Suivi automatique d'objets 3D basé sur l'apparence dans des séquences d'images

Florent Duculty

N° d'ordre : D. U. 1429

EDSPIC : 279

Université Blaise Pascal - Clermont II

Ecole Doctorale

Sciences Pour l'Ingénieur de Clermont-Ferrand

Formation Doctorale :

Electronique et Systèmes

Thèse

présentée par

Florent DUCULTY

Ingénieur du CUST

pour obtenir le grade de

Docteur d'Université

SPÉCIALITÉ : Vision pour la Robotique

Suivi automatique d'objets 3D basé sur l'apparence dans des séquences d'images

---

Soutenue publiquement le 11 Juillet 2003 devant le Jury :

Monsieur	Jean-Paul GERMAIN	Président
Madame	Françoise PRETEUX	Rapporteur
Monsieur	Jean-Pierre COCQUEREZ	Rapporteur
Monsieur	Michel DHOME	Directeur de thèse
Monsieur	Frédéric JURIE	Examinateur
Monsieur	Jean GALLICE	Examinateur

A ma Mère Gisèle,

à ma Soeur Olivia,

à mon Grand-Père Marcel,

à Mayté.

Remerciements

Les travaux présentés dans ce mémoire ont été réalisés au LAboratoire des Sciences et Matériaux pour l'Electronique, et d'Automatique de l'Université Blaise Pascal, Unité Mixte de Recherche 6602 du CNRS, à Clermont-Ferrand. Je remercie Monsieur Jean-Paul GERMAIN, Professeur à l'Université Blaise Pascal et directeur du LASMEA, pour m'avoir accueilli. Je le remercie également pour avoir accepté la présidence de mon jury de thèse.

Je remercie l'ensemble des membres du jury pour l'intérêt qu'ils ont manifesté à évaluer et juger ce travail de recherche et tout particulièrement Madame Françoise PRETEUX, Professeur à l'Institut National des Télécommunications et Monsieur Jean-Pierre COCQUEREZ, Professeur à l'Université de Technologie de Compiègne, en acceptant d'en être les rapporteurs. Je remercie également Monsieur Jean GALLICE, Professeur à l'Université Blaise Pascal, pour avoir accepté d'être un des examinateurs.

J'exprime ma sincère reconnaissance à mon directeur de thèse, Monsieur Michel DHOME, Directeur de Recherche au CNRS, pour ses conseils, son aide et sa disponibilité dont il a toujours fait preuve malgré ses responsabilités au sein du laboratoire. Je remercie particulièrement Monsieur Frédéric JURIE, Chargé de Recherche au CNRS, responsable de ces travaux, pour ses précieux conseils.

Mes remerciements vont également à l'ensemble des étudiants et doctorants du laboratoire, en particulier à Messieurs Jérôme BRUNET et Lionel MAZET pour leur dévouement et leur sincère amitié mais également à Messieurs Thierry CROZET et Stéphane LECARPENTIER, stagiaires CNAM, que je continue à côtoyer. Je remercie aussi les permanents du laboratoire, trop nombreux pour être cités nommément, grâce à qui j'ai travaillé dans un environnement scientifique de qualité et une ambiance amicale.

Enfin, merci à l'ensemble de mes proches pour le soutien qu'ils m'ont apporté durant cette thèse.

Résumé

Ce travail de thèse s'inscrit dans le domaine de la vision artificielle. Plus précisément, nous nous sommes intéressés au suivi temps réel vidéo d'objets 3D mobiles dans des séquences d'images. A l'origine de ces travaux, se trouve un algorithme, développé au LASMEA, dédié au suivi de motifs planaires texturés.

Nous nous sommes proposés d'adapter cette méthode de suivi 2D à l'estimation du mouvement apparent d'objets 3D. Pour cela, l'objet 3D est modélisé à l'aide d'une collection d'images de référence. Pour chacune de ces vues, la solution 2D citée précédemment permet de suivre les mouvements *fronto parallèles* (déplacement de l'objet parallèlement au plan image) qui ne modifient pas de façon majeure l'aspect apparent dans l'image. Le point délicat, solutionné dans le cadre de cette thèse, est la détection et la gestion du changement d'aspect du motif suivi dû à des rotations relatives (caméra/objet) en site et azimut.

Sur le plan pratique, l'approche proposée a permis le développement d'un système expérimental de suivi de visage et la navigation automatique d'un bras robotique, muni d'une caméra embarquée, autour d'un objet 3D.

Mots-clés : vision artificielle, suivi d'objets 3D, apparence, temps réel vidéo, visioconférence, robotique manufacturière.

Abstract

This thesis is concerned with the computer vision. More precisely, we are interested in the video real time tracking of 3D objects in video sequences, using only object appearances. An algorithm, developed earlier, allows to track textured planar patterns. It constitutes the core of our approach.

We propose to adapt this 2D tracking method to the estimation of the 3D objects movement. For that, the 3D object is represented by a collection of reference images. For each one of these views, the 2D solution, quoted previously, allows to track *fronto parallel* movements (movement of the object in a plane parallel to the image plane). The aspect of the pattern representing the tracked object is not really modified by this motion. The delicate point, solved during this thesis, is the detection and the management of appearance changes of the tracked pattern due to relative rotations (between camera and object) in roll and pitch.

In practice, the suggested approach allowed us the development of an experimental system dedicated to the tracking of human faces and the automatic navigation of a robotic arm, with a camera on its effector, around a 3D object.

Key-words : computer vision, tracking of 3D objects, appearance, video real time, visioconference, manufacturing robotics.

Table des matières

Introduction **1**

1 Introduction au suivi d'objets en vision artificielle **9**

1.1 Notion de suivi pour les machines intelligentes 9

1.2 Etude d'un algorithme de suivi par vision 11

1.2.1 Rappel sur le principe du suivi 11

1.2.2 Décomposition d'un algorithme de suivi 11

1.3 Les modèles de mouvement 12

1.3.1 Modélisation du mouvement par une transformation dans l'espace 3D . . 12

1.3.2 Modélisation du mouvement 2D apparent 14

1.4 Le filtrage ou l'estimation de l'état 16

1.4.1 Vecteur d'état X_i 17

1.4.2 Matrice de covariance P_i 17

1.4.3 Le filtre de Kalman 18

1.4.4 Le filtre à particules 21

1.5 Présentation des méthodes de suivi d'objets 22

1.5.1 Suivi basé sur le mouvement 22

1.5.2 Notion de mise en correspondance d'un modèle 24

1.5.3 Suivi basé sur la modélisation d'objets 26

1.6 Introduction aux travaux présentés dans ce mémoire 29

2 Suivi efficace d'un motif visuel 2D : approche par hyperplans **33**

2.1 Suivi efficace d'une région dans l'image 33

2.2 Approximation Jacobienne (AJ) contre Approximation par Hyperplans (AH) . . 35

2.2.1 Approximation Jacobienne (AJ) [60] 35

2.2.2 Approximation par Hyperplans (AH) [67] [68] 36

2.2.3 Interprétation géométrique des deux approximations 37

2.3 Avantages de l'Approximation par Hyperplans (AH) . . . 39
2.3.1 Simple application numérique . . . 40
2.4 Suivi efficace à l'aide de l'approximation par hyperplans . . . 41
2.4.1 Phase d'apprentissage . . . 42
2.4.2 Phase de suivi . . . 43
2.4.3 Modèles linéaires de mouvement . . . 44
2.5 Expérimentations et Résultats . . . 45
2.5.1 Comparaison entre AJ et AH : résultats sur une image statique . . . 46
2.5.2 Expérimentations sur des objets en mouvement . . . 47
2.6 Traitement des occultations . . . 52
2.6.1 Méthode de seuillage adaptatif [46] . . . 53
2.6.2 Essais . . . 56
2.7 Conclusion . . . 58

3 Extension de l'approximation par hyperplans au suivi 3D d'objets 61
3.1 Présentation de la solution de suivi 3D . . . 62
3.2 Modélisation de l'apparence de l'objet 3D . . . 64
3.2.1 Construction du modèle d'un objet 3D . . . 64
3.2.2 Représentation de l'apparence d'un motif . . . 67
3.3 Suivi 3D d'un objet . . . 69
3.3.1 Paramétrisation des mouvements possibles d'un objet dans l'image . . . 70
3.3.2 Interprétation géométrique du suivi 3D . . . 71
3.3.3 Estimation des matrices d'interaction pour un motif de référence donné . 74
3.3.4 Gestion du passage d'un motif de référence à l'autre . . . 83
3.3.5 Optimisation de la taille du motif suivi dans l'image . . . 90
3.4 Expérimentations . . . 96
3.4.1 Initialisation du suivi . . . 96
3.4.2 Suivi d'objets texturés 3D . . . 97
3.5 Conclusion et perspectives . . . 105

4 Applications dédiées au suivi 3D 107
4.1 Détection et suivi 3D d'un visage dans une séquence vidéo . . . 108
4.1.1 Etat de l'art . . . 109
4.1.2 Solution retenue pour le suivi 3D d'un visage . . . 111

4.1.3 Simulation et Expérimentation temps réel 120
4.2 Commande en position "plus ou moins" de la table à déplacement micrométrique 130
4.2.1 Solution retenue pour la commande en position "plus ou moins" 131
4.2.2 Simulation de l'algorithme de suivi 3D pour la commande en position "plus ou moins" 133
4.2.3 Expérimentation de l'algorithme de commande en position "plus ou moins" de la table à déplacement micrométrique 136
4.3 Commande en position "plus ou moins" d'un bras robotique autour d'un objet connu 140
4.3.1 Présentation générale de l'application 140
4.3.2 Commande d'un bras manipulateur par vision 143
4.3.3 Utilisation de l'algorithme de suivi 3D pour la commande en position "plus ou moins" 145
4.3.4 Simulation de l'algorithme de suivi 3D pour la navigation en position . . 150
4.3.5 Expérimentation de l'algorithme de navigation autour d'un objet 153
4.4 Commande en vitesse d'un bras robotique 158
4.4.1 Solution retenue pour la commande en vitesse 160
4.4.2 Simulation de l'algorithme de suivi 3D pour la commande en vitesse . . . 163
4.4.3 Expérimentation de l'algorithme de commande en vitesse du bras robotique 166
4.5 Conclusion 175

Conclusion **177**

A Rappels sur la résolution des systèmes d'équations linéaires **181**
A.1 Formulation matricielle d'un système d'équations linéaires 181
A.1.1 Cas d'une matrice régulière 182
A.1.2 Cas d'une matrice singulière 183
A.2 Résolution des systèmes d'équations linéaires 184
A.2.1 Méthode de Gauss-Jordan 184
A.2.2 Décomposition **LU** 184
A.2.3 Décomposition **QR** 185
A.2.4 Décomposition en valeurs singulières **SVD** 186
A.2.5 Méthode de Cholesky 187

B Parallélisation d'un algorithme de suivi 3D à l'aide de squelettes fonctionnels 189

B.1 Objectif de la parallélisation . . . 191

B.2 Généralités sur la parallélisation d'algorithmes par squelettes fonctionnels . . . 193

B.2.1 Définition d'un squelette de parallélisation . . . 193

B.2.2 Programmation d'une application parallèle . . . 194

B.3 Parallélisation de l'algorithme de suivi 3D avec imbrication de squelettes . . . 197

B.3.1 Principe de la parallélisation . . . 197

B.3.2 Implantation . . . 199

B.4 Résultats de l'expérimentation . . . 200

B.5 Discussion et conclusion . . . 203

C Notions de robotique 207

C.1 Introduction . . . 207

C.2 Rappels sur l'attitude d'un repère par rapport à un autre . . . 209

C.2.1 Les degrés de liberté d'un solide . . . 209

C.2.2 Les degrés de liberté d'un robot . . . 209

C.2.3 Attitude d'un repère par rapport à un autre . . . 209

C.3 Modélisation d'un robot . . . 211

C.3.1 Les différents modèles composant un robot . . . 211

C.3.2 Le modèle géométrique . . . 213

C.4 Asservissement visuel . . . 217

C.4.1 Capteurs proprioceptifs et extéroceptifs . . . 217

C.4.2 Informations visuelles et matrice d'interaction . . . 218

C.4.3 Notion de fonction de tâche . . . 219

C.4.4 Formalisme de la Commande Référencée Capteur . . . 221

C.4.5 Les différentes approches en asservissement visuel . . . 221

C.4.6 Asservissement en situation ou asservissement dans l'espace opérationnel du robot . . . 222

C.4.7 Asservissement dans l'image . . . 225

C.5 Plate-forme robotique du LASMEA . . . 226

C.5.1 Le robot "AFMA Robots" . . . 226

C.5.2 Le rack VME et la station de travail Silicon Graphics O_2 . . . 228

Bibliographie 229

Table des figures

1 exemple d'objets que l'on souhaite suivre. 3

2 correction des mouvements fronto parallèles du motif suivi dans l'image. 5

3 calculs des angles de site α et d'azimut β en fonction du changement d'aspect du motif courant suivi. 5

1.1 transformation 3D d'un objet. 13

1.2 transformation 2D d'un objet. 15

1.3 représentation de la densité de probabilité (*filtre de Kalman*). 18

1.4 représentation de la densité de probabilité (*filtre à particules*). 21

1.5 illustration du principe de suivi 2D. 30

1.6 illustration du principe de suivi 3D. 30

2.1 interprétation géométrique de l'approximation Jacobienne. 38

2.2 interprétation géométrique de l'approximation par hyperplans. 39

2.3 comparaison entre l'approximation Jacobienne et l'approximation par hyperplans. 41

2.4 phase d'apprentissage hors ligne. 42

2.5 phase de suivi en ligne. 43

2.6 image test et région cible. 46

2.7 comparaison des méthodes AJ et AH sur un modèle de mouvement de type TRE. 47

2.8 phase d'apprentissage multiéchelle hors ligne. 48

2.9 suivi 2D temps réel : séquence 1. 49

2.10 suivi 2D temps réel : séquence 2. 50

2.11 suivi 2D temps réel : séquence 3. 51

2.12 suivi 2D temps réel : séquence 4. 51

2.13 principe du traitement des occultations dans la phase de suivi. 55

2.14 recherche des Inliers et des Outliers dans le vecteur de résidus ΔVI. 56

2.15 quelques exemples de détection d'occultations. 57

3.1 principe du suivi 3D d'objets. 62
3.2 exemple de construction du modèle d'un objet 3D. 64
3.3 mouvements autorisés par la table à déplacement micrométrique. 65
3.4 présentation complète de la table à déplacement micrométrique. 65
3.5 acquisition d'une collection d'images 2D pour la modélisation d'objets. 66
3.6 échantillonnage du motif à l'intérieur d'une ellipse. 67
3.7 robustesse de l'algorithme de suivi aux variations de luminance. 68
3.8 mouvements possibles d'un objet dans l'image. 71
3.9 principe du suivi d'un motif de référence. 72
3.10 interprétation géométrique des paramètres de l'ellipse dans le suivi 3D. 73
3.11 positionnement du motif courant dans la collection d'images 2D suivant les variations d'aspect. 74
3.12 ambiguïté dans la reconnaissance du mouvement détecté. 75
3.13 perturbations des paramètres de l'ellipse pour l'estimation de la *matrice d'interaction A*. 76
3.14 sélection des vues nécessaires au calcul de la *matrice d'interaction B*. 79
3.15 perturbations et corrections des paramètres de l'ellipse pour l'estimation de la *matrice d'interaction B*. 80
3.16 résultats sur l'estimation de l'angle de site α à partir de la *matrice d'interaction B* pour un objet de forme cylindrique (canette de soda). 82
3.17 résultats sur l'estimation de l'angle de site α à partir de la *matrice d'interaction B* pour un objet de forme quelconque (figurine). 83
3.18 exemple de 3 motifs de référence consécutifs pour des paramètres de transformation géométrique différents. 84
3.19 correction intermédiaire des paramètres de l'ellipse lors du changement de motif de référence pendant le suivi (objet centré). 85
3.20 erreurs sur les paramètres actualisés de l'ellipse lors du changement de motif de référence pendant le suivi (objet centré). 87
3.21 corrections intermédiaires des paramètres de l'ellipse lors du changement de motif de référence pendant le suivi (objet non centré). 90
3.22 principe d'acquisition sur 3 niveaux des différentes collections d'images de référence. 93
3.23 exemple de suivi d'une canette de soda sur 3 niveaux d'apprentissage (ellipses blanche (niveau 1), bleue (niveau 2) et rouge (niveau 3)). 95
3.24 principe d'initialisation de la phase de suivi. 97

3.25 différentes informations temps réel calculées par l'algorithme de suivi d'objets 3D. 99
3.26 suivi 3D temps réel d'une canette de soda. 100
3.27 suivi 3D temps réel d'un cube : séquence 1. 101
3.28 suivi 3D temps réel d'un cube : séquence 2. 102
3.29 suivi 3D temps réel d'une figurine : séquence 1. 103
3.30 suivi 3D temps réel d'une figurine : séquence 2. 104

4.1 présentation des applications de suivi 3D. 108
4.2 principe du suivi 3D de visages. 112
4.3 modélisation 3D du visage et échantillonnage du motif dans une ellipse. 113
4.4 sauvegarde des résultats du suivi de 2 motifs de référence consécutifs dans une image intermédiaire de la base d'apprentissage. 114
4.5 estimation des positions relatives des ellipses (p) et (s) par rapport à la position prédite de l'ellipse (c). 115
4.6 rappel sur la définition des paramètres caractérisant l'ellipse. 116
4.7 rappel du principe de suivi 2D avec la matrice d'interaction A. 117
4.8 utilisation des vues intermédiaires de la base d'apprentissage pour le positionnement des ellipses prédites dans la phase de suivi. 118
4.9 informations nécessaires pour le calcul du positionnement des ellipses prédites dans la phase de suivi. 118
4.10 résultats de l'algorithme de suivi sur l'ensemble des 71 vues de la base d'apprentissage (de la première image vers la dernière image). 121
4.11 résultats de l'algorithme de suivi sur l'ensemble des 71 vues de la base d'apprentissage (de la dernière image vers la première image). 122
4.12 présentation de l'application automatique de suivi 3D d'un visage. 123
4.13 visualisation des différents résultats issus de l'algorithme de suivi. 124
4.14 détection et reconnaissance d'un motif de référence dans la base d'apprentissage. 125
4.15 suivi 3D d'un visage : séquence 1. 126
4.16 suivi 3D d'un visage : séquence 2. 127
4.17 comportement de l'algorithme aux variations d'éclairement. 128
4.18 images compressées pour différentes qualités de rendu. 130
4.19 modélisation de la figurine pour une variation en site de 180 degrés. 131
4.20 commande en position "plus ou moins" de la table à déplacement micrométrique. 131
4.21 position caméra/objet lors de la commande en site de la table. 132

