, le positionnement du robot pour mieux observer la personne. Un des avantages du système proposé est qu'il repose sur la grille d'occupation pour modéliser un environnement dynamique, extraire la silhouette de la personne et gérer les occlusions. Dans la littérature, les grilles d'occupation sont aussi utilisées pour traiter le problème du SLAM et la navigation du robot. Il est alors possible, dans des travaux futurs, d'étendre notre système pour traiter le problème SLAM et la navigation du robot pour l'évitement des obstacles et aussi pour mieux positionner le robot dans le but d'avoir une meilleure observation sur l'état de la personne. La question de positionnement du robot de façon à mieux observer la personne est particulièrement intéressante. Ce sujet devra nécessairement être traité dans la suite de cette thèse pour la réalisation d'un robot compagnon d'assistance, capable de suivre efficacement les mouvements d'une personne.

Annexes

Annexe A

Ré-échantillonnage multinomial

Soit $S_t = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}^{i:1\cdots N}$ l'ensemble de N particules obtenues après l'étape de pondération. Soit $U(0,1)$ une distribution uniforme sur l'intervalle $[0,1)$. L'échantillonnage multinomial consiste à tirer d'abord N valeurs ordonnées selon la loi uniforme U notées :

$$k_1, k_2, \cdots k_i, \cdots k_N.$$

Un nouveau système de particule S' avec des particules équiprobables est généré égal à :

$$S'_t = \{\mathbf{x}_t^{(i),D(k_i)}, \frac{1}{N}\},$$

où $D(k_i)$ est l'entier unique q qui indique le nombre de fois la particule i sera dupliquée tel que :

$$\sum_{n=1}^{q-1} w_t^{(n)} < k_i \leq \sum_{n=1}^{q} w_t^{(n)},$$

où $\sum_{n=1}^{q} w_t^{(n)}$ représente la fonction de répartition de la densité *a posteriori*. De cette façon, l'ensemble des particules pondérées est transformé en un nouvel ensemble de particules non pondérées mais, cette fois-ci, les particules sont dupliquées proportionnellement à leur poids.

Annexe B

Calcul des composants de la jacobienne

En utilisant le théorème de décomposition, nous pouvons décomposé J_w de la façon suivante :

$$J_w = J_P \cdot J_g \cdot J_\xi$$

1. J_P est la matrice jacobienne 2×3 de la dérivée de la transformation P de l'équation 7.5 par rapport aux coordonnées (X', Y', Z') de M' :

$$J_P = \begin{bmatrix} \frac{\partial P_u}{\partial X'} & \frac{\partial P_u}{\partial Y'} & \frac{\partial P_u}{\partial Z'} \\ \frac{\partial P_v}{\partial X'} & \frac{\partial P_v}{\partial Y'} & \frac{\partial P_v}{\partial Z'} \end{bmatrix} = \begin{bmatrix} f_x \cdot \frac{1}{Z'} & 0 & -f_x \cdot \frac{X'}{Z'^2} \\ 0 & f_y \cdot \frac{1}{Z'} & -f_y \cdot \frac{Y'}{Z'^2} \end{bmatrix}. \tag{B.1}$$