4.22 résultats de l'algorithme de suivi sur l'ensemble des 181 vues de la base d'apprentissage (de la première image vers la dernière image). 133
4.23 résultats de l'algorithme de suivi sur l'ensemble des 181 vues de la base d'apprentissage (de la dernière image vers la première image). 134
4.24 phase de détection automatique d'un motif dans l'image. 136
4.25 visualisation des résultats lors de la commande en position de la table. 137
4.26 résultats de la commande en position de la table à déplacement micrométrique. . 138
4.27 résultats de l'algorithme de suivi pour la commande en position de la table lors de l'expérimentation. 139
4.28 définition de la trajectoire du bras robotique autour de l'objet 3D. 141
4.29 acquisition de la collection d'images 2D à l'aide du bras robotique. 142
4.30 éclairage artificiel de la scène. 143
4.31 trois types de configuration de la position de la caméra. 144
4.32 commande en position "plus ou moins" du robot portique AFMA. 146
4.33 estimation des coordonnées courantes de l'effecteur (caméra) dans le repère physique du robot (repère absolu). 147
4.34 coordonnées 3D d'un point sur une sphère de rayon unité. 149
4.35 résultats de l'algorithme de suivi sur l'ensemble des 121 vues de la base d'apprentissage (de la première image vers la dernière image). 151
4.36 résultats de l'algorithme de suivi sur l'ensemble des 121 vues de la base d'apprentissage (de la dernière image vers la première image). 152
4.37 principe de l'algorithme de navigation du bras robotique autour d'un objet connu. 154
4.38 visualisation des résultats lors de la commande en position du robot. 155
4.39 exemple de résultats de la commande en position du robot. 156
4.40 résultats de l'algorithme de navigation en condition réelle. 157
4.41 modélisation du globe lumineux pour une variation en site de 140 degrés. 159
4.42 commande en vitesse du robot portique AFMA. 160
4.43 définition des paramètres de l'ellipse utilisés comme consigne. 161
4.44 résultats de l'algorithme de suivi sur l'ensemble des 141 vues de la base d'apprentissage (de la première image vers la dernière image). 164
4.45 résultats de l'algorithme de suivi sur l'ensemble des 141 vues de la base d'apprentissage (de la dernière image vers la première image). 165
4.46 phase de détection automatique d'un motif dans l'image. 166
4.47 visualisation des résultats lors de la commande en vitesse du robot. 167

4.48 résultats de la commande en vitesse du robot portique : séquence 1. 169
4.49 résultats de la commande en vitesse du robot portique : séquence 2. 170
4.50 résultats du changement de motif de référence donnés par l'algorithme de suivi 3D en condition réelle. 171
4.51 écarts relatifs de position entre les deux ellipses (courante et consigne). 172
4.52 orientation et échelle de l'ellipse courante. 173
4.53 composantes du torseur cinématique. 174

B.1 phase de suivi 3D en ligne à paralléliser. 190
B.2 algorithme séquentiel de suivi 3D à paralléliser. 191
B.3 algorithme de suivi 3D parallélisé. 192
B.4 exemple d'implantation du *squelette SCM* sur n processeurs ($n = 4$). 196
B.5 structure générale de la version parallèle de l'algorithme de suivi. 198
B.6 temps d'exécution en ms pour 170 et 373 points d'échantillonnage. 201
B.7 accélération pour 170 et 373 points d'échantillonnage. 202
B.8 représentation interne de l'application de suivi d'un visage. 205
B.9 prototypes des fonctions utilisateur pour l'application de suivi d'un visage. . . . 206

C.1 robot manipulateur dans un atelier de stockage. 207
C.2 robots dédiés à l'exploration du système solaire. 208
C.3 changements de repère. 210
C.4 modèle complet d'un robot. 212
C.5 schéma du robot cartésien à six d.d.l.. 213
C.6 référence robot et repère caméra. 214
C.7 convention de Denavit-Hartenberg. 215
C.8 tâche de suivi de cible mobile. 219
C.9 tâche de positionnement par rapport à une cible fixe. 220
C.10 tâche de navigation entre plusieurs cibles. 220
C.11 asservissement en situation (3D) pour une commande en vitesse. 223
C.12 comparaison Position/Image - Based Look and Move. 223
C.13 asservissement dans l'image (2D) pour une commande en vitesse. 225
C.14 vue générale de la plate-forme robotique du LASMEA. 226
C.15 caméra embarquée. 227
C.16 rack VME. 228

Introduction

L'espace qui nous entoure a une structure tri-dimensionnelle (3D). Lorsque l'on demande à une personne de décrire ce qu'elle *voit*, elle n'éprouve aucune difficulté à nommer les objets qui l'entourent : téléphone, table, livre... Et pourtant l'information qui est réellement disponible sur la rétine de ses yeux, n'est, ni plus ni moins, une collection de points (environ un million !). En chaque point ou *pixel* (picture element), il y a tout simplement une information qui donne une indication quant à la quantité de lumière et la couleur qui proviennent de l'espace environnant et qui ont été projetées à cet endroit de la rétine. Le téléphone, la table ou le livre *n'existent pas* sur la rétine. Guidé à la fois par l'information codée dans l'image (la rétine) et par ses propres connaissances, le processus visuel *construit* des percepts (objet dont la représentation nous est donnée par la perception sensorielle). Le téléphone ou le livre sont les réponses finales, résultant d'un *processus d'interprétation* qui fait partie intégrante du système de vision. De plus, il n'y a pas de correspondance terme à terme entre l'information sensorielle (la lumière et la couleur) et la réponse finale (des objets 3D). Le système de vision doit *fournir* les connaissances nécessaires afin de permettre une interprétation non ambiguë.

La vision a suscité l'intérêt de nombreux scientifiques et philosophes depuis déja très longtemps. Parmi ceux-ci, les neurobiologistes mènent des recherches théoriques et expérimentales afin d'essayer de comprendre l'anatomie et le fonctionnement du cerveau dans son ensemble. Ils ont découvert l'une des plus complexes *inventions* de la nature, qui est loin de leur avoir révélé tous ses secrets. On ne connaît pas encore ses limites. Mais ces limites ne sont-elles pas repoussées à chaque découverte ? David Hubel a merveilleusement bien exprimé ce paradoxe : *"Le cerveau peut-il comprendre le cerveau ?"*.

En donnant à la machine la possibilité de voir, l'homme a certainement franchi un pas important dans l'automatisation de ses tâches quotidiennes. Les progrès techniques, tant au niveau des caméras que des systèmes informatiques de traitement des images, ont permis un

élargissement considérable du champ d'application de ce qu'il est convenu d'appeler la *vision artificielle* (dernière née des approches de la recherche sur le mécanisme de la perception visuelle).
En vision artificielle, le paradigme dominant est celui énoncé par David Marr dans son livre *Vision* (W.H. Freeman Company, San Francisco, CA, 1982). Il énonce des principes généraux qui s'appliquent à tous les systèmes visuels (selon lui et d'après les connaissances de l'époque) et une méthodologie de travail pour concevoir et produire des systèmes artificiels.

Le paradigme de David Marr est basé sur trois concepts bien définis :

1. **une notion de modules visuels** : se basant sur les résultats obtenus à l'époque par les psychologues et les psychophysiciens, il définit un certain nombre de modules visuels accomplissant des fonctions de perception telles que la distinction d'arêtes dans une image, l'inférence de la géométrie 3D à partir des contours, la reconnaissance de motifs répétés (texture)...
2. **une hiérarchie des niveaux de représentation de l'information visuelle** : à partir de l'image brute représentée par l'intensité lumineuse en chacun de ses points, il s'agit d'extraire des informations (par exemple des arêtes et des contours) pour arriver à une modélisation 3D de la forme des objets.
3. **une méthodologie de création des systèmes de vision artificielle** : à partir des calculs à réaliser, il s'agit de développer l'algorithme en s'assurant de son implémentation matérielle.

On demande aujourd'hui aux machines intelligentes munies d'un système de vision d'être capables d'interagir avec leur environnement de travail dans des situations réelles, parfois complexes. Cela suppose donc de pouvoir analyser les images pour en extraire les informations importantes, telles que la présence, la position ou le déplacement de certains objets :

- *reconnaître un objet* : c'est déterminer si un objet, parmi un ensemble d'objets possibles, apparaît dans une image (ou une séquence d'images). C'est donc un problème de mise en correspondance (ou d'appariement) entre une base de données contenant des représentations d'objets et des indices visuels extraits de l'image.
- *suivre un objet* consiste à déterminer sa position dans le champ visuel (et éventuellement inférer son attitude spatiale) tout au long d'une séquence d'images. La nature de l'objet ainsi que la position dans la première image de la séquence sont supposées être connues.

Ces problèmes sont trop complexes pour être abordés dans leur globalité et amènent les chercheurs du domaine à agir de manière pragmatique en traitant les problèmes cas par cas, tout en tentant d'accumuler les connaissances obtenues.

Les travaux réalisés dans le cadre de cette thèse se situent dans cette dernière problèmatique. Le sujet qui m'a été proposé est de développer un algorithme *efficace* et *rapide* de suivi d'objets mobiles 3D dans des séquences d'images en se basant sur l'apparence. Cet algorithme doit être aussi *robuste* aux déplacements de forte amplitude de l'objet entre deux images successives de la séquence. Pour simplifier le problème, la localisation de l'objet dans la première image est réalisée par l'opérateur lors de l'initialisation du processus (notion d'interactivité). Ces objets peuvent être de formes et de textures plutôt complexes (figure 1).

FIGURE 1 – exemple d'objets que l'on souhaite suivre.

Les algorithmes de suivi classiques proposés dans la littérature s'exécutent en deux étapes :

1. **une phase de prédiction**, durant laquelle une ou plusieurs hypothèses sur la position de l'objet dans l'image ou sur sa localisation par rapport à la caméra sont formulées.
2. **une phase d'exploration autour de la prédiction**, durant laquelle la position de l'objet dans l'image ou l'attitude précise de l'objet est déterminée.

L'originalité principale de la solution que nous proposons est, pour un domaine applicatif donné, de supprimer cette étape d'exploration et de pouvoir corriger directement la position prédite de l'objet dans l'image grâce à une méthode d'apprentissage hors ligne. Cela est d'autant plus intéressant qu'elle s'avère généralement la plus coûteuse en terme de coût algorithmique et ne permet pas de travailler en temps réel vidéo.

Dans notre approche de suivi, l'objet 3D est représenté par une collection d'images 2D appelées *vues de référence*. Un motif est une région de l'image définie à l'intérieur d'une zone d'intérêt (une ellipse dans notre cas) et son échantillonnage donne un vecteur de niveaux de gris. L'une des caractéristiques de cette méthode est de ne pas utiliser des primitives (exemples :

points d'intérêt, segments de droite, ...) pour suivre le déplacement de l'objet dans l'image mais plutôt la différence de vecteurs de niveaux de gris entre le motif de référence suivi et le motif courant échantillonné dans une zone d'intérêt de l'image. Le problème du suivi se ramène alors à l'estimation des paramètres qui caractérisent les mouvements possibles de l'objet dans l'image par la détermination de *matrices dites d'interaction* apprises lors d'une phase d'apprentissage hors ligne, et cela pour chacune des vues de référence. La première matrice (notée A dans la figure 2) lie la différence de niveaux de gris entre le motif de référence suivi et le motif courant échantillonné dans la zone d'intérêt à son déplacement fronto parallèle (déplacement parallèle au plan image). Sous l'hypothèse d'un tel mouvement, l'aspect apparent de l'objet suivi ne subit pas de modification majeure. Toutefois, sa position, son orientation planaire et sa taille peuvent changer. Pour cela, nous supposons également que la taille de l'objet selon la direction d'observation reste faible par rapport à la distance objet-caméra (utilisation d'une caméra équipée d'un système optique à focale relativement longue pour ne pas induire de trop fortes distorsions liées à la projection perspective). Ainsi, une translation 3D (même selon l'axe optique) de l'objet dans la scène correspondra à une similitude 2D dans l'image (mouvement fronto parallèle). La deuxième matrice d'interaction (notée B dans la figure 3), quant à elle, relie les variations d'apparence du motif suite à un changement d'orientation de l'objet par rapport au capteur (modification des angles de site et d'azimut).

L'utilisation en ligne de ces matrices pour le suivi 3D de l'objet dans l'image correspond à un coût algorithmique très faible (multiplication d'une matrice par un vecteur) permettant une mise en oeuvre temps réel. Cette étape en ligne consiste à prédire la position de l'objet dans l'image (en position, échelle et orientation), à multiplier la différence entre le motif observé à l'endroit prédit avec le motif de référence qui doit être suivi par la première matrice d'interaction A pour corriger les erreurs sur les mouvements fronto parallèles de l'objet dans l'image (figure 2). Le problème du suivi du motif dans l'image se ramène alors à la correction des paramètres d'une transformation géométrique planaire par la détermination d'un vecteur d'offset.

Une nouvelle différence entre le nouveau motif courant (obtenu après application de la correction préalablement mentionnée) et le motif de référence multipliée par la deuxième matrice d'interaction B nous donne les variations d'aspect du motif suivi par rapport au motif de référence le plus proche dans la collection d'images dues aux orientations 3D relatives (site et azimut) de l'objet (figure 3). Nous pouvons ainsi, si nécessaire, prendre la décision de changer

de motif de référence pour continuer à suivre l'objet dans l'image en temps réel vidéo (moins de 20 millisecondes par itération).

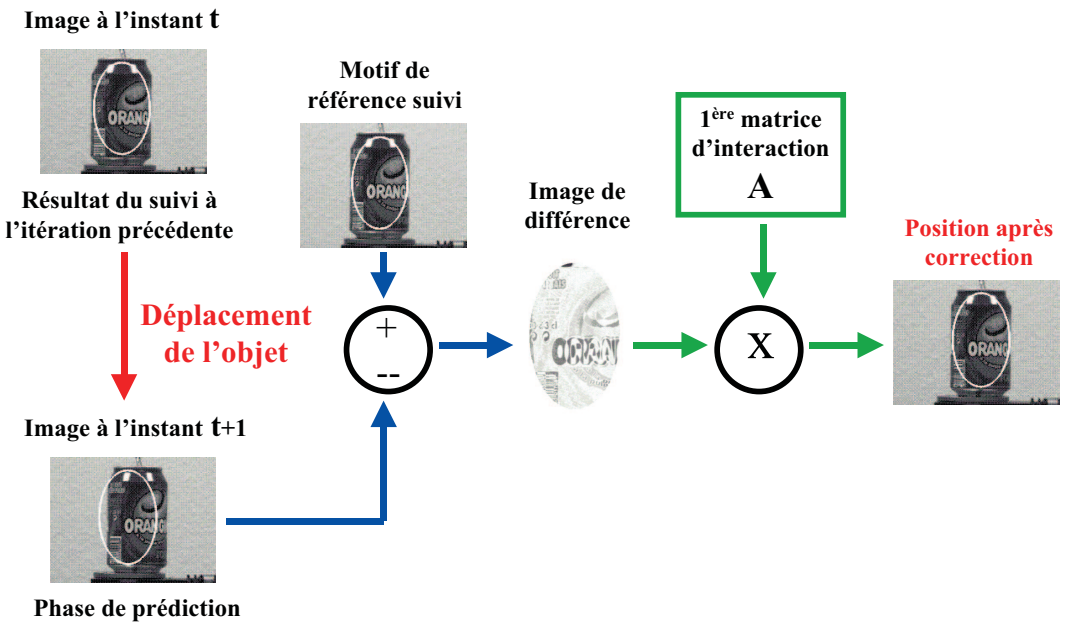


FIGURE 2 – correction des mouvements fronto parallèles du motif suivi dans l'image.

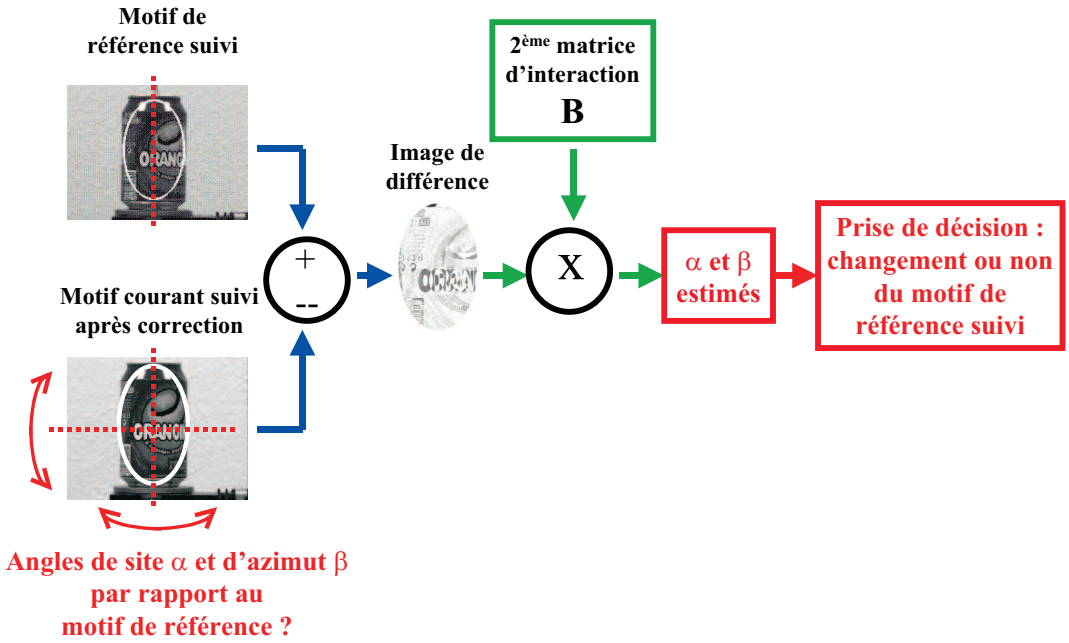


FIGURE 3 – calculs des angles de site α et d'azimut β en fonction du changement d'aspect du motif courant suivi.

Compte tenu de la rapidité des traitements (multiplication d'une matrice par un vecteur) par rapport à la vitesse de déplacement des objets dans les séquences d'images, nous n'avons

pas besoin d'utiliser d'algorithme de prédiction de mouvement. En effet, l'écart de position du motif entre deux images successives reste compatible avec les variations apprises lors de la phase d'apprentissage, dans le cas des applications envisagées.

Ce mémoire de thèse se décompose en quatre chapitres. Tout d'abord, nous posons, de manière générale, le problème du suivi d'objets dans des séquences d'images avant de faire un état de l'art des différentes méthodes de suivi 2D et 3D d'un objet. Ceci nous permettra de positionner nos travaux par rapport à l'ensemble des méthodes déja existantes.

Dans un second chapitre, nous présentons l'algorithme qui permet de suivre le déplacement d'un motif visuel 2D donné dans un flot vidéo sans occultation et qui est à l'origine de mes travaux de recherche. Cette méthode, illustrée par la figure 2, consiste à mesurer l'erreur entre le motif de référence à suivre et le motif observé à l'endroit prédit, et à exploiter cette différence multipliée par une matrice dite *d'interaction* (apprise durant une phase d'apprentissage hors ligne) pour corriger les erreurs entachant la prédiction. Cette méthode de suivi, mettant en relation l'image de différence de niveaux de gris et le mouvement modélisée par des hyperplans, s'avère plus simple et plus efficace que toutes les autres techniques proposées à ce jour. Pour cela, nous allons comparer notre solution aux travaux menés par Hager et Belhumeur [60] qui utilisent l'inverse d'une image Jacobienne pour estimer cette relation linéaire (la matrice d'interaction). Les occultations sont ensuite prises en compte par une méthode de seuillage adaptatif.

Dans un troisième chapitre, nous étendons notre approche 2D au suivi d'objets 3D dont le principe a été illustré par les figures 2 et 3. Tout d'abord, nous voyons comment modéliser l'objet 3D et son apparence, puis nous définissons les paramètres qui caractérisent les mouvements possibles de l'objet dans l'image et leurs interprétations géométriques dans le suivi. Nous terminons par la gestion du passage d'un motif de référence à l'autre qui permet d'assurer le suivi 3D. Cette approche est alors illustrée par des exemples concrets de suivi d'objets volumiques.

Différentes applications 3D, développées au laboratoire, sont ensuite décrites dans le dernier chapitre. Elles sont dédiées, plus particulièrement, aux domaines applicatifs de la visioconférence et de la robotique manufacturière. Le premier algorithme permet la détection automatique d'un visage dans une séquence d'images pour assurer son suivi 3D. En cas de perte du motif suivi, il réinitialise automatiquement l'application en recherchant un nouveau motif de référence

dans l'image. Le deuxième algorithme permet, quant à lui, de commander en position une table à déplacement micrométrique utilisée lors de la création de nos collections d'images 2D pour modéliser les différents objets 3D. A partir du motif suivi dans l'image courante, nous commandons la position angulaire de la table pour atteindre le motif de référence désiré. Cette étape intermédiaire nous a permis d'intégrer, pour la première fois, notre algorithme de suivi 3D dans une boucle d'asservissement avant de travailler sur le robot portique du laboratoire. Pour cette dernière application, deux commandes ont été développées :

- *une commande en position* : à partir du motif suivi dans l'image courante, le bras manipulateur, équipé d'une caméra sur son effecteur, doit naviguer autour d'un objet connu pour atteindre le motif de référence désiré.
- *une commande en vitesse* : le robot suit les déplacements de l'objet devant la caméra de façon à garder au centre de l'image courante le motif suivi tout en gérant ses changements d'aspect.

Pour ces différentes applications, nous n'avions pas besoin d'une estimation précise des angles de site et d'azimut (utilisation de la matrice d'interaction B). C'est pourquoi, nous avons développé une nouvelle méthode pour gérer les changements de motif de référence lors de la phase de suivi. Elle consiste à comparer les erreurs quadratiques de la différence de niveaux de gris entre le motif échantillonné dans l'ellipse corrigée et les différents motifs de référence testés (motif courant suivi et ses plus proches voisins dans la collection d'images de référence). Le motif de référence donnant l'erreur quadratique la plus faible sera alors considéré comme le nouveau motif de référence à suivre dans la prochaine image.

Chapitre 1

Introduction au suivi d'objets en vision artificielle

Dans ce premier chapitre, nous allons traiter, de manière générale, le problème du suivi d'objets dans des séquences d'images. Nous présentons, tout d'abord, la notion de suivi pour les machines intelligentes, puis développons les principales caractéristiques d'un algorithme de suivi d'objets par vision. Dans les sections suivantes, nous reviendrons sur certains points spécifiques d'un algorithme de suivi et ferons un état de l'art des différentes méthodes de suivi 2D et 3D d'un objet. Ceci nous permettra, dans les prochains chapitres de ce mémoire, de positionner nos travaux par rapport à l'ensemble des méthodes déja existantes.

1.1 Notion de suivi pour les machines intelligentes

On demande aux machines intelligentes d'être capables d'interagir avec leur environnement dans des situations réelles, parfois complexes. La vision artificielle peut être, pour la machine, un organe de perception important, qui lui fournit des informations adaptées à la situation, tout au long de l'exécution d'une tâche. Cela suppose de pouvoir analyser les images pour en extraire les informations importantes, telle que la présence, la position ou le mouvement de certains objets.

Le suivi d'objets consiste à déterminer la position (et éventuellement l'attitude) d'un objet tout au long d'une séquence d'images. La nature de l'objet ainsi que sa position dans la première image de la séquence sont supposées connues. Deux problèmes se posent alors pour un système de suivi visuel. Ce sont les problèmes de mouvement et de mise en correspondance :

1. problème du mouvement : il s'agit de prédire la localisation d'un élément image ou une attitude spatiale de l'objet à suivre dans l'image, en utilisant les positions précédentes. Il faut ensuite identifier une région de recherche limitée dans l'image ou dans l'espace des poses dans lequel on s'attend à trouver l'élément avec une forte probabilité.
2. problème de mise en correspondance (aussi connu comme détection ou localisation) : il s'agit ici d'identifier l'élément image à l'intérieur de la région de recherche désignée et de trouver des correspondances en comparant des paires candidates d'éléments images entre :
   - image courante / image suivante,
   - image courante / modèle de l'objet.

Les performances et les caractéristiques d'un algorithme de suivi visuel doivent répondre, quant à elles, aux contraintes suivantes :

1. robustesse au changement de fond qui peut être lui même riche en objets parasites.
2. robustesse aux occultations : l'algorithme ne doit pas perdre la cible suivie lors de l'apparition temporaire d'une occultation partielle. Si l'occultation est totale, il devra être capable de retrouver la cible quand elle apparaîtra de nouveau dans l'image et reprendre correctement son suivi.
3. robustesse aux fausses alarmes : seules les cibles validées devront être classées en tant que tel, et les autres éléments images ignorés (le nombre de fausses alarmes doit être aussi faible que possible).
4. agilité : l'algorithme de suivi doit permettre un déplacement de la cible avec une vitesse et une accélération significatives.
5. stabilité : la précision du suivi doit être maintenue indéfiniment au cours du temps.
6. coût algorithmique du calcul : il doit rester compatible avec les temps d'exécution demandés dans les applications robotiques (temps réel si possible).

Le problème du suivi est trop complexe pour être traité dans sa globalité. Cette complexité amène les chercheurs du domaine de la vision artificielle à agir de manière pragmatique, en examinant les problèmes cas par cas, tout en tentant d'accumuler les connaissances obtenues. Pour l'instant, aucune théorie générale ne semble se dessiner.

1.2 Etude d'un algorithme de suivi par vision

1.2.1 Rappel sur le principe du suivi

Nous avons vu que le suivi en vision artificielle avait pour objectif d'estimer l'évolution de l'état d'un objet au cours du temps. Son estimation est réalisée à partir de mesures (bruitées) sur une séquence d'images. Suivre un objet dans l'image revient donc à optimiser l'estimation du mouvement pour qu'elle puisse correspondre au mieux aux mesures extraites dans l'image. Ceci nécessite la mise en place de deux modèles :

1. *modèle de mouvement* pour décrire l'évolution de l'état dans le temps,
2. *modèle de mesure* (ou *mise en correspondance*) pour relier les mesures dans l'image avec l'état.

1.2.2 Décomposition d'un algorithme de suivi

D'un point de vue algorithmique, le processus permettant cette estimation est cyclique. Il peut se décomposer en plusieurs étapes successives : la prédiction, la mesure, l'observation, la validation et enfin l'estimation. Notons que dans la littérature, il existe deux techniques basées sur ces principes : les *filtres de Kalman* et les *filtres à particules*.

1. la *prédiction* calcule la position la plus probable de l'état, à priori de la cible dans l'image courante (sans les mesures). Cette étape fait appel notamment à la connaissance des états dans les images précédentes ou d'un état initial déterminé par l'intermédiaire d'un processus de détection. Dans le premier cas, la prédiction est réalisée selon le modèle de mouvement choisi, ainsi qu'un modèle d'incertitudes.
2. la *mesure* consiste à estimer une ou plusieurs propriétés dans les images de la séquence, à la position prédite (ou autour de cette position). Ces propriétés sont propres à la représentation ou signature(s) visuelle(s) de l'objet choisie dans les méthodes de suivi développées.
3. l'*observation* consiste à déterminer la position de l'objet à partir de la mesure réalisée précédemment. Cette étape nécessite donc la définition d'un modèle de mesure. Il s'agit de déterminer la position optimisant un ou plusieurs critères de mesure. A noter que, dans certains travaux [32], les deux étapes *mesure* et *observation* sont regroupées en une seule, nommée alors uniquement *observation*.
4. la *validation* examine la validité de la position estimée de l'objet. Ce processus peut utiliser

des mesures dans l'image, la position prédite précédemment calculée ou des connaissances externes liées à l'application visée.

5. l'*estimation* conclue ce cycle en fournissant une estimation de l'état de l'objet, ainsi que les incertitudes éventuelles sur celui-ci. Cette mise à jour tient compte de ou des observations réalisées (si elles sont validées) dans les cycles précédents et du modèle de mouvement choisi. Dans certains systèmes [16], l'*observation* tient lieu d'*estimation*, la position estimée correspondant à la position observée et validée.

Dans les sections suivantes, nous allons revenir sur quelques notions qui se sont dégagées de cette revue algorithmique : les *modèles de mouvement*, les *techniques d'estimation* et enfin les *modèles de mesure* où nous présenterons les différentes méthodes de suivi basées tout d'abord sur le mouvement puis sur une modélisation 2D ou 3D d'un objet. Pour cette seconde approche, cela revient à décrire en premier lieu l'ensemble des variables (les paramètres du modèle en mouvement) que l'on veut estimer, puis les outils utilisés pour cette estimation et enfin le type des mesures sur lesquelles ces outils sont appliqués.

1.3 Les modèles de mouvement

Le choix du *modèle de mouvement* dans les processus de suivi est déterminant. D'une manière générale, la précision dans l'estimation du mouvement dépend du nombre de paramètres. Cependant, un nombre trop important de ces paramètres peut entraîner des instabilités numériques [102], un coût en temps de calcul élevé ou une trop grande sensibilité aux bruits [15].

Nous pouvons distinguer deux approches pour modéliser le mouvement :
- dans la première, le modèle du mouvement est exprimé sous la forme d'une **transformation dans l'espace 3D**.
- dans la seconde, on modélise le **mouvement 2D apparent** de l'objet dans l'image.

1.3.1 Modélisation du mouvement par une transformation dans l'espace 3D

Dans cette première approche, il s'agit d'exprimer le modèle sous la forme d'une **transformation dans l'espace 3D**. Le processus de suivi inclut alors un calcul de pose 3D de l'objet

d'après les mesures effectuées dans l'image. L'estimation de la transformation est réalisée selon ses paramètres.

Pour un objet rigide, il y a six paramètres à estimer : trois pour la translation et trois pour la rotation de l'objet. La transformation est souvent donnée par rapport à un repère de référence fixe. Pour illustrer ceci, considérons la figure 1.1 qui représente un repère R_{obj} lié à l'objet par rapport à un repère référence R_{ref}.

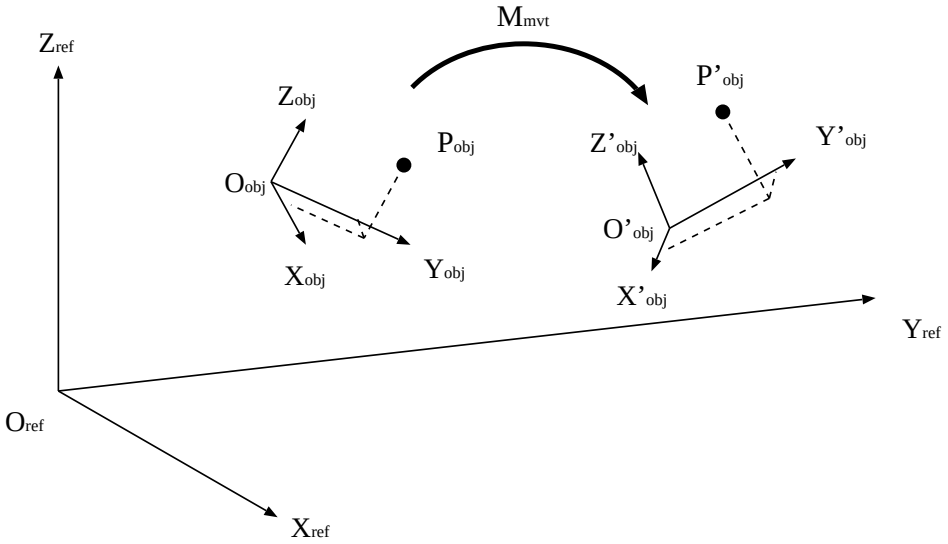


FIGURE 1.1 – transformation 3D d'un objet.

Soit un point $\mathbf{P}_{obj}$ de l'objet de coordonnées $(X_{ref}, Y_{ref}, Z_{ref})$ dans le repère de référence. La transformation, induite par le mouvement de l'objet, telle que $\mathbf{P}_{obj} \rightarrow \mathbf{P'}_{obj}$, peut s'exprimer sous la forme de la relation matricielle suivante (dans le cas d'un objet rigide) :

$$\begin{pmatrix} X'_{ref} \\ Y'_{ref} \\ Z'_{ref} \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{ref} \\ Y_{ref} \\ Z_{ref} \\ 1 \end{pmatrix} \tag{1.1}$$

Les neuf éléments r_{ij} (i=1..3 et j=1..3) paramètrent les rotations et les trois éléments t_i (i=1..3) les translations.

Par exemple, dans une représentation Eulérienne, où (α, β, γ) sont les angles de rotation et (t_x, t_y, t_z) les translations suivant les axes, nous obtenons :

$$\mathbf{M}_{mvt} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{M}_{rot}\mathbf{M}_{trans} \tag{1.2}$$

La *matrice de mouvement* $\mathbf{M}_{mvt}$ est le produit de deux matrices : une *matrice de rotation* $\mathbf{M}_{rot}$ et une *matrice de translation* $\mathbf{M}_{trans}$:

$$\mathbf{M}_{rot} = \mathbf{M}_{\alpha}\mathbf{M}_{\beta}\mathbf{M}_{\gamma} \tag{1.3}$$

avec :

$$\mathbf{M}_{\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{M}_{\beta} = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{M}_{\gamma} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.4}$$

$$\mathbf{M}_{trans} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.5}$$

Le modèle géométrique est ensuite reprojeté dans les images, selon le modèle de la caméra choisi (orthographique, projectif,...), pour déterminer la position des mesures image à effectuer lors de l'*observation*.

1.3.2 Modélisation du mouvement 2D apparent

Cette seconde approche consiste à modéliser le **mouvement 2D apparent** de l'objet. Les paramètres estimés seront donc des paramètres décrivant la trajectoire 2D d'un motif (lié à l'objet). Notons que cette approche n'empêche pas de déterminer la transformation 3D par

un calcul de pose, si un modèle géométrique 3D de l'objet est connu. Mais contrairement à l'approche précédente, ce calcul n'intervient pas à proprement parlé dans le processus de suivi.

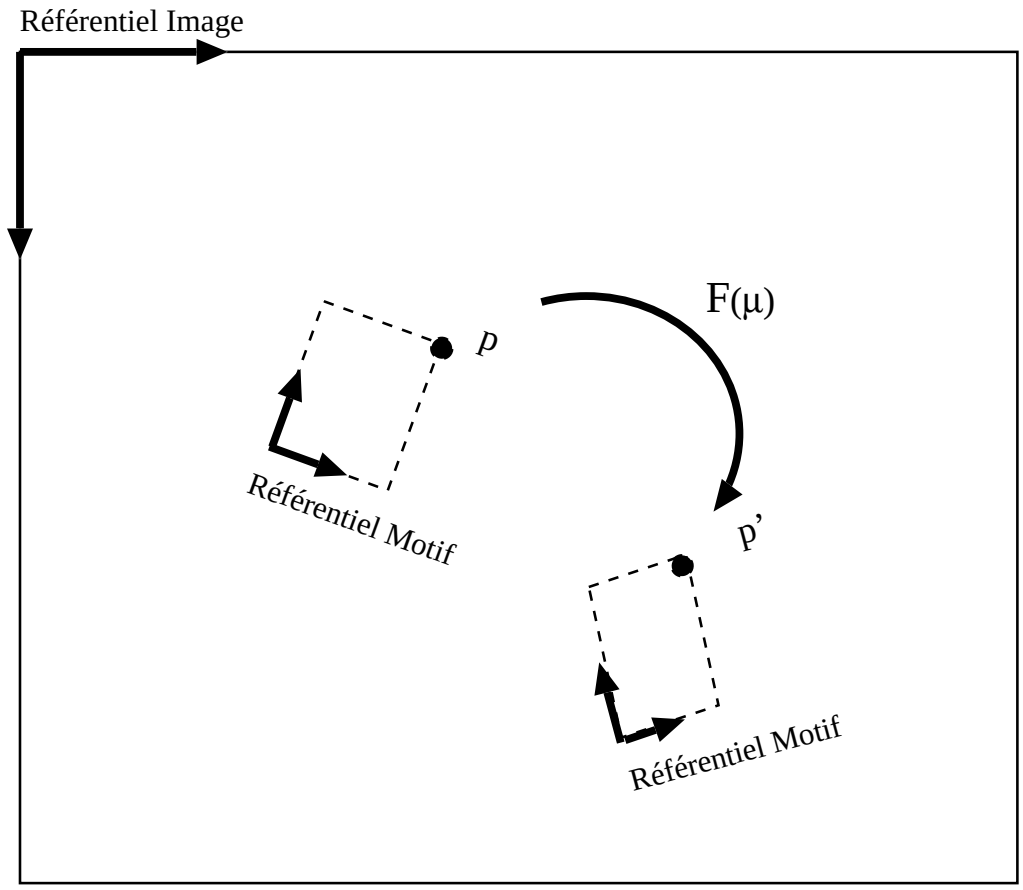


FIGURE 1.2 – transformation 2D d'un objet.

Dans un modèle 2D de mouvement, les paramètres sont souvent regroupés dans un *vecteur de paramètres* noté μ. Ce vecteur décrit la transformation $\mathbf{p} \rightarrow \mathbf{p}'$ d'un point $\mathbf{p}$ de coordonnées (x, y) dans un repère lié au motif, en un point $\mathbf{p}'$ de coordonnées (x', y'). Ceci est illustré par la figure 1.2. Dans la littérature [8] [103], plusieurs modèles paramétriques sont présentés : des modèles linéaires ou non-linéaires. Les modèles linéaires de déplacement sont les plus couramment utilisés. Ils peuvent s'écrire sous la forme d'une matrice homogène paramétrée $\mathbf{F}(\mu)$, telle que :

$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \mathbf{F}(\mu) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.6}$$

De manière hiérarchique, on peut définir :

– la *translation* qui se définit avec deux paramètres (t_x, t_y) tels que :

$$\mathbf{F}(\mu) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

– la *similarité* (translation, rotation planaire et changement d'échelle) se caractérise avec (t_x, t_y) pour la translation, θ pour la rotation planaire et ρ pour le changement d'échelle.

$$\mathbf{F}(\mu) = \begin{bmatrix} \rho cos(\theta) & -\rho sin(\theta) & t_x \\ \rho sin(\theta) & \rho cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.8}$$

– la *transformation affine* intègre six paramètres :

$$\mathbf{F}(\mu) = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.9}$$

– l'*homographie* est un modèle considérant 8 paramètres : $\mu = (a, b, c, d, e, f, g, h)$.

$$\mathbf{F}(\mu) = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \tag{1.10}$$

Les transformations telles que la *translation*, la *similarité* et l'*affinité* considèrent un modèle orthographique de la caméra ; l'*homographie* considère un modèle projectif.

1.4 Le filtrage ou l'estimation de l'état

Pour chaque image, les paramètres du mouvement (et par conséquent l'état de l'objet) sont donc calculés à partir de mesures dans l'image. Ces mesures sont naturellement bruitées. Aussi l'évolution de l'état obtenue par la seule *observation* de la position de l'objet est elle-même bruitée.

Dans certaines applications [16], ce bruit n'est pas génant. Il s'agit souvent d'applications pour lesquelles l'*estimation* d'un état est limitée à la position d'un motif dans l'image sans la prise en compte de ses caractéristiques dynamiques. Pour des applications nécessitant ces

dernières, il est indispensable de mettre en oeuvre une étape de filtrage. Il doit permettre de connaître les états estimés ainsi que les incertitudes sur cette estimation. Dans une formulation bayésienne [6], il s'agit de déterminer une densité de probabilité qui décrit la probabilité du vecteur d'état $\mathrm{X_k}$ à l'instant k connaissant les observations (les mesures) $\mathrm{Y_j}$ à des instants différents (j variant de 1 à k). Nous pouvons citer deux techniques employées dans les systèmes de suivi : le *filtre de Kalman* et le *filtre à particules*.

Mais avant de présenter ces deux méthodes dans les paragraphes suivants et plus particulièrement le *filtre de Kalman* qui est la méthode la plus utilisée, nous rappelons ici les notions de vecteur d'état et de matrice de covariance associés à un système.