2. J_g est la matrice jacobienne 3×12 de la dérivée de la transformation rigide g par rapport à ses composants.

$$J_g = \begin{bmatrix} \frac{\partial g_{X'}}{r_{11}} & \frac{\partial g_{X'}}{r_{21}} & \frac{\partial g_{X'}}{r_{31}} & \frac{\partial g_{X'}}{r_{12}} & \frac{\partial g_{X'}}{r_{22}} & \frac{\partial g_{X'}}{r_{32}} & \frac{\partial g_{X'}}{r_{13}} & \frac{\partial g_{X'}}{r_{23}} & \frac{\partial g_{X'}}{r_{33}} & \frac{\partial g_{X'}}{t_x} & \frac{\partial g_{X'}}{y_y} & \frac{\partial g_{X'}}{t_z} \\ \frac{\partial g_{Y'}}{r_{11}} & \frac{\partial g_{Y'}}{r_{21}} & \frac{\partial g_{Y'}}{r_{31}} & \frac{\partial g_{Y'}}{r_{12}} & \frac{\partial g_{Y'}}{r_{22}} & \frac{\partial g_{Y'}}{r_{32}} & \frac{\partial g_{Y'}}{r_{13}} & \frac{\partial g_{Y'}}{r_{23}} & \frac{\partial g_{Y'}}{r_{33}} & \frac{\partial g_{Y'}}{t_x} & \frac{\partial g_{Y'}}{y_y} & \frac{\partial g_{Y'}}{t_z} \\ \frac{\partial g_{Z'}}{r_{11}} & \frac{\partial g_{Z'}}{r_{21}} & \frac{\partial g_{Z'}}{r_{31}} & \frac{\partial g_{Z'}}{r_{12}} & \frac{\partial g_{Z'}}{r_{22}} & \frac{\partial g_{Z'}}{r_{32}} & \frac{\partial g_{Z'}}{r_{13}} & \frac{\partial g_{Z'}}{r_{23}} & \frac{\partial g_{Z'}}{r_{33}} & \frac{\partial g_{Z'}}{t_x} & \frac{\partial g_{Z'}}{y_y} & \frac{\partial g_{Z'}}{t_z} \end{bmatrix},$$

ce qui donne :

$$J_g = \begin{bmatrix} X & 0 & 0 & Y & 0 & 0 & Z & 0 & 0 & 1 & 0 & 0 \\ 0 & X & 0 & 0 & Y & 0 & 0 & Z & 0 & 0 & 1 & 0 \\ 0 & 0 & X & 0 & 0 & Y & 0 & 0 & Z & 0 & 0 & 1 \end{bmatrix}. \tag{B.2}$$

3. J_ξ est la matrice jacobienne 12×6 de l'application exponentielle par rapport à ξ. Pour la calculer, g peut s'écrire en fonction des matrices génératrices $(Gi)_{i=1\cdots6}$ qui sont les bases de l'algèbre de Lie. Ainsi nous avons :

$$g = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = e^{\hat{\xi}} = e^{G1 \cdot v_1 + G2 \cdot v_2 + G3 \cdot v_3 + G4 \cdot w_1 + G5 \cdot w_2 + G6 \cdot w_3}.$$

avec :

$$G1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$G4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Ainsi J_ξ s'écrit sous la forme suivante :

$$J_\xi = \begin{bmatrix} \frac{\partial g}{\partial v_1} & \frac{\partial g}{\partial v_2} & \frac{\partial g}{\partial v_3} & \frac{\partial g}{\partial w_1} & \frac{\partial g}{\partial w_2} & \frac{\partial g}{\partial w_3} \end{bmatrix} = \begin{bmatrix} G1 \cdot g & G2 \cdot g & G3 \cdot g & G4 \cdot g & G5 \cdot g & G6 \cdot g \end{bmatrix}.$$

En remplaçant g par :

$$g = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

et en ré-arrangeant Gi sous forme vectorielle, nous obtenons :

$$J_\xi = \begin{bmatrix} 0 & 0 & 0 & 0 & r_{31} & -r_{21} \\ 0 & 0 & 0 & -r_{31} & 0 & r_{11} \\ 0 & 0 & 0 & r_{21} & -r_{11} & 0 \\ 0 & 0 & 0 & 0 & r_{32} & -r_{22} \\ 0 & 0 & 0 & -r_{32} & 0 & r_{12} \\ 0 & 0 & 0 & r_{22} & -r_{12} & 0 \\ 0 & 0 & 0 & 0 & r_{33} & -r_{23} \\ 0 & 0 & 0 & -r_{33} & 0 & -r_{13} \\ 0 & 0 & 0 & r_{23} & -r_{13} & 0 \\ 1 & 0 & 0 & 0 & t_z & -t_y \\ 0 & 1 & 0 & -t_z & 0 & t_x \\ 0 & 0 & 1 & t_y & -t_x & 0 \end{bmatrix}. \tag{B.3}$$

Publications

[Dib and Charpillet, 2015a] A. Dib and F. Charpillet. Pose estimation for a partially observable human body from rgb-d cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4915–4922, Sept 2015.