1.4.1 Vecteur d'état $\mathrm{X_i}$

Un système physique peut toujours être caractérisé, à un instant donné, par un certain nombre de paramètres. Plus le nombre de paramètres est grand, plus la connaissance que l'on a du système est complexe. A l'opposé, si on diminue le nombre de paramètres, on s'éloigne peu à peu de la réalité du phénomène. Déterminer le nombre optimum de paramètres à prendre en compte dans un problème, revient à faire un *compromis entre précision des résultats et complexité des calculs*. L'ensemble des paramètres retenus pour décrire un système à un instant donné constitue les composantes d'un vecteur $\mathrm{X_i}$ de dimension N appelé *vecteur d'état*.

1.4.2 Matrice de covariance $\mathrm{P_i}$

Lorsqu'un système est imparfaitement connu, la dimension du *vecteur d'état* est inférieure au nombre de paramètres pour représenter entièrement ce système. Le vecteur $\mathrm{X_i}$ est alors une *estimation* de l'état du système, et il convient de mesurer la qualité de cette *estimation*. La *matrice de covariance* de cet état, notée $\mathrm{P_i}$, est une matrice de dimensions $N \times N$, qui mathématiquement est égale à la moyenne de $(\mathrm{X_i} - \overline{\mathrm{X}}_\mathrm{i})(\mathrm{X_i} - \overline{\mathrm{X}}_\mathrm{i})^t$ pour toutes les réalisations possibles de $\mathrm{X_i}$ à la date i ($\overline{\mathrm{X}}_\mathrm{i}$ étant la moyenne des $\mathrm{X_i}$). La covariance est la moyenne des écarts à la moyenne. La diagonale de cette matrice représente notamment les variances des composantes de $\mathrm{X_i}$. Elle permettra donc par la suite d'évaluer la *qualité des restitutions de chacune des composantes du vecteur d'état*.

Après ces quelques rappels, nous pouvons maintenant décrire les différentes phases d'un *filtre de Kalman* appliquée à une série de mesures. Ces phases successives ont été présentées

lors de la décomposition d'un algorithme de suivi.

1.4.3 Le filtre de Kalman

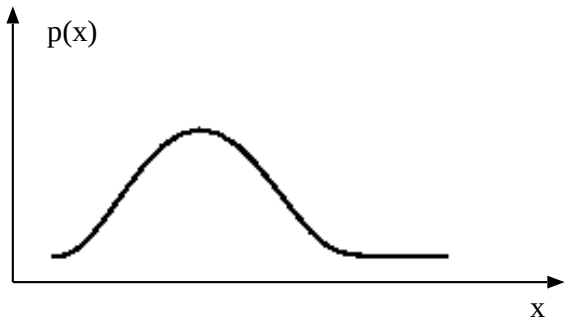


FIGURE 1.3 – représentation de la densité de probabilité (*filtre de Kalman*).

La technique la plus utilisée est le *filtrage de Kalman* [70]. Il considère la fonction de probabilité comme une Gaussienne (figure 1.3). L'*estimation* de l'état sera alors souvent donnée via un vecteur moyen et sa matrice de covariance associée.

Le *filtre de Kalman* permet donc d'estimer et de prédire, à chaque instant de la séquence, l'état de la cible à partir de mesures bruitées. Le calcul peut être décomposé en deux phases :

1. *une phase de prédiction* : à partir de l'estimation du vecteur d'état à l'instant k, on va prédire le vecteur d'état à l'instant $k+1$. Cette prédiction est réalisée par une modélisation du mouvement de l'objet.
2. *une phase de recalage* : les valeurs mesurées sont utilisées afin de corriger l'état prédit.

Phase de prédiction

Cette première phase dans l'algorithme de filtrage comprend quatre étapes :

1. en supposant que l'on connaisse l'état $\mathrm{X_k}$ à l'instant k, l'évolution de l'état de la cible c'est à dire la *prédiction de l'état de la cible a priori* peut être décrite par l'équation récurrente suivante :

$$\widehat{\mathrm{X}}_{\mathrm{k+1/k}} = \mathrm{A_k}\widehat{\mathrm{X}}_{\mathrm{k/k}} + \mathrm{V_k} \tag{1.11}$$

   – $\widehat{\mathrm{X}}_{\mathrm{k+1/k}}$ correspond à l'estimation de la *prédiction à l'instant* k du vecteur d'état pour *l'instant suivant* $k+1$.
   – $\widehat{\mathrm{X}}_{\mathrm{k/k}}$ est le vecteur d'état de dimension N correspondant à *l'estimée optimale de l'instant* k.

- A_k est la *matrice d'évolution* de la cible dans la séquence. C'est une matrice $N \times N$ qui dépend de la date k et se nomme également *matrice de transition*.
- le produit $A_k \widehat{X}_{k/k}$ doit être le reflet des équations modélisant le mouvement de la cible dans la séquence d'images.
- V_k est un *bruit d'état* modélisant l'imperfection du modèle de mouvement. Dans la pratique, ce bruit est supposé de moyenne nulle et n'intervient dans les équations que par l'intermédiaire de sa matrice de covariance Q_k.

Cette équation d'évolution permet ainsi de prédire l'état X_{k+1}, noté $\widehat{X}_{k+1/k}$ à partir de l'état connu X_k. Cette prédiction est plus ou moins précise. Pour que ce résultat soit utilisable, on doit mesurer sa précision. Habituellement, on mesure la qualité d'une prédiction par une matrice de covariance dans le cas d'un vecteur X de dimension N, c'est à dire $P_{k+1/k}$.

2. l'*estimation de la matrice de covariance* $P_{k+1/k}$ de l'état $\widehat{X}_{k+1/k}$ se traduit par l'équation :

$$P_{k+1/k} = A_k P_{k/k} A_k^t + Q_k \tag{1.12}$$

- $P_{k/k}$ est la *matrice de covariance* de l'erreur de prédiction de $\widehat{X}_{k/k}$. C'est une matrice de dimensions $N \times N$.
- Q_k est la *matrice de covariance* de V_k. C'est aussi une matrice de dimensions $N \times N$. Elle représente l'erreur de modélisation et se compose de valeurs faibles si le modèle de mouvement est précis.
- $P_{k+1/k}$ caractérise ainsi la *variance de l'erreur de prédiction* de $\widehat{X}_{k+1/k}$ et cette variance dépend de la précision de l'estimation précédente $P_{k/k}$ et de l'amplitude du bruit.

3. la *prédiction de l'observation* (de la mesure brute) $\widehat{Y}_{k+1/k}$ s'écrit sous la forme matricielle :

$$\widehat{Y}_{k+1/k} = H_k \widehat{X}_{k+1/k} + W_k \tag{1.13}$$

- H_k est la *matrice d'observation* de dimensions $N \times N$. En notant H_k=H, on suppose maintenant que cette matrice reste constante. Mais en pratique, une mise à jour de H_k est possible pour chaque nouvelle image de la séquence.
- W_k est un *bruit* modélisant l'erreur faite sur la mesure brute. Dans la pratique, ce bruit est supposé de moyenne nulle et n'intervient dans les équations que par l'intermédiaire de sa matrice de covariance R_k.

On dispose alors de la meilleure prédiction de l'état X_{k+1} ne connaissant que les informations de l'état X_k sans tenir compte des informations a posteriori. Les équations d'évolution permettent donc de prédire l'état futur d'une cible dans une séquence d'images. Cependant, en l'absence de mesures, les prédictions deviennent rapidement imprécises. La matrice de covariance $P_{k/k}$ devient très grande et la précision de la prédiction se dégrade d'autant. A ce stade, interviennent les mesures réalisées à la date $k+1$.

4. la dernière étape de la phase de prédiction correspond donc à la réalisation des *mesures brutes* $Y_{k+1/k+1}$ issues du module de détection de cible.

Phase de recalage

Comme pour la phase de prédiction présentée auparavant, cette deuxième phase de calcul se décompose en plusieurs étapes :

1. les observations externes $Y_{k+1/k+1}$ permettent de conforter ou d'invalider les prédictions du modèle. On parle alors de *recalage de l'état ou d'innovation entre mesures brutes et estimation* qui s'exprime par la relation :

$$S_{k+1} = Y_{k+1/k+1} - \widehat{Y}_{k+1/k} \tag{1.14}$$

Lorsqu'on effectue la mesure $Y_{k+1/k+1}$, la différence entre celle-ci est la valeur prédite $\widehat{Y}_{k+1/k}$ fournit une indication sur l'erreur d'estimation dont on tient compte pour améliorer l'estimation. De plus, la variance des bruits de mesure (matrice R_{k+1}) et la variance de la prédiction a priori de l'état de la cible sont connues, ainsi :

- si le bruit de mesure est nul, la meilleure estimée (ou l'estimée optimale) est fournie par la mesure.
- si la variance $P_{k+1/k}$ de la prédiction $\widehat{X}_{k+1/k}$ est nulle, il n'y a pas d'erreur de prédiction, la meilleure estimée $\widehat{X}_{k+1/k+1}$ est fournie par la prédiction, sans tenir compte de l'observation $Y_{k+1/k+1}$. C'est le cas d'un bruit V_k nul et de conditions initiales nulles.
- si les variances du bruit et de la prédiction (R_{k+1} et $P_{k+1/k}$) sont différentes de zéro, on effectuera une correction proportionnelle à l'écart entre la valeur mesurée et la valeur prédite.

2. connaissant $\widehat{X}_{k+1/k}$, $P_{k+1/k}$ et les mesures $Y_{k+1/k+1}$ associées à leur matrice de covariance R_{k+1}, on peut estimer de manière optimale l'état $\widehat{X}_{k+1/k+1}$:

– tout d'abord, on calcule le *gain de Kalman* $\mathrm{K}_{k+1/k+1}$ obtenu à partir de l'équation :

$$\mathrm{K}_{k+1/k+1} = \mathrm{P}_{k+1/k}\mathrm{H}^{t}(\mathrm{H}\mathrm{P}_{k+1/k}\mathrm{H}^{t} + \mathrm{R}_{k+1})^{-1} \tag{1.15}$$

où R_{k+1} est la *matrice de covariance* associée aux précisions des mesures externes. Si les mesures sont beaucoup moins précises que la prédiction, le gain de Kalman est faible. Le filtre tiendra alors peu compte des mesures.

– l'*estimée optimale de l'état* X_{k+1} est finalement calculée à l'aide de l'expression :

$$\widehat{\mathrm{X}}_{k+1/k+1} = \widehat{\mathrm{X}}_{k+1/k} + \mathrm{K}_{k+1/k+1}\mathrm{S}_{k+1} \tag{1.16}$$

L'estimée optimale est linéaire, sans biais et minimise l'erreur d'estimation au sens de la minimisation de la matrice de covariance de cette erreur.

3. le calcul de la *matrice de covariance de l'erreur d'estimation au sens de la minimisation* décrit la qualité de l'estimée optimale :

$$\mathrm{P}_{k+1/k+1} = (\mathrm{I} - \mathrm{K}_{k+1/k+1}\mathrm{H})\mathrm{P}_{k+1/k} \tag{1.17}$$

La solution $(\widehat{\mathrm{X}}_{k+1/k+1}, \mathrm{P}_{k+1/k+1})$ constitue la *réponse du filtre de Kalman*. Elle est statistiquement la meilleure réponse possible, connaissant l'état précédent de la cible et les mesures observées à la date $k+1$.

1.4.4 Le filtre à particules

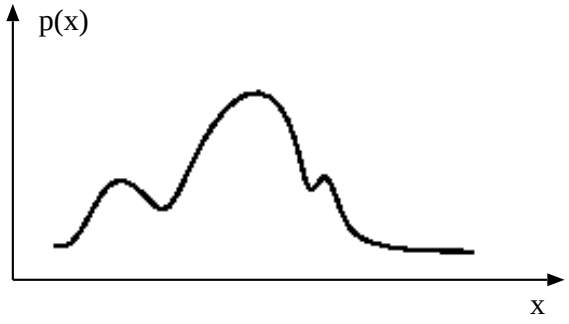


FIGURE 1.4 – représentation de la densité de probabilité (*filtre à particules*).

Pour des problèmes de suivi non-linéaires et non-Gaussiens, il existe des techniques basées sur des algorithmes Bayésiens, nommés *filtres à particules* [6] [63]. Ils reposent sur une caractérisation de la fonction de probabilité selon la méthode de Monté-Carlo, c'est à dire de la

représenter par un très grand nombre d'échantillons tirés au hasard et pondérés. Les estimées sont alors calculées à partir de ces échantillons et de leurs poids associés. Ces filtres permettent d'avoir des densités de probabilité multi-modales et non-Gaussiennes (figure 1.4).

Ces différentes techniques permettent donc un filtrage de l'état et de ses dynamiques. Et aussi de déterminer (une ou) des zones de mesure et d'observation dont (la ou) les positions et les dimensions sont déterminées selon l'état prédit ainsi que l'incertitude associée. Ceci nous amène à évoquer les *modèles de mesure*, liés à la représentation de l'objet dans les méthodes de suivi 2D et 3D.

1.5 Présentation des méthodes de suivi d'objets

Le suivi d'objets dans une séquence vidéo est un thème très apprécié dans le domaine de la vision artificielle. D'une manière générale, il existe deux approches de suivi fondamentalement différentes : le suivi basé sur le mouvement et le suivi basé sur la mise en correspondance d'un modèle qui peut être géométrique ou photométrique. La première approche est plus particulièrement dédiée au suivi 2D d'objets alors que la deuxième peut être aussi bien appliquée au suivi 2D ou 3D d'objets.

1.5.1 Suivi basé sur le mouvement

Les systèmes de suivi basés sur le mouvement [87] [40] [52] comptent entièrement sur la détection de mouvement pour détecter le mouvement d'un objet. Ils ont l'avantage de pouvoir suivre n'importe quel objet en mouvement sans tenir compte de la taille ou de la forme. Ces techniques basées sur le mouvement comprennent alors les méthodes dites *"Optic Flow"* (intensité lumineuse dans l'image représentée par une fonction continue) et *"Motion Energy Tracking"* (segmentation de l'image en régions de mouvement et d'inactivité).

"Optic Flow Tracking"

Le champ de la vélocité rétinienne est connu sous le terme *"optic flow"* (flot optique) [80] [96]. La difficulté avec le *"optic flow tracking"* (suivi du flot optique) est l'extraction du champ de vélocité. En supposant que l'intensité de l'image peut être représentée par une fonction continue $f(x, y, t)$, nous pouvons utiliser un développement en série de Taylor pour montrer que :

$$\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t} = 0 \tag{1.18}$$

où $u = \frac{dx}{dt}$ et $v = \frac{dy}{dt}$ sont les vélocités 2D instantanées au point de coordonnées (x, y). C'est une équation pratique puisque les dérivées partielles $\frac{\partial f}{\partial t}$, $\frac{\partial f}{\partial x}$ et $\frac{\partial f}{\partial y}$ peuvent être localement approximées. La difficulté d'utiliser l'équation 1.18 est que nous avons deux inconnues (u et v) et seulement une équation. Pour résoudre cette équation, il faut imposer des contraintes supplémentaires [49] [99].

Déterminer complètement un champ de flot optique est une opération coûteuse en temps de calcul. C'est pourquoi, résoudre le problème avec quelques points discrets a été une alternative pratique pour les systèmes. Cette méthode consiste à identifier des points d'intérêt (aussi connus comme caractéristiques) dans une série d'images et de suivre leur mouvement [11]. L'inconvénient avec cette technique est que les points d'intérêt dans chaque scène doivent être mis en correspondance avec ceux de l'image précédente. Ceci est généralement un problème très délicat où les difficultés augmentent dans le cas d'une caméra active (apparition et disparition de certains points dans le champ de vision dont on va chercher une mise en correspondance). La complexité de ce problème fait donc que cette méthode est inappropriée pour les applications temps réel.

Les techniques traditionnelles dites *"Optic Flow"* traitent donc une région de l'image comme un *"moving stuff"* [2] et ne peuvent ainsi faire la différence entre les changements de point de vue ou de configuration de l'objet et les changements relatifs en position de la caméra. Ces techniques ont simplement une notion de l'objet à suivre en utilisant le modèle d'une forme quelconque. Si la vue de l'objet à suivre change significativement alors sa forme est différente et le suivi peut être un échec.

"Motion Energy Tracking"

Une autre méthode de suivi de mouvement est la *"motion energy detection"* (détection de l'énergie du mouvement). En calculant la dérivée temporelle de l'image et en utilisant convenablement un filtre pour éliminer le bruit, nous pouvons segmenter une image en des régions de mouvement et d'inactivité. Bien que la dérivée temporelle peut être estimée par une méthode plus exacte, en pratique, elle est calculée par une simple différence d'images :

$$\frac{df(x,y,t)}{dt} \approx \frac{f(x,y,t) - f(x,y,t-\delta t)}{\delta t} \tag{1.19}$$

Cette méthode de détection de mouvement est sensible au bruit et donne des valeurs imprécises. En général, pour améliorer le résultat de la différence d'images, ces techniques utilisent en plus une information spatiale de contour pour permettre l'extraction du mouvement des contours. P. Allen, B. Yoshimi et A. Timcenko [4] ont utilisé les passages à zéro de la dérivée seconde du filtre Gaussien comme un détecteur de contours et ont combiné cette information de localisation de contours avec les dérivées spatiales et temporelles locales pour estimer le flot optique.

En pratique, l'implémentation de la détection de mouvement, basée sur la combinaison d'une différence d'images avec une information spatiale, est une méthode très largement utilisée. En plus de la simplicité de calcul, la *"motion energy detection"* est adaptée aux architectures dites *"pipeline"* permettant une implémentation facile sur des machines dédiées à la vision. Un inconvénient majeur de cette méthode est que le mouvement du pixel est détecté mais pas quantifié. De plus, elle n'est pas adaptée pour une application sur des systèmes à caméra active. En effet, comme l'utilisation d'une caméra active peut induire un mouvement apparent de la scène observée, il faut d'abord compenser ce type de mouvement avant d'appliquer la méthode dite *"motion energy detection"*.

1.5.2 Notion de mise en correspondance d'un modèle

Dans le cas du suivi 2D ou 3D d'objets, la mise en correspondance suppose connu le modèle de l'objet observé et permet de superposer ce modèle avec les données fournies par le ou les capteurs.

D'une manière générale, on dispose de deux formes : une forme-modèle M connue et une forme-données D résultant des informations fournies par des capteurs 2D ou 3D. Ces deux formes étant décrites dans des systèmes de coordonnées différents (système de coordonnées du capteur et système de coordonnées de modélisation), il est nécessaire, afin de pouvoir les exploiter, de les exprimer dans un système de coordonnées commun. La phase de mise en correspondance consiste donc, après choix d'un mode de représentation, à trouver la transformation rigide ou non-rigide à appliquer sur D afin de minimiser un critère de distance entre D et

M. Les méthodes de mise en correspondance peuvent être classées suivant les dimensions des données et du modèle :

1. données 2D / modèle 2D : simples à manipuler,
2. données 2D / modèle 3D : très utilisé grâce à la combinaison du pouvoir descriptif des modèles 3D avec la stratégie rapide et peu coûteuse des capteurs 2D,
3. données 3D / modèle 2D : perte d'informations au niveau des données,
4. données 3D / modèle 3D : allie la puissance de description des modèles 3D, avec celle des informations 3D.

La complexité et l'exactitude de la méthode de mise en correspondance dépendent principalement du choix des primitives utilisées. Toutefois, le choix d'une méthode de mise en correspondance peut dépendre des différences observées entre les images :

1. différences dues à l'acquisition, qui peuvent être corrigées : ces distorsions (changement de point de vue ou bruit du capteur) peuvent être modélisées, ce qui permet de déterminer le type de transformation à rechercher.
2. différences dues à l'acquisition, mais qui ne peuvent pas être corrigées : ces distorsions sont difficiles à modéliser, car dépendantes de la scène (éclairage, conditions atmosphériques, ombrage, effet de perspective, certains bruits du capteur, ...) et induisent par conséquent, des erreurs dans la mise en correspondance.

Afin de pouvoir détecter un objet dans l'image, il faut connaître une information a priori sur cet objet (son modèle). Pour cette connaissance, deux approches sont distinguées dans la littérature :

1. les approches par primitives consistent à déterminer explicitement des invariants dans la classe objet, et leur relation entre eux. La détection de l'objet consistera alors en deux phases :
   (a) extraction des primitives bas niveaux ou des invariants dans l'image (niveaux de gris, contours ou segments, ...),
   (b) analyse selon un modèle a priori de l'objet.

   A travers cette algorithmie, se dessine un schéma souvent rencontré en vision : analyse bas niveau, et recherche des correspondances avec un modèle. Ce type de système exploite différentes propriétés de l'objet (sa géométrie par exemple) à l'aide de mesures de distances, d'angles ou/et de tailles sur les primitives extraites. Il s'agit souvent de systèmes

pouvant être exploitables en temps réel. Le plus grand défaut des méthodes par primitives est qu'elles sont très sensibles à l'environnement de l'objet. Elles nécessitent souvent, pour être efficace, un arrière plan à l'opposé des primitives utilisées. Par exemple, pour une méthode utilisant les contours, un environnement contenant peu de contours marqués est préférable. Ainsi la robustesse d'une approche par primitives est souvent lié à l'environnement dans lequel la détection doit avoir lieu. Les hypothèses à vérifier en limitent le domaine d'application.

2. les techniques basées sur l'apparence répondent dans une certaine mesure à ce problème. L'apparence peut se définir comme la forme de la surface d'intensité de l'image. Par exemple, il peut s'agir basiquement du tableau 2D des pixels. La détection de l'objet est alors traitée comme un problème de reconnaissance d'images. Ces méthodes permettent de contourner les erreurs potentielles dues à une modélisation incomplète ou inappropriée. Mais elles sont souvent très coûteuses en temps de calcul car elles requièrent une exploration exhaustive, multi-échelles, multi-résolutions des images. Notons cependant que l'apparence d'un objet dans une image est très variable. Elle va dépendre de la position et de l'orientation de l'objet par rapport à la caméra mais également de l'éclairage de la scène et de la nature même de l'objet. Cela va jouer sur la taille, la forme et la texture de l'objet à suivre (notamment si certains détails seront visibles ou pas dans l'image).

1.5.3 Suivi basé sur la modélisation d'objets

Nous avons vu que dans les approches traditionnelles de suivi basées sur la modélisation d'objets, deux approches principales sont généralement distinguées.

Les ***approches basées sur la mise en correspondance de primitives visuelles*** utilisent des caractéristiques locales comme des points, des segments de droite, des arêtes ou des régions. Avec ces techniques, il est possible de localiser l'objet [73] dans l'image courante et de prédire les positions des caractéristiques dans les images suivantes, selon un modèle de mouvement [112] [114] et un modèle d'incertitude [81]. L'avantage de cette méthode est de pouvoir travailler en trois dimensions : les translations et rotations de l'objet peuvent donc être estimées. Dans cette approche, il s'agit donc de mettre en correspondance un modèle de référence M et une projection P de cet objet dans l'image courante. Ceci est réalisé par le biais de l'estimation des paramètres caractérisant la transformation T du modèle d'objet M en la projection P de cet objet dans l'image. Lorsque l'on s'intéresse uniquement au suivi de l'objet

dans l'image, une alternative consiste à utiliser des modèles approximatifs et à n'exploiter que des modèles d'objet et de mouvement 2D. Il est alors nécessaire d'introduire une composante déformable pour pouvoir s'adapter aux déformations non linéaires des projections de l'objet dans le plan image liées à des effets de perspective non prises en compte par ces modèles 2D.

Koller et *al.* [72] obtiennent de bons résultats de suivi en utilisant un modèle paramétrique 3D d'un véhicule. Ce type de modèle peut être ajusté pour différentes gammes de véhicules. Les mouvements compliqués, tels que les demi-tours ou les manoeuvres pour se garer, ont été correctement suivis en incluant une modélisation de l'ombre dans le modèle 3D du véhicule pour l'améliorer de manière satisfaisante. A la différence des *méthodes de suivi basées sur le mouvement*, seul un objet modélisé peut être suivi. Nous citons également les travaux de Strom *et al.* [104] et Basu *et al.* [7]. Ils décrivent un système temps réel de suivi et une modélisation 3D. L'idée directrice est de sélectionner un ensemble dense de points caractéristiques. Ils sont ensuite mis en correspondance d'images en images pour mettre à jour la pose du modèle 3D. Pour cela, un modèle générique 3D (approximation polygonale) de l'objet est nécessaire. D'une manière générale, les techniques de recherche de pose sont naturellement moins sensibles aux occultations. En effet, elles sont basées sur des correspondances locales. Si plusieurs correspondances sont manquantes, la pose peut encore être calculée.

Ces techniques de suivi, basées sur des primitives [92] [20] extraites de chaque image de la séquence, différent par les outils de mise en correspondance utilisés (dans le cas de suivi de points par exemple, choix du plus proche voisin validé par des mesures de corrélation ou par cohérence temporelle des trajectoires). Des filtres de Kalman sont généralement employés pour réaliser le suivi et la prédiction des primitives dans l'image suivante. La qualité des résultats de ces techniques dépend fortement du contenu de la scène et est très sensible à la densité des primitives dans l'image. Par conséquent, la robustesse du suivi est étroitement liée à la méthode utilisée pour le calcul des caractéristiques dans l'image.

Notons également qu'il est intéressant de pouvoir suivre les contours d'un objet non rigide et d'analyser leurs mouvements. C'est ainsi que Terzopoulos *et al.* [71, 83] ont introduit la notion de *contours actifs* ou *snake model*, c'est à dire des contours qui peuvent se déformer sous l'action de certaines contraintes internes. Par la suite, Curwen et Blake [33] ont proposé une représentation *B-spline* des contours actifs et Dubuisson *et al.* [41] ont utilisé une représentation polygonale dans les problèmes de suivi de véhicules. Ces différentes techniques sont basées sur

la mesure spatiale des niveaux de gris ou sur la position des caractéristiques dans une image.

D'autre part, les ***approches globales ou basées sur un modèle (ou un motif)*** utilisent le modèle dans sa totalité. Il ne s'agit plus, dans ce cas, d'utiliser de primitives de haut niveau (points d'intérêt) mais plutôt l'apparence de l'objet. Il s'agit donc de déterminer la correspondance entre le modèle de l'apparence choisi et l'estimation de la pose. Ces techniques utilisent, par exemple, une classification par hyperplan ("Support Vector Machine") [93], une représentation des variations de l'apparence de l'objet par une distribution possibiliste [94] [95], des histogrammes de champs réceptifs multidirectionnels [36] ou des transformations de caractéristiques optimales [62]. Une méthode hybride développée par T.F. Cootes, G.J. Edwards et C.J. Taylor [25] utilise un modèle statistique de forme et d'apparence en niveaux de gris de l'objet.

Le point fort de ces méthodes est leur capacité à traiter des motifs ou des modèles complexes qui ne peuvent être modélisés par des caractéristiques locales. Elles sont très robustes et ont été énormément utilisées. Elles sont aussi appelées *sum-of-square-difference (SSD)* puisque elles consistent à minimiser la somme des carrés des différences entre un modèle de référence et une région de l'image. Une norme L_2 est généralement utilisée pour mesurer cette erreur. Historiquement, une recherche exhaustive était utilisée. Mais cette stratégie n'est pas applicable dans le cas de transformations plus complexes que des translations 2D, qui nécessitent des espaces de paramètres de dimensions supérieures. Des méthodes plus récentes posent le problème comme un problème de minimisation non linéaire, utilisant des algorithmes du type Newton ou Levenberg-Marquard. Darell *et al.* [35], Brunelli *et al.* [17] proposent de maximiser un critère de corrélation entre un vecteur caractérisant le modèle de référence et le contenu de l'image. Les temps de calcul, significatifs dans ce cas, peuvent être réduits en travaillant dans des sous espaces de la représentation initiale de l'image [109, 85, 86].

La limitation principale de ces approches est leur manque de résistance au regard des occultations. Black et Jepson [13] ont surmonté cette limitation en reconstruisant les parties occultées. Ils remplacent la norme quadratique généralement utilisée pour construire l'approximation de l'image dans *l'espace propre* par une norme d'erreur robuste. Cette reconstruction revient à une minimisation d'une fonction non linéaire, optimisée en utilisant une méthode de descente de gradient simple. Ils utilisent la même stratégie pour trouver la transformation paramétrique alignant le motif sur l'image. Mais cette mise en correspondance est une opération coûteuse en

temps d'exécution. Elle nécessite pas moins de 30 secondes de traitement par image sur une SGI Indy Workstation. Par conséquent, les performances du système de suivi sont limitées par l'efficacité de la méthode retenue et les types d'objets modélisés.

Des travaux similaires reposant sur l'utilisation *d'espaces propres* ont été réalisés par K. Deguchi *et al.* [106, 38], Shree K. Nayar *et al.* [89, 86] pour le suivi d'objets et du positionnement d'un robot par vision. Ils proposent un algorithme général d'apprentissage basé sur une analyse en composantes principales ("Eigenspaces") pour déterminer la correspondance entre la position du robot et l'apparence de l'objet. Pour cela, l'objet 3D est représenté par une collection d'images 2D prises pour différentes positions du robot. Néanmoins, cette technique présente un inconvénient majeur en reconnaissance : elle nécessite une normalisation précise des images (objet intégralement visible, correctement localisé dans l'image et séparé du fond). Ces algorithmes restent sensibles aux variations d'éclairement, aux ombres, aux saturations de caméra et aux problèmes des occultations.

1.6 Introduction aux travaux présentés dans ce mémoire

Plus récemment, de nouvelles méthodes efficaces de suivi ont été proposées : le problème du suivi est formulé comme un problème de recherche du meilleur ensemble de paramètres (au sens des *moindres carrés*) décrivant le mouvement et la déformation de la cible au cours de la séquence. Dans ce cas, les variations des paramètres sont écrites comme une fonction linéaire d'une image de différence (la différence entre l'image de référence et l'image courante). Cette approche est très efficace car le mouvement peut être facilement déduit de l'image de différence. Cootes, Edwards et Taylor [25] l'utilisent pour estimer dynamiquement les paramètres d'un modèle de visage en se basant sur l'apparence (modèle 2D). Seuls, quelques travaux utilisent cette approche avec des transformations projectives [55, 76], car ces dernières sont non linéaires et la taille de l'espace des paramètres est trop importante.

Hager et Belhumeur [60] ont récemment proposé une méthode efficace pour ce type de problème. La position du modèle de la cible dans la première image est supposée être connue. Le problème est alors d'estimer la position de ce modèle dans les images suivantes. La position actuelle du modèle dans l'image courante peut être calculée en comparant les valeurs de niveaux de gris du modèle de la cible avec les valeurs de niveaux de gris de la région prédite (figure 1.5). Ce calcul est possible car au cours d'une phase d'apprentissage hors ligne, une relation

entre les variations d'intensité lumineuse et les variations de position a été apprise. Hager et Belhumeur [60] proposent d'estimer cette relation en utilisant l'inverse d'une image Jacobienne. Nous allons montrer, dans le chapitre suivant, que cette relation peut être obtenue en utilisant une approche différente (une approximation par hyperplans) menant à de meilleurs résultats, sans calcul additionnel.

Le suivi 3D d'objets que nous allons présenter dans ce mémoire est une méthode basée sur l'apparence où l'objet 3D est modélisé par une collection d'images 2D de référence.

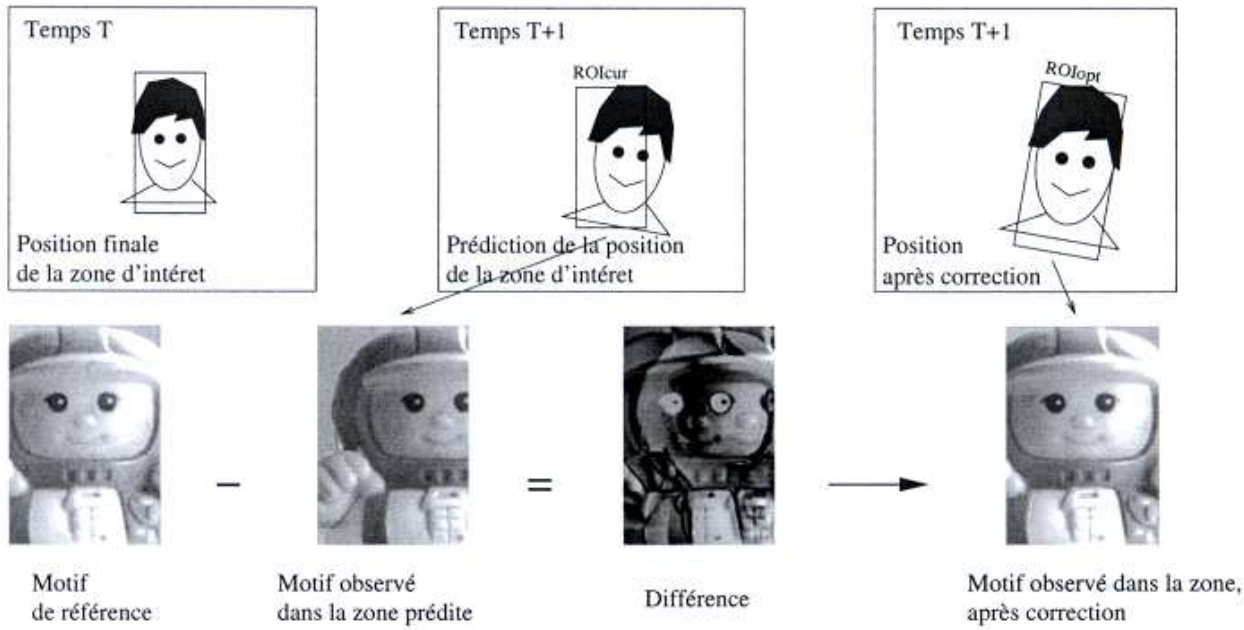


FIGURE 1.5 – illustration du principe de suivi 2D.

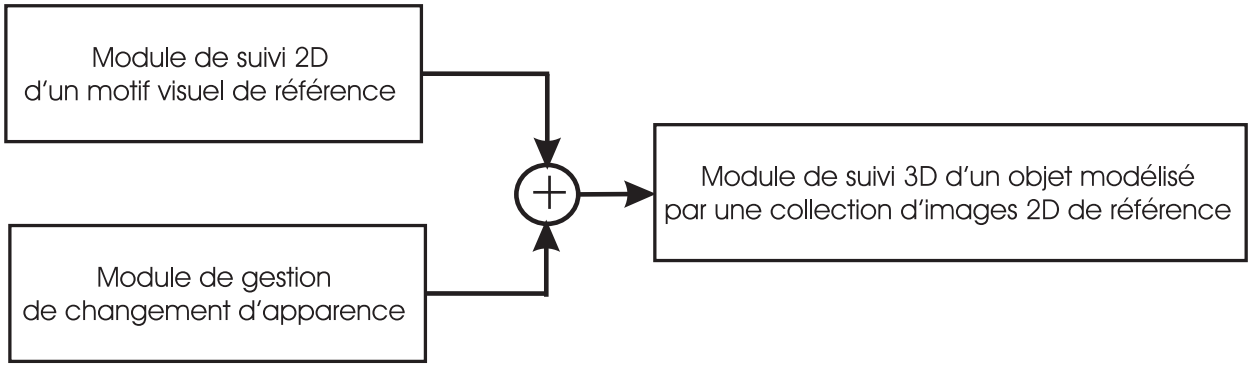


FIGURE 1.6 – illustration du principe de suivi 3D.

C'est un algorithme temps réel composé de deux modules (figure 1.6) :

– le premier permettant le suivi 2D d'un motif visuel de référence en utilisant l'approximation par hyperplans,

– le second assurant le passage d'un motif de référence à l'autre pour gérer les changements d'apparence de l'objet 3D dans la séquence d'images.

Chapitre 2

Suivi efficace d'un motif visuel 2D : approche par hyperplans

Dans ce chapitre, nous allons présenter une méthode permettant de suivre efficacement et rapidement le déplacement d'un motif visuel 2D donné dans un flot vidéo. Cette approche, basée sur l'apparence, qui est à l'origine de mes travaux, a fait l'objet de plusieurs publications [66] [67] [68].
Elle consiste à écrire les variations des paramètres décrivant le mouvement et la déformation de la cible comme une fonction linéaire d'une image de différence entre le motif de référence à suivre et le motif échantillonné dans l'image courante. Nous proposons d'estimer cette relation, lors d'une phase d'apprentissage hors ligne, en utilisant une *approximation par hyperplans*. Ces travaux différent de ceux présentés par Hager et Belhumeur [60] qui utilisent l'inverse d'une image Jacobienne pour estimer cette relation linéaire.

Dans les sections suivantes, nous allons présenter, de manière succincte, la méthode de suivi proposée par Hager et Belhumeur, puis décrire l'approche que nous proposons avant de les comparer. Des expérimentations sur des images réelles sont effectuées dans le but de montrer la supériorité de notre approche de suivi de motif visuel 2D sans occultation. Finalement, le problème des occultations sera traité par une méthode de seuillage adaptatif.

2.1 Suivi efficace d'une région dans l'image

Notons $I(\mathbf{x}, t)$ la valeur de l'intensité lumineuse au point de coordonnées $\mathbf{x} = (x, y)$ dans une image acquise au temps t. Posons $\mathcal{R} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ l'ensemble des coordonnées des N

points de l'image qui définissent une *région cible*. $\mathbf{I}(\mathcal{R},t) = (I(\mathbf{x}_1,t), I(\mathbf{x}_2,t), \cdots, I(\mathbf{x}_N,t))$ est un vecteur contenant les valeurs de l'intensité lumineuse de la région cible au temps t. Nous appelons $\mathbf{I}(\mathcal{R},t_0)$ le *modèle de référence*. C'est le modèle (ou le motif) qui doit être suivi ; t_0 est le temps initial ($t = 0$).

Le mouvement relatif entre l'objet et la caméra entraîne des changements de position du modèle dans l'image. Nous supposons que ces transformations peuvent être parfaitement modélisées par un *modèle paramétrique de mouvement* $\mathbf{f}(\mathbf{x};\mu(t))$ où $\mathbf{x}$ indique les coordonnées d'un point dans l'image et $\mu(t) = (\mu_1(t), \mu_2(t), \ldots, \mu_n(t))$ un ensemble de paramètres. Nous supposons que $N > n$ et que $\mathbf{f}$ est une fonction différenciable à la fois en $\mathbf{x}$ et μ. Nous appelons μ le vecteur des paramètres du mouvement. L'ensemble des coordonnées des N points de l'image $(\mathbf{f}(\mathbf{x}_1;\mu(t)), \mathbf{f}(\mathbf{x}_2;\mu(t)), \ldots, \mathbf{f}(\mathbf{x}_N;\mu(\mathbf{t})))$ est noté $\mathbf{f}(\mathcal{R};\mu(t))$. Au temps t_0, la position du modèle est $\mu(t_0)$, alors notée μ_0^*.