[Dib and Charpillet, 2015b] A. Dib and F. Charpillet. Robust dense visual odometry for rgb-d cameras in a dynamic environment. In *IEEE International Conference on Advanced Robotics (ICAR)*, pages 1–7, July 2015.

[Dib *et al.*, 2010] A. Dib, C. Rose, and F. Charpillet. Bayesian 3d human motion capture using factored particle filtering. In *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 370–372, Oct 2010.

[Dib *et al.*, 2014] A. Dib, N. Beaufort, and F. Charpillet. A real time visual slam for rgb-d cameras based on chamfer distance and occupancy grid. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 652–657, July 2014.

[Dubois *et al.*, 2011] Amandine Dubois, Abdallah Dib, and Francois Charpillet. Using hmms for discriminating mobile from static objects in a 3d occupancy grid. In *Proceedings of the IEEE 23rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 170–176, Nov 2011.

Références

[Ananth, 2004] Ranganathan Ananth. The levenberg-marquardt algorithm, 2004.

[Arras *et al.*, 2007] Kai O. Arras, Óscar Martínez Mozos, and Wolfram Burgard. Using boosted features for detection of people in 2d range scans. In *IEEE International Conference on Robotics and Automation*, 2007.

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

[Asus Xtion,] Xtion by asus. `https://www.asus.com/3D-Sensor/Xtion_PRO/`.

[Audras *et al.*, 2011] C. Audras, A.I. Comport, M. Meilland, and P. Rives. Real-time dense RGB-D localisation and mapping. In *Australian Conference on Robotics and Automation*, Monash University, Australia, December 7-9 2011.

[Azad *et al.*, 2007] P. Azad, A. Ude, T. Asfour, and R. Dillmann. Stereo-based markerless human motion capture for humanoid robot systems. In *IEEE International Conference on Robotics and Automation*, pages 3951–3956, April 2007.

[Baker and Matthews, 2004] Simon Baker and Iain Matthews. Lucas-kanade 20 years on : A unifying framework. *International Journal of Computer Vision*, 56(3) :221 – 255, March 2004.

[Bandouch and Beetz, 2009] J. Bandouch and M. Beetz. Tracking humans interacting with the environment using efficient hierarchical sampling and layered observation models. In *IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2040–2047, Sept 2009.

[Bandouch *et al.*, 2008] Jan Bandouch, Florian Engstler, and Michael Beetz. Evaluation of Hierarchical Sampling Strategies in 3D Human Pose Estimation. In *Proceedings of the 19th British Machine Vision Conference (BMVC)*, 2008.

[Bay *et al.*, 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features SURF. *Comput. Vis. Image Underst.*, 110(3) :346–359, June 2008.

[Besl and McKay, 1992] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2) :239–256, February 1992.

[Biclops,] Biclops by traclabs. `http://traclabs.com/products/biclops-pt/`.

[Bjorck, 1996] Ake Bjorck. *Numerical Methods for Least Squares Problems*. Siam Philadelphia, 1996.

[Breiman, 2001] Leo Breiman. Random forests. *Mach. Learn.*, 45(1) :5–32, October 2001.

[Burt and Adelson, 1983] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2 :217–236, 1983.

[Cabaret and Lacassagne, 2014] L. Cabaret and L. Lacassagne. What is the world's fastest connected component labeling algorithm ? In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, Oct 2014.

[Comaniciu *et al.*, 2002] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift : A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 :603–619, 2002.

[Comport *et al.*, 2010] A.I. Comport, E. Malis, and P. Rives. Real-time quadrifocal visual odometry. *Int. J. Rob. Res.*, 29(2-3) :245–266, February 2010.

[Dalal and Triggs, 2005] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893 vol. 1, June 2005.

[Darrell *et al.*, 1998] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 601–608, Jun 1998.

[David, 1970] Herbert A. David. *Order statistics*. A Wiley series in probability and mathematical statistics. Applied probability and statistics. J. Wiley and sons, New York, 1970.

[Deutscher *et al.*, 1999] J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through singularities and discontinuities by random sampling. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1144–1149 vol.2, 1999.

[Deutscher *et al.*, 2000] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 126–133 vol.2, 2000.