Avec ces hypothèses, "suivre l'objet au temps t" signifie :

$$\text{"calculer } \mu(t) \text{ tel que } \mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)),t) = \mathbf{I}(\mathbf{f}(\mathcal{R};\mu_0^*),t_0)\text{"}.$$

Nous écrivons $\mu(t)$ l'estimation de la valeur réelle $\mu^*(t)$. Le vecteur des paramètres du mouvement de la région cible $\mu(t)$ peut être estimé en minimisant au sens *des moindres carrés* la fonction :

$$O(\mu(t)) = \|\mathbf{I}(\mathbf{f}(\mathcal{R};\mu_0^*),t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)),t)\|$$

Cette formulation très générale du suivi a été utilisée par plusieurs auteurs. Black et Jepson [13] donnent un bon exemple d'utilisation de cette minimisation. Ils ont proposé un algorithme d'optimisation (Levenberg-Marquard) qui est malheureusement lent en exécution et ne tolère seulement que des petits mouvements de l'objet.

Hager et Belhumeur [60] proposent un calcul très simple et efficace de $\mu(t+\tau)$ en écrivant :

$$\mu(t+\tau) = \mu(t) + \mathbf{A}(t+\tau)\left(\mathbf{I}(\mathbf{f}(\mathcal{R};\mu_0^*),t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)),t+\tau)\right) \tag{2.1}$$

où $\mathbf{A}(t+\tau)$ peut être obtenue avec un simple calcul en ligne, et τ exprime le temps entre deux images successives. Cette formulation fait qu'il est possible de l'implémenter en temps réel sur des stations de travail standards.

Si nous notons :

$$\begin{cases} \delta\mathbf{i}(t+\tau) = \mathbf{I}(\mathbf{f}(\mathcal{R};\mu_0^*),t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)),t+\tau) \\ \\ \delta\mu(t+\tau) = \mu(t+\tau) - \mu(t) \end{cases}$$

l'équation (2.1) peut être écrite sous la forme :

$$\delta\mu(t+\tau) = \mathbf{A}(t+\tau)\delta\mathbf{i}(t+\tau) \tag{2.2}$$

2.2 Approximation Jacobienne (AJ) contre Approximation par Hyperplans (AH)

2.2.1 Approximation Jacobienne (AJ) [60]

L'équation (2.2) montre clairement que $\mathbf{A}(t+\tau)$ joue le rôle d'une matrice Jacobienne. Pour cette raison, nous la noterons $\mathbf{A}_j(t+\tau)$. L'estimation de $\mathbf{A}_j(t+\tau)$ peut être obtenue, comme proposée par Hager et Belhumeur [60], en utilisant l'image Jacobienne.

Dans le but de simplifier les notations, nous indiquerons $\mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)),t)$ par $\mathbf{I}(\mu,t)$. Si les valeurs des composantes $\delta\mu$ et τ sont petites, il est possible de linéariser le problème en développant $\mathbf{I}(\mu+\delta\mu,t+\tau)$ en une série de Taylor par rapport à μ et t :

$$\begin{aligned} \mathbf{I}(\mu+\delta\mu,t+\tau) &= \mathbf{I}(\mu,t) \\ &+ \mathbf{I}_\mu(\mu,t)\delta\mu \\ &+ \mathbf{I}_t(\mu,t)\tau + h.o.t. \end{aligned} \tag{2.3}$$

où $h.o.t.$ sont des termes d'ordre supérieur du développement qui peuvent être négligés. $\mathbf{I}_\mu(\mu,t) = \mathbf{M}(\mu,t)$ est la matrice Jacobienne de $\mathbf{I}$ relativement à μ au temps t, et $\mathbf{I}_t$ est la dérivée de $\mathbf{I}$ par rapport à t.

En négligeant les termes d'ordre supérieur $h.o.t.$ et en supposant l'approximation suivante $\mathbf{I}_t(\mu,\tau)\tau = \mathbf{I}(\mu,t+\tau) - \mathbf{I}(\mu,t)$ avec $\delta\mathbf{i} = \mathbf{I}(\mu+\delta\mu,t+\tau) - \mathbf{I}(\mu,t+\tau)$, l'équation précédente (2.3) devient :

$$\delta\mathbf{i} = \mathbf{M}(\mu,t)\delta\mu \tag{2.4}$$

En écrivant :

$$\mathbf{A}_j(t) = (\mathbf{M}^t(\mu,t)\mathbf{M}(\mu,t))^{-1}\mathbf{M}^t(\mu,t) \tag{2.5}$$

nous obtenons directement une expression de $\mathbf{A}_j(t)$ (où $\mathbf{M}^t$ est la matrice transposée de $\mathbf{M}$). En combinant les équations (2.2) et (2.5), nous obtenons :

$$\delta\mu = (\mathbf{M}^t(\mu,t)\mathbf{M}(\mu,t))^{-1}\mathbf{M}^t(\mu,t)\delta\mathbf{i} = \mathbf{A}_j(t)\delta\mathbf{i} \tag{2.6}$$

Le simple calcul de $\mathbf{M}$ nécessite le calcul du gradient de l'image par rapport à la composante du vecteur $\mathbf{f}$. Donc $\mathbf{M}$ doit être entièrement recalculée à chaque nouvelle itération. Ceci est une procédure coûteuse en temps de calcul. Heureusement, il est possible d'exprimer $\mathbf{M}$ comme une fonction du gradient de l'image de référence, permettant d'obtenir $\delta\mu = (\mathbf{M}^t\mathbf{M})^{-1}\mathbf{M}^t\delta\mathbf{i}$ avec seulement quelques calculs en ligne [60].

L'équation (2.6) implique le calcul de la différence d'intensité $\delta\mathbf{i}$. Il est possible de lier $\delta\mathbf{i}$ au *modèle de référence* donné dans la première image. Si nous supposons que le motif est correctement localisé après la correction du vecteur des paramètres du mouvement $\delta\mu$, l'hypothèse de cohérence d'image donne $\mathbf{I}(\mu+\delta\mu, t+\delta\tau) = \mathbf{I}(\mu_0^*, t_0)$, menant à la relation $\delta\mathbf{i} = \mathbf{I}(\mu_0^*, t_0) - \mathbf{I}(\mu, t+\tau)$.

Dans ce cas, l'équation (2.6) relie la différence entre le modèle dans la région courante et le modèle de la cible avec un déplacement $\delta\mu$ alignant la région sur la cible.
Avec ces notations, le suivi consiste à évaluer $\delta\mathbf{i}(t+\tau)$ et en déduire par conséquent $\delta\mu(t+\tau)$, puis finalement, mettre à jour $\mu(t+\tau)$ en accord avec l'équation : $\mu(t+\tau) = \mu(t) + \delta\mu(t+\tau)$.

2.2.2 Approximation par Hyperplans (AH) [67] [68]

Nous proposons une interprétation différente du calcul de la matrice A. L'équation (2.2) $\delta\mu(t+\tau) = \mathbf{A}(t+\tau)\delta i(t+\tau)$ peut être vue comme une modélisation par n hyperplans. Dans ce paragraphe, la matrice $\mathbf{A}$ est écrite $\mathbf{A}_h$ pour la distinguer de $\mathbf{A}_j$. Cependant elle joue le même rôle. Écrivons a_{ij} les élements de la matrice $\mathbf{A}_h$ (la variable temps t étant supprimée pour simplifier les notations).

L'équation (2.2) peut être écrite sous la forme :

$$\begin{cases} (a_{11},\ldots,a_{1j},\ldots,a_{1N},-1)(\delta i_1,\ldots,\delta i_j,\ldots,\delta i_N,\delta\mu_1)^t=0 \\ (a_{i1},\ldots,a_{ij},\ldots,a_{iN},-1)(\delta i_1,\ldots,\delta i_j,\ldots,\delta i_N,\delta\mu_i)^t=0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ (a_{n1},\ldots,a_{nj},\ldots,a_{nN},-1)(\delta i_1,\ldots,\delta i_j,\ldots,\delta i_N,\delta\mu_n)^t=0 \end{cases}$$

Sous cette forme, nous pouvons clairement observer que $a_{i1},\ldots,a_{ij},\ldots,a_{iN}$ sont les coefficients de n hyperplans qui peuvent être estimés en utilisant une estimation au sens des *moindres carrés*.

Durant la phase d'apprentissage, la région d'intérêt est écartée de la position de référence μ_0^* (définie manuellement en sélectionnant la région d'intérêt dans la première image) de $\mu_0{}' = \mu_0^* + \delta\mu_0$. Une fois que la région est déplacée, le vecteur $\delta\mathbf{i} = \mathbf{I}(\mathcal{R},\mu_0^*) - \mathbf{I}(\mathcal{R},\mu_0{}')$ est calculé. Cette procédure de "perturbation" est répétée N_p fois, avec $N_p > N$.

Finalement, nous collectons N_p couples $(\delta\mathbf{i}^k,\delta\mu^k), k\in[1,N_p]$ sachant que $\delta\mathbf{i}^k = (\delta\mathbf{i}_1^k,\ldots,\delta\mathbf{i}_N^k)^t$ et $\delta\mu^k = (\delta\mu_1^k,\ldots,\delta\mu_N^k)^t$. Il est alors possible d'obtenir une matrice A_h telle que :

$$\textstyle\sum_{k=1}^{k=N_p}(\delta\mu^k - \mathbf{A}_h\delta\mathbf{i}^k)^2 \text{ soit minimale.}$$

En écrivant $\mathbf{H} = (\delta\mathbf{i}^1,\ldots,\delta\mathbf{i}^{N_p})$, $\mathbf{Y} = (\delta\mu^1,\ldots,\delta\mu^{N_p})$, et en supposant $N_p > N$ nous obtenons un système surdéterminé ($\mathbf{Y} = \mathbf{A}_h\mathbf{H}$). $\mathbf{A}_h$ peut être calculée par $\mathbf{A}_h = \mathbf{Y}\mathbf{H}^t(\mathbf{H}\mathbf{H}^t)^{-1}$, où $\mathbf{H}^t$ est la transposée de $\mathbf{H}$. Un schéma similaire a déja été proposé par Cootes *et al.* [25]. Ils l'ont utilisé dans le cadre spécifique de l'estimation dynamique des paramètres d'un modèle d'apparence d'un visage.

2.2.3 Interprétation géométrique des deux approximations

Définie dans l'*approximation Jacobienne*, l'équation $\delta\mathbf{i} = \mathbf{M}(\mu,t)\delta\mu$ peut être interprétée comme un ensemble de N hyperplans (un pour chaque niveau de gris). Chaque hyperplan exprime une relation entre un niveau de gris et les n paramètres de la transformation dans un espace à $n+N$ dimensions.

L'équation $A_j(t) = (M^t(\mu,t)M(\mu,t))^{-1}M^t(\mu,t)$ permet alors de définir l'ensemble des n hyperplans donnant les meilleures approximations de chaque paramètre de la transformation à

partir des N hyperplans initiaux.

Ceci peut être illustré en prenant l'exemple simple suivant : supposons que nous avons seulement deux niveaux de gris (ou pixels) notés i_0 et i_1 dans un modèle (ou motif) et un paramètre de transformation μ_0.
L'équation $\delta\mathbf{i} = \mathbf{M}(\mu, t)\delta\mu$ nous donne alors pour $N = 2$ et $n = 1$:

$$\begin{cases} \delta i_0 = m_0 \delta\mu_0 \\ \delta i_1 = m_1 \delta\mu_0 \end{cases} \quad ou \quad \begin{cases} \delta i_0 - m_0 \delta\mu_0 = 0 \quad (P_0) \\ \delta i_1 - m_1 \delta\mu_0 = 0 \quad (P_1) \end{cases} \tag{2.7}$$

Ces deux équations (2.7) peuvent être interprétées comme la définition de deux plans P_0 et P_1 dans un espace $(\delta i_0, \delta i_1, \delta\mu_0)$ à trois dimensions $(N+n=3)$. Ils constituent une approximation du premier ordre des deux fonctions f_0 et f_1 donnant les différences de niveaux de gris δi_0 et δi_1 obtenues pour une variation $\delta\mu_0$ du paramètre de la transformation :

$$\delta i_0 = f_0(\delta\mu_0) \quad et \quad \delta i_1 = f_1(\delta\mu_0) \tag{2.8}$$

Nous écrivons alors les éléments de la matrice M sous la forme :

$$m_0 = \frac{\delta f_0}{\delta\mu_0} \quad et \quad m_1 = \frac{\delta f_1}{\delta\mu_0} \tag{2.9}$$

$(\delta i_0, \delta i_1, \delta\mu_0)$ ou $(f_0(\delta\mu_0), f_1(\delta\mu_0), \delta\mu_0)$ peut être assimilé à une courbe paramétrique C que nous supposons planaire dans l'espace 3D. Ceci est illustré par la figure 2.1.

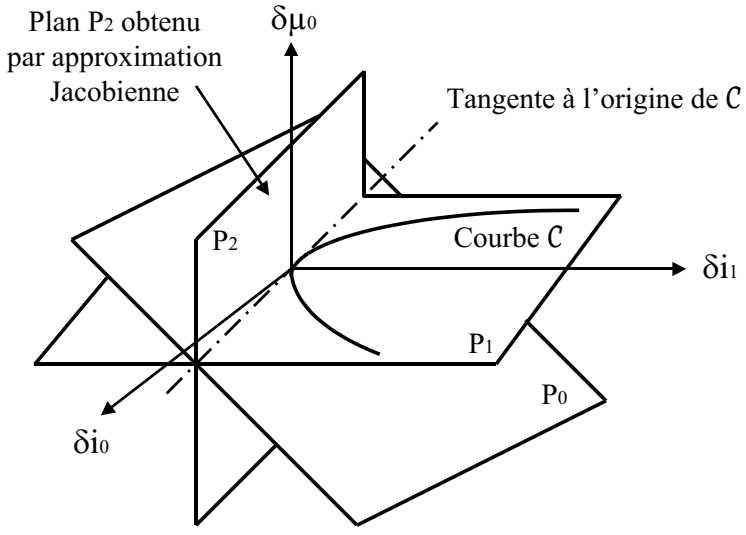


FIGURE 2.1 – interprétation géométrique de l'approximation Jacobienne.

L'équation $A_j(t) = (M^t(\mu,t)M(\mu,t))^{-1}M^t(\mu,t)$ représente un plan P_2 qui est la meilleure approximation des plans P_0 et P_1 au *sens des moindres carrés*. Il n'y aucune raison pour que ce plan P_2 contienne la courbe C.

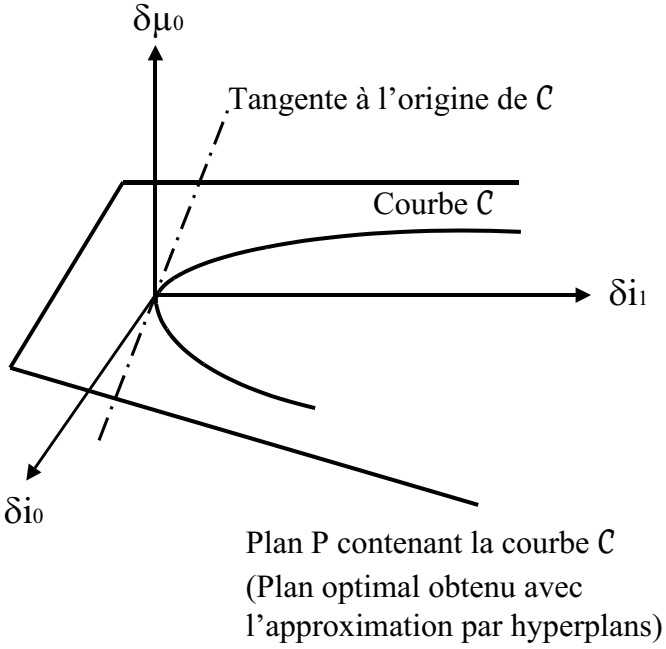


FIGURE 2.2 – interprétation géométrique de l'approximation par hyperplans.

Dans le cas de l'*approximation par hyperplans* illustrée figure 2.2, nous utilisons directement les points de la courbe C pour l'approximer par un plan. Dans le cas de l'hyperplan, le plan obtenu dans notre exemple est le plan optimal P et dans le cas Jacobien, la seule contrainte pour le plan P_2 est d'inclure la tangente à la courbe.

2.3 Avantages de l'Approximation par Hyperplans (AH)

Dans la suite de ce chapitre, les arguments μ ou t seront supprimés quand le contexte nous le permettra.

Bien que la même équation $\delta\mu = \mathbf{A}\delta\mathbf{i}$ soit utilisée dans les deux cas, nous montrerons que les résultats obtenus en utilisant la modélisation par hyperplans sont meilleurs que ceux obtenus en utilisant l'image Jacobienne.

La raison est que dans le cas de l'Approximation par Hyperplans (AH), la meilleure approximation linéaire donnant le mouvement, à partir des différences d'images, est obtenue en utilisant

directement la donnée (des paires de vecteurs (δi et $\delta\mu$) produites par de petites perturbations).

Dans le cas de l'Approximation Jacobienne (AJ), ces données sont d'abord utilisées pour estimer, à partir d'un échantillonnage point par point, une relation linéaire entre des différences d'images et une fonction paramétrique du mouvement. Au final, les relations attendues sont calculées à partir des premières approximations. L'AJ suppose que les niveaux de gris sont des combinaisons linéaires des paramètres du mouvement. Cette hypothèse, qui n'est pas confirmée dans les images réelles, n'est pas nécessaire dans le cas de l'AH.

2.3.1 Simple application numérique

Supposons une ligne image ayant des valeurs de niveaux de gris correspondant à la fonction : $I(x) = exp(-\frac{x^2}{2})$. La région à suivre est $\mathcal{R} = (x_1, x_2) = (-0.1, 0.0)$. Supposons maintenant que la transformation est une pure translation ($\mu = tx$). La fonction $\mathbf{f}(x, \mu)$ s'écrit alors $\mathbf{f}(x, \mu) = x + tx$.

La phase d'apprentissage consiste à produire 1000 perturbations aléatoires sur tx, en utilisant une loi uniforme dans un intervalle $[-.25, +.25]$, donnant des couples de valeurs de δtx et δi.

Dans le cas de l'approximation par hyperplans, nous obtenons $\mathbf{A}_h = (13.42, -13.41)$ directement. Dans le cas de l'approximation Jacobienne, nous calculons d'abord chaque valeur de la matrice Jacobienne $\mathbf{M}$. Les mêmes couples de vecteurs $\delta\mu^k$ et δi^k donnés par les perturbations aléatoires sont utilisés pour calculer $\mathbf{M}$. En écrivant $\mathbf{H} = (\delta\mathbf{i}^1, \ldots, \delta\mathbf{i}^{N_p})$, $\mathbf{Y} = (\delta\mu^1, \ldots, \delta\mu^{N_p})$, une estimation de $\mathbf{M}$ est donnée par : $\mathbf{M} = \mathbf{H}\mathbf{Y}^t(\mathbf{Y}\mathbf{Y}^t)^{-1}$, parce que $\mathbf{M}$ doit satisfaire $\mathbf{H} = \mathbf{M}\mathbf{Y}$ (équation (2.4)). Par conséquent, les mêmes données sont exactement utilisées dans les deux approximations (AH et AJ). Nous obtenons finalement $\mathbf{A}_j = (\mathbf{M}^t\mathbf{M})^{-1}\mathbf{M}^t = (11.05, 1.06)$.

Comparaison : il faut maintenant évaluer quelle est la meilleure modélisation entre AJ et AH. Ceci a été fait en prenant des valeurs de $\delta\mu^*$ dans l'intervalle $[-1, 1]$ de l'exemple précédent. Chacune de ces perturbations donne un vecteur $\delta\mathbf{i}$. Une estimation du mouvement $\delta\mu$ peut être obtenue en utilisant la relation $\delta\mu = \mathbf{A}\delta\mathbf{i}$. Si l'approximation était parfaite, nous devrions obtenir $\delta\mu = \delta\mu^*$.