[Dorai *et al.*, 1997] C. Dorai, J. Weng, and A.K. Jain. Optimal registration of object views using range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10) :1131–1138, Oct 1997.

[Douc and Cappe, 2005] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69, Sept 2005.

[Doucet and Johansen, 2011] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing : fifteen years later, 2011.

[Doucet *et al.*, 2000] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3) :197–208, July 2000.

[Efron and Tibshirani., 1993] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman Hall, 1993.

[Elfes, 1989a] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6) :46–57, June 1989.

[Faugeras, 1992] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In *Proceedings of the Second European Conference on Computer Vision*, ECCV, pages 563–578, London, UK, UK, 1992. Springer-Verlag.

[Felzenszwalb and Huttenlocher, 2005] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *Int. J. Comput. Vision*, 61(1) :55–79, January 2005.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6) :381–395, June 1981.

[Forster *et al.*, 2014] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO : Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[Freund and Schapire, 1997] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1) :119–139, August 1997.

[Friedman *et al.*, 1977] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3) :209–226, September 1977.

[Ganapathi *et al.*, 2012] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real-time human pose tracking from range data. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pages 738–751, Berlin, Heidelberg, 2012. Springer-Verlag.

[Gavrila and Davis, 1996] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action : a multi-view approach. In *CVPR*, pages 73–80, 1996.

[Ghiasi *et al.*, 2014] G. Ghiasi, Yi Yang, D. Ramanan, and C.C. Fowlkes. Parsing occluded people. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2401–2408, June 2014.

[Grest and Koch, 2004] D. Grest and R. Koch. Realtime multi-camera person tracking for immersive environments. In *IEEE 6th Workshop on Multimedia Signal Processing*, pages 387–390, Sept 2004.

[Harris and Stephens, 1988] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[Hartley and Zisserman, 2004] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN : 0521540518, second edition, 2004.

[Harville, 2002] Michael Harville. Stereo person tracking with adaptive plan-view statistical templates. *Image and Vision Computing*, 22 :127–142, 2002.

[Hasler *et al.*, 2009] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel. Markerless motion capture with unsynchronized moving cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 224–231, June 2009.

[Hayashi *et al.*, 2004] K. Hayashi, M. Hashimoto, K. Sumi, and K. Sasakawa. Multiple-person tracker with a fixed slanting stereo camera. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 681–686, May 2004.

[Hewson *et al.*, 2007] David J. Hewson, Jacques Duchêne, François Charpillet, Jamal Saboune, Valérie Michel-Pellegrino, Hassan Amoud, Michel Doussot, Jean Paysant, Anne Boyer, and Jean-Yves Hogrel. The parachute project : Remote monitoring of posture and gait for fall prevention. *EURASIP J. Appl. Signal Process.*, 2007(1) :109–109, January 2007.

[Ho, 1995] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Aug 1995.

[Hokuyo,] Scanning range finder by hokuyo. `http://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html`.

[Huang *et al.*, 2011a] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *In Proc. of the Intl. Sym. of Robot. Research*, 2011.

[Huber, 1981] P. Huber. *Robust Statistics*. 1981.

[Importance Sampling,] Importance sampling by standford university. `http://statweb.stanford.edu/~owen/mc/Ch-var-is.pdf`.

[Instanced Rendering,] Opengl instanced rendering. `https://www.opengl.org/sdk/docs/man3/xhtml/glDrawElementsInstanced.xml`.

[Intel Threading Building Blocks,] Intel threading building blocks tbb. `https://www.threadingbuildingblocks.org/`.

[Isard and Blake, 1998] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29 :5–28, 1998.

[Jafari *et al.*, 2014] O.H. Jafari, D. Mitzel, and B. Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5636–5643, May 2014.

[Jeanpierre, 2002] Laurent Jeanpierre. *Apprentissage et adaptation pour la modélisation stochastique de systèmes dynamiques réels*. PhD thesis, université de Lorraine, 2002.

[Jiu *et al.*, 2014] Mingyuan Jiu, Christian Wolf, Graham W. Taylor, and Atilla Baskurt. Human body part estimation from depth images via spatially-constrained deep learning . *Pattern Recognition Letters*, 50(1) :122–129, December 2014.

[Kalman, 1960] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

[Kerl *et al.*, 2013b] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2013.

[Khoshelham and Elberink, 2012] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2) :1437–1454, 2012.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

[Kitagawa, 1996] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1) :1–25, 1996.

[Konolige *et al.*, 2006] Kurt Konolige, Motilal Agrawal, Robert C. Bolles, Cregg Cowan, Martin Fischler, and Brian Gerkey. Outdoor mapping and navigation using stereo vision. In *Proceedings of the International Symposium on Experimental Robotics*, 2006.

[Lallemand *et al.*, 2013] J. Lallemand, O. Pauly, L. Schwarz, D. Tan, and S. Ilic. Multi-task forest for human pose estimation in depth images. In *International Conference on 3D Vision*, pages 271–278, June 2013.

[Lecun *et al.*, 1998] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, Nov 1998.

[Lee and Nevatia, 2009] Mun Wai Lee and R. Nevatia. Human pose tracking in monocular sequence using multilevel structured models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1) :27–38, Jan 2009.

[Lee *et al.*, 2002] Mun Wai Lee, Isaac Cohen, and Soon Ki Jung. Particle filter with analytical inference for human body tracking. In *Workshop on Motion and Video Computing*, pages –, 2002.

[Leonard and Durrant-Whyte, 1991] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS)*, pages 1442–1447 vol.3, Nov 1991.

[Liu and Chen, 1998] Jun S. Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93 :1032–1044, 1998.

[Liu *et al.*, 2013] Yebin Liu, J. Gall, C. Stoll, Qionghai Dai, Hans-Peter Seidel, and C. Theobalt. Markerless motion capture of multiple characters using multiview image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11) :2720–2735, 2013.

[Longuet-Higgins, 1987] H. C. Longuet-Higgins. Readings in computer vision : Issues, problems, principles, and paradigms. In Martin A. Fischler and Oscar Firschein, editors, *Nature*, chapter A Computer Algorithm for Reconstructing a Scene from Two Projections, pages 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[Lovegrove *et al.*, 2011] Steven Lovegrove, Andrew J. Davison, and Javier Ibanez Guzman. Accurate visual odometry from a rear parking camera. In *Intelligent Vehicles Symposium*, pages 788–793. IEEE, 2011.

[Lowe, 1999] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision -Volume 2-*, ICCV '99, Washington, DC, USA, 1999. IEEE Computer Society.

[Lucas and Kanade, 1981] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, April 1981.

[Ma *et al.*, 2003] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision : From Images to Geometric Models*. SpringerVerlag, 2003.

[MacCormick and Isard, 2000] John MacCormick and Michael Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, ECCV '00, pages 3–19, London, UK, UK, 2000. Springer-Verlag.

[Metamotion,] Mechanical motion capture system meta motion. `http://www.metamotion.com/`.

[Meyer-Delius *et al.*, 2012] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *AAAI*, 2012.

[Microsoft,] Microsoft kinect sdk. `https://dev.windows.com/en-us/kinect`.

[Microsoft Kinect,] Kinect by microsoft. `https://en.wikipedia.org/wiki/Kinect`.

[Mip-mapping,] Opengl 4.0 mip-mapping. `https://www.opengl.org/sdk/docs/man/html/glGenerateMipmap.xhtml`.

[Modern OpenGL tutorial,] Modern opengl tutorial. `http://www.labri.fr/perso/nrougier/teaching/opengl/`.

[Moeslund *et al.*, 2006] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2) :90–126, November 2006.

[Motion Analysis,] Motion capture system motion analysis. `http://www.motionanalysis.com/`.

[motion capture dataset, 2015] Synchronized rgbd and motion capture dataset. `https://team.inria.fr/larsen/software/datasets`, February 2015.

[Murphy, 2002] Kevin Patrick Murphy. Dynamic bayesian networks : Representation, inference and learning, 2002.

[Newcombe *et al.*, 2011a] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion : Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[Newcombe *et al.*, 2011b] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam : Dense tracking and mapping in real-time. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2320–2327, Washington, DC, USA, 2011. IEEE Computer Society.