Ces résultats sont illustrés par la figure 2.3. Cette figure représente $\delta\mu$ en fonction de $\delta\mu^*$. L'approximation par hyperplans donne visiblement et indéniablement de meilleurs résultats que

l'approximation Jacobienne.

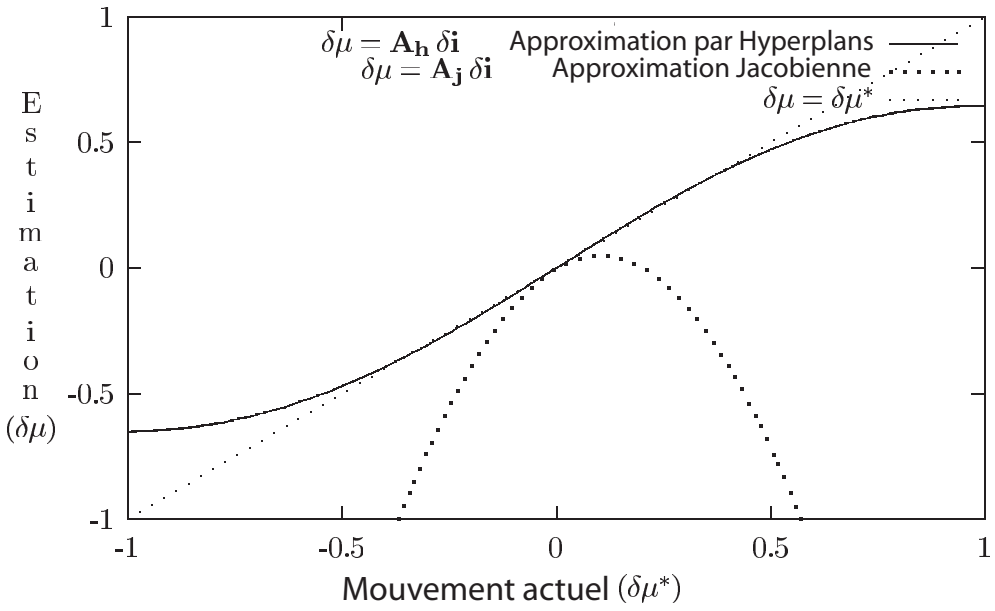


FIGURE 2.3 – comparaison entre l'approximation Jacobienne et l'approximation par hyperplans.

2.4 Suivi efficace à l'aide de l'approximation par hyperplans

Dans la section précédente, nous avons utilisé un exemple simple pour montrer la supériorité de l'Approximation par Hyperplans (AH) sur l'Approximation Jacobienne (AJ). Cependant, le schéma de l'AH est très inefficace, pris sous sa forme initiale. Le calcul direct de la matrice $\mathbf{A}_h$ implique une minimisation au sens des *moindres carrés*, laquelle doit être répétée pour chaque nouvelle image. La matrice dépend de la position courante, de l'orientation, etc. données par μ. La phase d'apprentissage consiste à calculer une relation linéaire entre un ensemble de différences de niveaux de gris et une correction des paramètres μ. Cette relation est calculée autour de la valeur μ_0^* (connue quand l'utilisateur sélectionne une région de l'image) et n'est pas valable pour d'autres valeurs de μ.

Nous allons voir comment il est possible d'établir cette relation pour n'importe quelle valeur de μ, sans recalculer la matrice.

2.4.1 Phase d'apprentissage

Soit la région définie par $\mathcal{R} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ l'ensemble des coordonnées des N points dans un référentiel local appelé *référentiel région*. La fonction $\mathbf{f}(\mathbf{x}; \mu)$ change les coordonnées de $\mathbf{x} = (x, y)$ dans le *référentiel région* en $\mathbf{u} = (u, v) = \mathbf{f}(\mathbf{x}; \mu)$ dans le *référentiel image*.

Quand l'utilisateur définit une région cible dans la première image, il définit un ensemble de correspondances entre des points dans le *référentiel région* et des points dans le *référentiel image* (par exemple, les coins de la région rectangulaire). Connaissant l'ensemble des correspondances, le calcul de μ_0^*, tel que $\mathbf{f}(\mathbf{x}; \mu_0^*)$ aligne le *référentiel région* sur la cible (définie dans le *référentiel image*), est alors possible.

La phase d'apprentissage consiste à produire de petites perturbations aléatoires $\delta\mu$ autour de μ_0^*. Ces perturbations, écrites sous la forme $\mu_0{}' = \mu_0^* + \delta\mu$ entraînent le changement de luminosité $\delta\mathbf{i} = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*)) - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0'))$.
Un ensemble de N_p perturbations $\delta\mu$ sont produites dans le but d'obtenir un modèle linéaire donnant $\delta\mu = \mathbf{A}_h\delta\mathbf{i}$. Durant la phase d'apprentissage, il est possible d'estimer le mouvement $\delta\mu$ connaissant $\delta\mathbf{i}$.

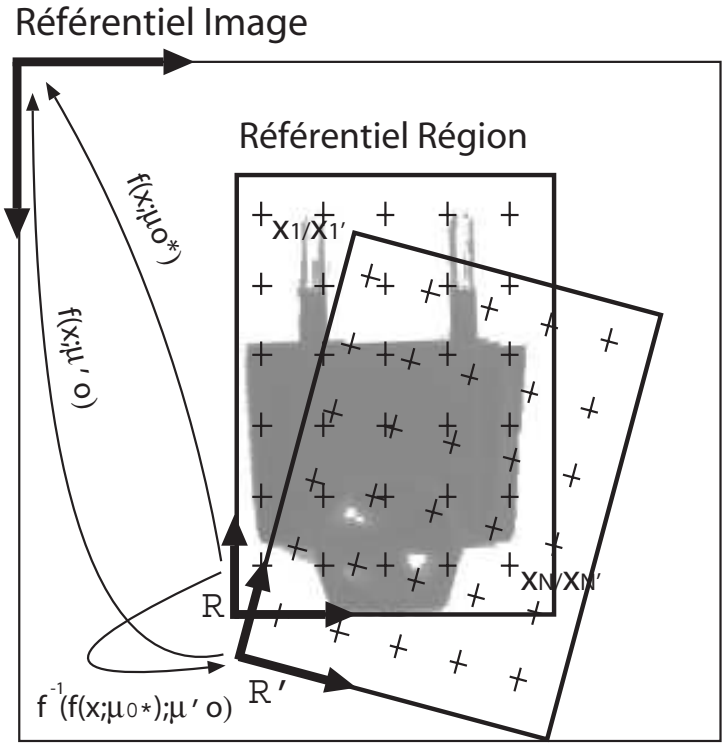


FIGURE 2.4 – phase d'apprentissage hors ligne.

Comme le montre la figure 2.4, si $\mathbf{u}'$ sont les coordonnées de $\mathbf{x}$ dans le *référentiel image* par la transformation μ_0', alors $\mathbf{u}' = \mathbf{f}(\mathbf{x}; \mu_0')$. Soit $\mathbf{x}'$ tel que $\mathbf{u} = \mathbf{f}(\mathbf{x}'; \mu_0')$. En supposant que $\mathbf{f}$ est inversible, nous obtenons $\mathbf{x}' = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}; \mu_0^*); \mu_0')$.

Par conséquent, connaissant $\delta\mathbf{i}$, nous pouvons estimer μ_0', et finalement calculer le déplacement de la région exprimé dans le *référentiel région*. Ce déplacement est seulement valable à proximité de μ_0^*.

2.4.2 Phase de suivi

Au début de la phase de suivi, une prédiction des paramètres est connue et est notée μ'. Le suivi consiste dans l'estimation de μ tel que :

$$\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu), t) = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0)$$

avec la notation $I_0(\mathbf{f}(\mathcal{R}; \mu_0^*)) = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0)$. Ici, le temps t est supprimé pour simplifier les notations.

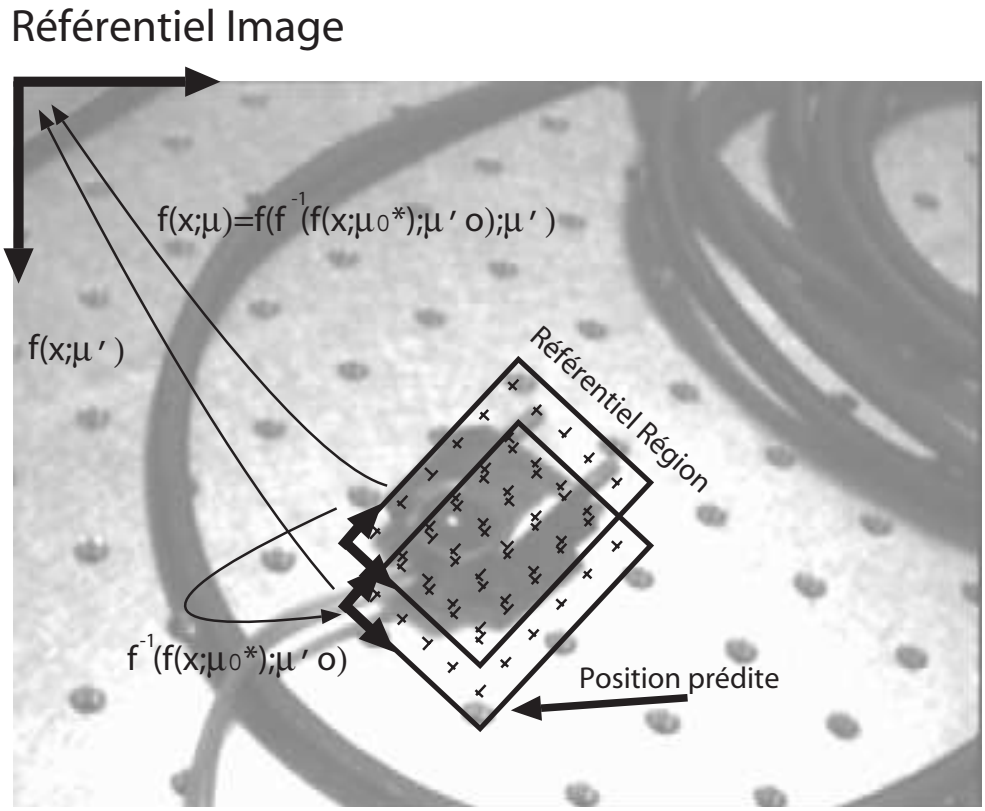


FIGURE 2.5 – phase de suivi en ligne.

En calculant :

$$\delta\mu = \mathbf{A_h}\delta\mathbf{i} = \mathbf{A_h}[\mathbf{I}_0(\mathbf{f}(\mathcal{R}; \mu_0^*)) - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu'))] \tag{2.10}$$

nous obtenons une perturbation $\delta\mu$ qui aurait produite une différence $\delta\mathbf{i}$ si le vecteur des paramètres avait été μ_0^*. Dans ce cas, une position $\mathbf{x}$ de la région est transformée en $\mathbf{x}' = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x};\mu');\mu_0^*)$, avec $\mu' = \mu_0^* + \delta\mu$.

L'actuelle modélisation transforme la position $\mathbf{x}$ en $\mathbf{u}$: $\mathbf{u} = \mathbf{f}(\mathbf{x},\mu)$. Comme décrite sur la figure 2.5, l'utilisation de l'équation $\mathbf{x}' = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x};\mu_0^*);\mu_0')$ dans la relation $\mathbf{u}' = \mathbf{f}(\mathbf{x}',\mu')$ donne :

$$\mathbf{u}' = \mathbf{f}(\mathbf{x}';\mu') = \mathbf{f}(\mathbf{f}^{-1}(\mathbf{f}(\mathbf{x};\mu_0^*);\mu_0');\mu') \tag{2.11}$$

Cette équation est fondamentale pour le suivi : elle donne la transformation alignant la région sur la cible à l'instant courant, connaissant une prédiction μ' et une perturbation locale $\delta\mu$. Cette perturbation locale autour de la valeur initiale μ_0^* est obtenue en échantillonnant l'image courante dans le *référentiel région* et en calculant $\delta\mathbf{i} = \mathbf{I}(\mathbf{f}(\mathcal{R},\mu_0^*),t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R},\mu'),t)$. L'équation (2.2) donne $\mu_0' = \mu_0^* + \mathbf{A}_h\delta\mathbf{i}$.

L'idée principale est alors de corriger la transformation de la région dans le *référentiel région* (en agissant comme si les paramètres étaient μ_0^*) et de transformer cette correction en lui appliquant μ'.

2.4.3 Modèles linéaires de mouvement

Translation, rotation et changement d'échelle : considérant un mouvement défini par une translation planaire (tx,ty), une rotation planaire (θ) et un facteur d'échelle (s), la relation entre un point de coordonnées (x,y) dans le *référentiel région* et un point correspondant de coordonnées (u,v) dans le *référentiel image* est donnée par :

$$\begin{cases} u = s\cos(\theta)x - s\sin(\theta)y + tx \\ v = s\sin(\theta)x + s\cos(\theta)y + ty \end{cases}$$

Ce système d'équations peut être écrit sous forme matricielle (produit de matrices) en utilisant les coordonnées homogènes. Soient $(x,y,1)^t$ les coordonnées d'un point dans le *référentiel région* et $(\lambda u,\lambda v,\lambda)^t$ ses coordonnées dans le *référentiel image*. Nous obtenons alors le système suivant :

$$\begin{pmatrix} \lambda u \\ \lambda v \\ \lambda \end{pmatrix} = \mathbf{F}(\mu) \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \; avec\; \mathbf{F}(\mu) = \begin{pmatrix} s\cos(\theta) & -s\sin(\theta) & tx \\ s\sin(\theta) & s\cos(\theta) & ty \\ 0 & 0 & 1 \end{pmatrix}$$

Homographie : en gardant les mêmes notations (coordonnées homogènes), les mouvements homographiques peuvent être modélisés par un modèle à huit paramètres donné par la matrice suivante :

$$\mathbf{F} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$$

Généralisation des modèles linéaires de mouvement : dans le cas d'un modèle linéaire de mouvement, nous avons vu que la fonction $\mathbf{f}$ peut être écrite comme un produit de matrices :

$$\mathbf{f}(\mathbf{x}; \mu) = \mathbf{F}(\mu)\mathbf{x}$$

où $\mathbf{x}$ est écrit avec des coordonnées homogènes $\mathbf{x} = (sx, sy, s)$, et $\mathbf{F}$ est une matrice de dimensions 3×3.

Dans ce cas, l'équation (2.11) devient :

$$\mathbf{F}(\mu) = \mathbf{F}(\mu')\mathbf{F}^{-1}(\mu_0{}')\mathbf{F}(\mu_0^*) \tag{2.12}$$

où $\mathbf{F}(\mu_0{}') = \mathbf{F}(\mu_0^* + \mathbf{A}_h\delta\mathbf{i})$. $\mathbf{F}(\mu')$ est la transformation obtenue à l'image précédente. $\mathbf{F}(\mu_0^*)$ est calculée à partir de la sélection de la région dans l'image initiale (c'est à dire la première image). La matrice $\mathbf{A}_h$ est estimée durant la phase d'apprentissage hors ligne. Finalement, la perturbation $\delta\mu$ est donnée par l'équation (2.10).

2.5 Expérimentations et Résultats

Les expérimentations suivantes montrent la supériorité de l'Approximation par Hyperplans (AH) sur l'Approximation Jacobienne (AJ).
Nous avons testé trois modèles différents de mouvement :

1. Translation planaire, Rotation planaire et changement d'Échelle (TRE),
2. transformation affine,
3. transformation homographique.

Dans cette section, nous présentons seulement les résultats concernant le premier modèle de mouvement (TRE), car les quatres paramètres qui apparaissent dans cette transformation peuvent être intuitivement perçus. Ce n'est pas le cas pour les six paramètres de la transformation affine ou pour les huit paramètres du mouvement homographique. De plus, nous obtenons

dans chacun des cas le même type de résultat.
Ces algorithmes ont été implémentés sur une station de travail Silicon Graphics O_2. 400 points à l'intérieur d'une zone elliptique sont suivis pour une acquisition image toutes les 33 ms. Les traitements prennent moins de 10 ms dans les deux cas (AH et AJ).

2.5.1 Comparaison entre AJ et AH : résultats sur une image statique

Cette expérimentation consiste à sélectionner une région dans une image, à apprendre la matrice $\mathbf{A}$ par les deux méthodes d'approximation, et finalement à valider la matrice en déplaçant la région et en comparant le mouvement réel avec celui estimé par $\delta\mu = \mathbf{A}\delta\mathbf{i}$.

FIGURE 2.6 – image test et région cible.

L'estimation de la matrice Jacobienne et l'estimation par hyperplans doivent être faites dans les mêmes conditions, sinon cette comparaison n'aura aucune signification. De petites perturbations autour de la position de référence sont produites, en utilisant une loi uniforme aléatoire. Le même ensemble de perturbations est exactement utilisé pour calculer les deux approximations.

Nous avons conduit cette expérimentation sur plusieurs images et avons obtenu le même type de résultat. Nous présentons des résultats obtenus sur une seule image (figure 2.6). Les

graphiques de la figure 2.7 comparent les approximations obtenues avec AJ et AH. Dans les deux cas, l'approximation linéaire (phase d'apprentissage) a été faite autour de la transformation initiale, dans un intervalle de +/- 20 pixels pour les translations, +/- 10 degrés pour la rotation et +/- 10 pixels pour l'échelle. La variation d'échelle est calculée en fonction de la taille de la région dans la première image.

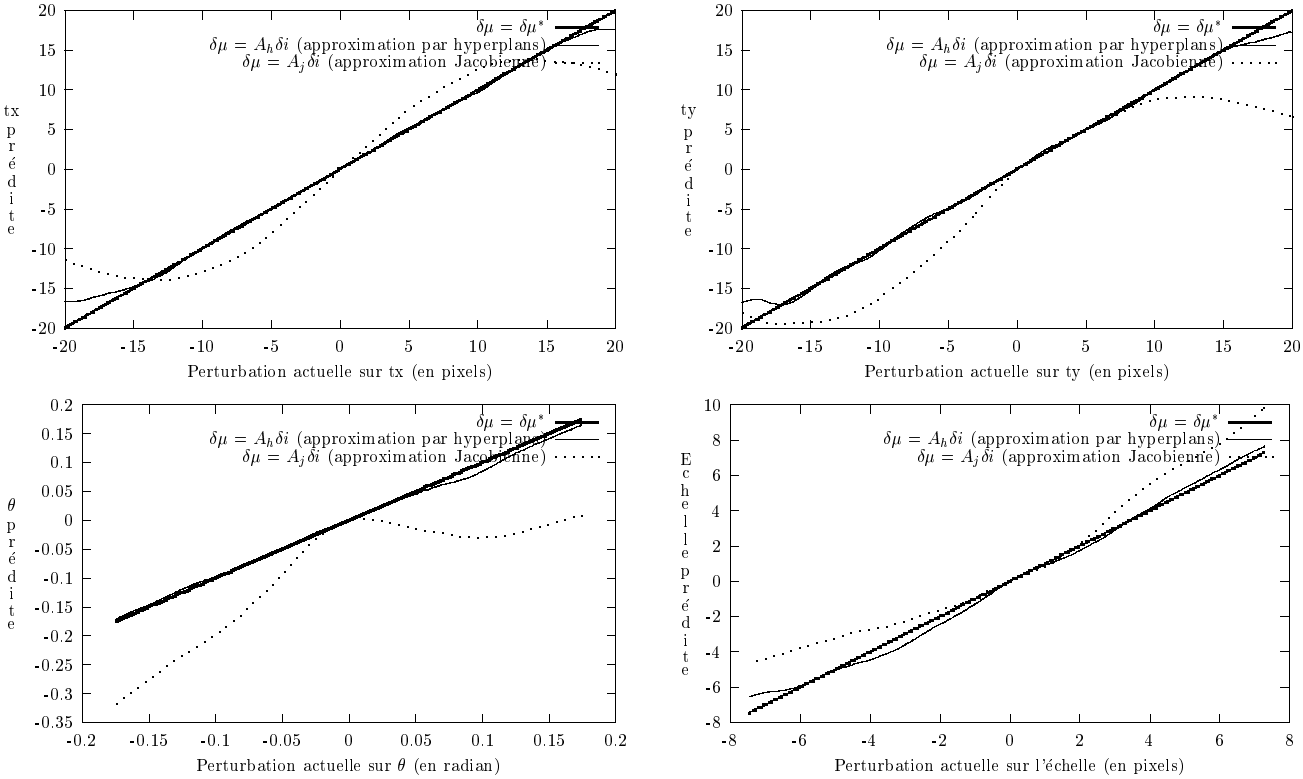


FIGURE 2.7 – comparaison des méthodes AJ et AH sur un modèle de mouvement de type TRE.