[Nguyen, 2013] Xuan Son Nguyen. *Exploitation of particle filter and dynamics bayesians networks for articulated object tracking.* PhD thesis, Sorbonne universités, 2013.

[Nister *et al.*, 2006] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23 :2006, 2006.

[Nuchter *et al.*, 2007] Andreas Nuchter, Kai Lingemann, and Joachim Hertzberg. Cached k-d tree search for ICP algorithms. *International Conference on 3D Digital Imaging and Modeling*, 0 :419–426, 2007.

[OpenGL Shading Language,] Opengl shading language glsl. `https://www.opengl.org/documentation/glsl/`.

[Optitrack,] Motion capture system optitrack. `http://www.optitrack.com/`.

[Organic Motion,] Markeless motion capture openstage. `http://www.organicmotion.com/`.

[Parzen, 1962] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3) :pp. 1065–1076, 1962.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[Pepin *et al.*, 2009] Nicolas Pepin, Olivier Simonin, and François Charpillet. Intelligent Tiles : Putting Situated Multi-Agents Models in Real World. In ACM AAAI, editor, *International Conference on Agents and Artificial Intelligence - ICAART*, Porto, Portugal, January 2009.

[Peursum *et al.*, 2007] P. Peursum, S. Venkatesh, and G. West. Tracking-as-recognition for articulated full-body human motion analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[Phasespace,] Motion capture system phasespace. `http://www.phasespace.com/`.

[Phoenix,] Motion capture system phoenix. `http://www.ptiphoenix.com/`.

[Pomerleau *et al.*, 2009] François Pomerleau, Francis Colas, François Ferland, and François Michaud. Relative motion threshold for rejection in ICP registration. In *Field and Service Robotics*, pages 229–238. Springer Berlin/Heidelberg, 2009.

[Pomerleau *et al.*, 2011] François Pomerleau, Stéphane Magnenat, Francis Colas, Ming Liu, and Roland Siegwart. Tracking a depth camera : parameter exploration for fast ICP. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3824–3829. IEEE, 2011.

[Pomerleau *et al.*, 2015] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1) :1–104, 2015.

[Poppe, 2007] Ronald Poppe. Vision-based human motion analysis : An overview. *Comput. Vis. Image Underst.*, 108(1-2) :4–18, October 2007.

[Qualisys,] Motion capture system qualisys. `http://www.qualisys.com/`.

[Rabiner, 1989] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[Rafi *et al.*, 2015] Umer Rafi, Juergen Gall, and Bastian Leibe. A semantic occlusion model for human pose estimation from a single depth image. In *In CVPR Workshops*, June 2015.

[Rose *et al.*, 2008] Cédric Rose, Jamal Saboune, and François Charpillet. Reducing particle filtering complexity for 3d motion capture using dynamic bayesian networks. In *Proceedings of the 23rd national conference on Artificial intelligence*, pages 1396–1401. AAAI Press, 2008.

[Rose, 2011] Cedric Rose. *Modélisation stochastique pour le raisonnement médical et ses applications à la télémédecine*. PhD thesis, université de Lorraine, 2011.

[Rosten and Drummond, 2006] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006.

[Rosten *et al.*, 2010] Edward Rosten, Gerhard Reitmayr, and Tom Drummond. Improved RANSAC performance using simple, iterative minimal-set solvers. *CoRR*, abs/1007.1432, 2010.

[Rublee *et al.*, 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb : An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[Rusinkiewicz and Levoy, 2001] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[Saboune and Charpillet, 2005a] Jamal Saboune and François Charpillet. Markerless human motion capture for Gait analyis. In *3rd European Medical and Biological Engineering Conference - EMBEC'05*, Prague, République Tchèque, November 2005.

[Saboune and Charpillet, 2005b] Jamal Saboune and François Charpillet. Using Interval Particle Filtering for Marker less 3D Human Motion Capture. In *17th IEEE International Conference on Tools with Artificial Intelligence - ICTAI'05*, pages 621–627, Hong Kong, China, November 2005. IEEE.

[Saboune, 2008] Jamal Saboune. *Développement d'un système passif de suivi 3D du mouvement humain par filtrage particulaire*. PhD thesis, université de Lorraine, 2008.

[Scaramuzza and Fraundorfer, 2011] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robot. Automat. Mag.*, 18(4) :80–92, 2011.

[Shotton *et al.*, 2011] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*. IEEE, June 2011.

[Sigal *et al.*, 2010] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva : Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Int. J. Comput. Vision*, 87(1-2) :4–27, March 2010.

[Spinello *et al.*, 2010] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart. A layered approach to people detection in 3d range data. In *Proc. of The AAAI Conference on Artificial Intelligence : Physically Grounded AI Track (AAAI)*, 2010.

[Steinbruecker *et al.*, 2011] F. Steinbruecker, J. Sturm, and D. Cremers. Real-time visual odometry from dense RGB-D images. In *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[Stoll *et al.*, 2011] Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 951–958, Washington, DC, USA, 2011. IEEE Computer Society.

[Sturm *et al.*, 2012a] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[Tea,] Tea captiv motion. `http://teaergo.com/`.

[The Captury,] The captury pure performance. `http://www.thecaptury.com/`.

[Thomesse *et al.*, 2001] Jean-Pierre Thomesse, David BELLOT, Anne Boyer, Eric Campo, Marie Chan, François Charpillet, Jocelyne Fayn, Claire Leschi, Norbert Noury, Vincent Rialle, Laurent Romary, Paul Rubel, and François Steenkeste. Integrated information technologies for patients remote follow-up and homecare. In *3rd International Workshop on Enterprise Networking and Computing in Healthcare - HealthCom 2001*, l'Aquila/Italie, June 2001. Congrès (éditeur). internationale.

[Torr and Zisserman, 2000a] P. H. S. Torr and A. Zisserman. MLESAC : A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78, 2000.

[Torr and Zisserman, 2000b] P. H. S. Torr and A. Zisserman. MLESAC : A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78 :138–156, 2000.

[Torr, 2002] Philip H. S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1) :35–61, 2002.

[Triggs *et al.*, 2000] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–375. Springer Verlag, 2000.

[Tykkala *et al.*, 2011] T. Tykkala, C. Audras, and A. I. Comport. Direct iterative closest point for real-time visual odometry. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2050–2056, Nov 2011.

[Vedaldi and Soatto, 2008] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

[Vicon,] Motion capture system vicon. `http://www.vicon.com/`.

[Viola and Jones, 2001] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.

[Wei *et al.*, 2012] Xiaolin Wei, Peizhao Zhang, and Jinxiang Chai. Accurate realtime full-body motion capture using a single depth camera. *ACM Trans. Graph.*, 31(6) :188 :1–188 :12, November 2012.

[Whelan *et al.*, 2013b] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J.B. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Karlsruhe, Germany, May 2013.

[XSens,] 3d motion tracking xsens. `https://www.xsens.com`.

[Zhang and Kanade, 1998] Zhengyou Zhang and T. Kanade. Determining the epipolar geometry and its uncertainty : A review. *International Journal of Computer Vision*, 27 :161–195, 1998.

[Zhang *et al.*, 2012] L. Zhang, J. Sturm, D. Cremers, and D. Lee. Real-time human motion tracking using multiple depth cameras. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[Zhang *et al.*, 2013] Hao Zhang, C. Reardon, and L.E. Parker. Real-time multiple human perception with color-depth cameras on a mobile robot. *IEEE Transactions on Cybernetics*, 43(5) :1429–1441, Oct 2013.

Résumé

Dans cette thèse nous intéressons à la conception d'un robot mobile capable d'analyser le comportement et le mouvement d'une personne en environnement intérieur et encombré, par exemple le domicile d'une personne âgée. Plus précisément, notre objectif est de doter le robot des capacités de perception visuelle de la posture humaine de façon à mieux maîtriser certaines situations qui nécessitent de comprendre l'intention des personnes avec lesquelles le robot interagit, ou encore de détecter des situations à risques comme les chutes ou encore d'analyser les capacités motrices des personnes dont il a la garde. Le suivi de la posture dans un environnement dynamique et encombré relève plusieurs défis notamment l'apprentissage en continue du fond de la scène et l'extraction la silhouette qui peut être partiellement observable lorsque la personne est dans des endroits occultés. Ces difficultés rendent le suivi de la posture une tâche difficile. La majorité des méthodes existantes, supposent que la scène est statique et la personne est toujours visible en entier. Ces approches ne sont pas adaptées pour fonctionner dans des conditions réelles.

Nous proposons, dans cette thèse, un nouveau système de suivi capable de suivre la posture de la personne dans ces conditions réelles. Notre approche utilise une grille d'occupation avec un modèle de Markov caché pour apprendre en continu l'évolution de la scène et d'extraire la silhouette, ensuite un algorithme de filtrage particulaire hiérarchique est utilisé pour reconstruire la posture. Nous proposons aussi un nouvel algorithme de gestion d'occlusion capable d'identifier et d'exclure les parties du corps cachées du processus de l'estimation de la pose. Finalement, nous avons proposé une base de données contenant des images RGB-D avec la vérité-terrain dans le but d'établir une nouvelle référence pour l'évaluation des systèmes de capture de mouvement dans un environnement réel avec occlusions. La vérité-terrain est obtenue à partir d'un système de capture de mouvement à base de marqueur de haute précision avec huit caméras infrarouges. L'ensemble des données est disponible en ligne. La deuxième contribution de cette thèse, est le développement d'une méthode de localisation visuelle à partir d'une caméra du type RGB-D montée sur un robot qui se déplace dans un environnement dynamique. En effet, le système de capture de mouvement que nous avons développé doit équiper un robot se déplaçant dans une scène. Ainsi, l'estimation de mouvement du robot est importante pour garantir une extraction de silhouette correcte pour le suivi. La difficulté majeure de la localisation d'une caméra dans un environnement dynamique, est que les objets mobiles de la scène induisent un mouvement supplémentaire qui génère des pixels aberrants. Ces pixels doivent être exclus du processus de l'estimation du mouvement de la caméra. Nous proposons ainsi une extension de la méthode de localisation dense basée sur le flux optique pour isoler les pixels aberrants en utilisant l'algorithme de RANSAC.

Mots-clés: Capture de mouvement, Odométrie visuelle, Filtrage particulaire, Modèle de Markov Caché, Caméra RGB-D. flux optique

Abstract

In this thesis we are interested in designing a mobile robot able to analyze the behavior and movement of a a person in indoor and cluttered environment. Our goal is to equip the robot by visual perception capabilities of the human posture to better analyze situations that require understanding of person with which the robot interacts, or detect risk situations such as falls or analyze motor skills of the person. Motion capture in a dynamic and crowded environment raises multiple challenges such as learning the background of the environment and extracting the silhouette that can be partially observable when the person is in hidden places. These difficulties make motion capture difficult. Most of existing methods assume that the scene is static and the person is always fully visible by the camera. These approaches are not able to work in such realitsit conditions.

In this thesis, We propose a new motion capture system capable of tracking a person in realistic world conditions. Our approach uses a 3D occupancy grid with a hidden Markov model to continuously learn the changing background of the scene and to extract silhouette of the person, then a hierarchical particle filtering algorithm is used to reconstruct the posture. We propose a novel occlusion management algorithm able to identify and discards hidden body parts of the person from process of the pose estimation. We also proposed a new database containing RGBD images with ground truth data in order to establish a new benchmark for the assessment of motion capture systems in a real environment with occlusions. The ground truth is obtained from a motion capture system based on high-precision marker with eight infrared cameras. All data is available online. The second contribution of this thesis is the development of a new visual odometry method to localize an RGB-D camera mounted on a robot moving in a dynamic environment. The major difficulty of the localization in a dynamic environment, is that mobile objects in the scene induce additional movement that generates outliers pixels. These pixels should be excluded from the camera motion estimation process in order to produce accurate and precise localization. We thus propose an extension of the dense localization method based on the optical flow method to remove outliers pixels using the RANSAC algorithm.

Keywords: Motion capture, Visual odometry, Particle filter, Hidden Markov Model, RGB-D camera, optical flow.