Nous avons clairement observé que l'approximation donnée par l'AH est de loin la meilleure. Pour exemple, dans le cas d'une translation de 15 pixels, plusieurs itérations sont nécessaires à l'AJ pour compenser cette translation alors qu'une seule itération suffit pour l'AH. Notre approche par hyperlans permet donc d'étendre la zone de convergence.

Cela signifie que dans les applications de suivi, l'algorithme proposé sera capable de suivre des objets avec des déplacements de grandes amplitudes.

2.5.2 Expérimentations sur des objets en mouvement

Séquences vidéos réelles : les algorithmes ont été implémentés sur une station de travail Silicon Graphics O_2 équipée d'un processeur R5000 cadencé à une fréquence de 150Mhz. 100

points à l'intérieur d'une zone polygonale sont suivis pour une acquisition image toutes les 33 ms. Les traitements prennent moins de 10 ms. Le mouvement est supposé être une homographie. Pour illustrer notre méthode de suivi 2D, nous utilisons des objets volumiques présentant des surfaces planaires. Ceci nous est imposé par le modèle de mouvement qui suppose que les points caractéristiques du motif suivi (les points où sont échantillonnés les niveaux de gris) appartiennent au même plan.

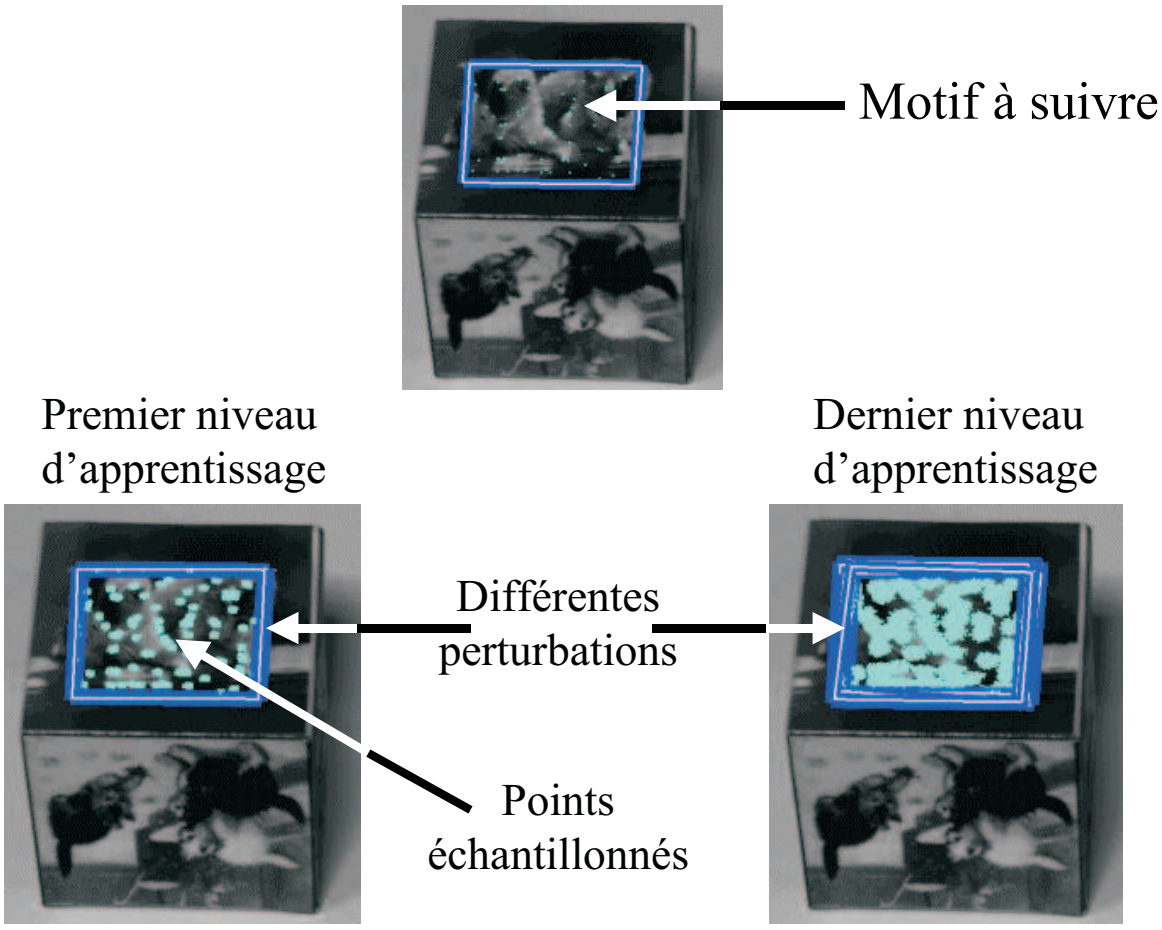


FIGURE 2.8 – phase d'apprentissage multiéchelle hors ligne.

Dans ce cas, nous utilisons une stratégie à quatre niveaux d'approximation appliqués successivement. Chacun a été appris distinctement, en appliquant un niveau différent de perturbations. Les amplitudes de ces perturbations ont été fixées respectivement à 20%, 10%, 5% et 1% de la taille de la région. Cette phase d'apprentissage hors ligne est illustrée par la figure 2.8. Cette approche multiéchelle a été utilisée pour augmenter la précision. L'échelle la plus grande permet, avec un manque de précision, de suivre les mouvements de fortes amplitudes ; l'échelle la plus petite, quant à elle, permet de suivre avec précision les mouvements de faibles amplitudes.

Les figures 2.9, 2.10, 2.11 et 2.12 présentent quatre séquences de suivi 2D en temps réel vidéo et ceci pour différents motifs visuels. Les images ont été sélectionnées pour montrer la robustesse de notre méthode aux variations possibles d'apparence du motif suivi dues aux changements d'orientation caméra/objet et aux déplacements de grandes amplitudes entre deux images consécutives.

FIGURE 2.9 – suivi 2D temps réel : séquence 1.

En particulier, dans la dernière séquence, l'objet est en rotation avec une vitesse supérieure à 760 degrés par seconde (15 degrés par image). A notre connaissance, aucun des algorithmes proposés auparavant peut suivre des objets à une telle vitesse de déplacement. Ces résultats obtenus nous ont encouragés à poursuivre nos travaux et à développer une approche de suivi 3D que nous présenterons dans le prochain chapitre. Mais auparavant, nous proposons une solution pour traiter le problème des occultations.

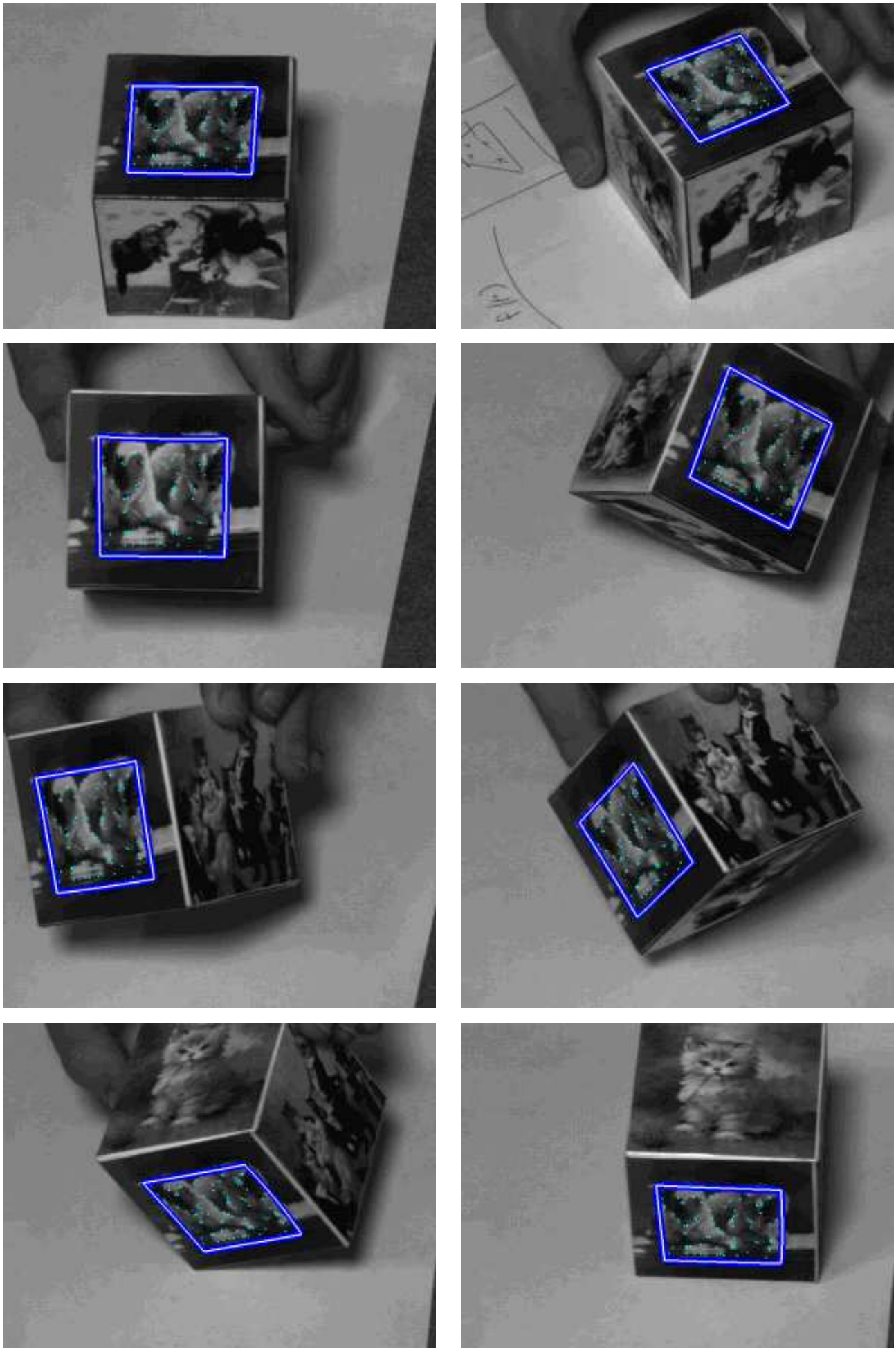


FIGURE 2.10 – suivi 2D temps réel : séquence 2.

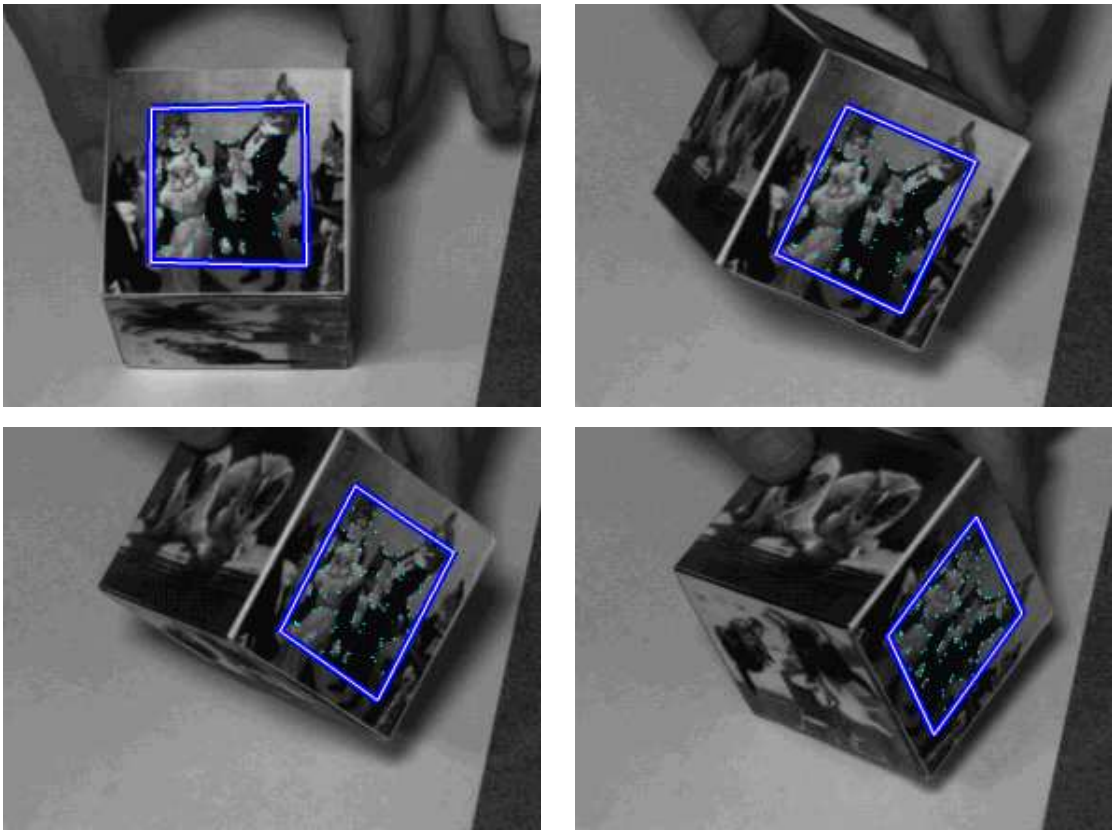


FIGURE 2.11 – suivi 2D temps réel : séquence 3.

FIGURE 2.12 – suivi 2D temps réel : séquence 4.

2.6 Traitement des occultations

Dans les **méthodes de suivi basées sur une modélisation photométrique** de l'objet rigide, la phase de mise en correspondance du modèle de l'objet actuellement suivi avec une région de l'image courante est très délicate. Elle peut s'avérer inexacte dans les situations suivantes :

1. le modèle ne représente plus l'apparence de l'objet courant à cause des changements d'orientation de l'objet ou des conditions d'illumination de la scène.
2. l'objet est partiellement visible du à l'apparition d'occultations.

Un algorithme de suivi devra être capable de traiter à la fois ces deux situations. Dans la première situation, l'algorithme devra mettre à jour le modèle pour satisfaire le changement d'apparence de l'objet. Dans le second cas, il devra être capable de détecter l'occultation et continuer à capturer l'objet jusqu'à ce qu'il redevienne entièrement visible. Cette *occultation inter-objets* se produit donc quand un ou plusieurs objets cachent la vue d'un ou plusieurs objets à l'intérieur du champ de vision de la caméra.

Pour garantir la stabilité et la robustesse face aux occultations, plusieurs méthodes de suivi utilisent un filtre de Kalman [90] [53] [14] [91]. Dans toutes ces méthodes, le filtre de Kalman permet de suivre la position de l'objet pour un modèle de mouvement donné. La position de l'objet est prédite dans l'image courante. Pour vérifier la prédiction, une technique d'analyse d'image indépendante est alors employée pour détecter l'objet. Dans une telle approche, le bénéfice dans l'utilisation du filtre de Kalman est seulement de contrôler (lisser) la trajectoire de l'objet. Le filtre a très peu d'effet sur l'amélioration de la détection de l'objet dans l'image, ce qui est crucial pour garantir la robustesse du suivi.

De nombreux travaux sur la **détection des occultations** se sont concentrés sur des méthodes de haut niveau utilisant les *"line drawings"*. Les approches les plus notables déduisent les occultations à partir des jonctions ou liaisons réalisées entre les différents segments de droite modélisant les contours de l'objet [61] [105] [65]. Horn [12] a indiqué brièvement la possibilité de reconnaître une arête cachée en analysant son profil d'intensité. Riley [82] a mentionné qu'un changement abrupt dans la texture pouvait signaler l'apparition d'une arête occultée. Pentland [5] a démontré qu'en utilisant des informations locales sur l'ombre, des contours lisses occultés pouvaient être détectés. Thompson et Whillock [111] ont utilisé des informations sur le mouve-

ment et le contraste pour détecter des surfaces occultées et ont amélioré ainsi la reconnaissance (basée sur un modèle) d'objets partiellement occultés. Nakayama et *al.* [69] ont suggéré de classer les occultations en intégrant uniquement la notion de profondeur. Toh et Forrest [107] ont montré qu'une occultation peut être directement détectée à partir des données contenues dans l'image en intégrant la notion de *discontinuité de profondeur*. Dans ce cas précis, il est supposé que la *discontinuité de profondeur* entre deux surfaces apparaîtra comme une *discontinuité d'intensité* dans l'image 2D. Pour cela, ils ont fusionné deux techniques : l'une basée sur la détection de rivalité au cours d'une fixation binoculaire (*"binocular fixation and rivalry method"*), l'autre utilise l'information focale qui résulte d'un changement de profondeur de champ. La faisabilité de la première technique suppose que la *fusion stéréo* et la *rivalité* coexistent dans la vision humaine. Cette méthode démontre donc que la détection des occultations est inhérente dans la vision binoculaire et suggère que de tels mécanismes existent dans la vision humaine. La deuxième méthode dite *"méthode de profondeur de champ"* n'implique pas des changements de point de vue et évite ainsi le problème des correspondances à la différence de la première. Notons également que les techniques de détection d'occultations peuvent être basées sur une minimisation d'erreur de mise en correspondance d'intensité, définie comme une somme des carrés des différences (*sum-of-squared difference* SSD) entre l'image et un modèle [53] [91].

2.6.1 Méthode de seuillage adaptatif [46]

Avant de présenter notre solution, nous adopterons les notations suivantes :

- $VI_{ref} = I_0(\mathbf{f}(\mathcal{R}; \mu_0^*))$ sera le vecteur de niveaux de gris correspondant à l'échantillonnage du motif de référence que l'on souhaite suivre dans la zone d'intérêt.
- VI_c correspondra soit à $\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0'))$ durant la phase d'apprentissage hors ligne (position perturbée autour de μ_0^*), soit à $\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu'))$ durant la phase de suivi en ligne (position prédite). En fait, VI_c est le vecteur de niveaux de gris correspondant à l'échantillonnage du motif courant dans la zone d'intérêt.
- $\Delta VI = \delta i = VI_{ref} - VI_c$ est la différence de niveaux de gris entre les deux vecteurs de forme.
- $A = A_h$ est la matrice apprise lors de la phase d'apprentissage hors ligne en utilisant la méthode d'approximation par hyperplans.

La solution mise en œuvre est basée sur les travaux de Gregory D. Hager et Peter N. Belhumeur [60]. La méthode de détection et de traitement des occultations est une méthode de seuillage appliquée au vecteur de différence de niveaux de gris ΔVI entre le vecteur de référence

suivi VI_{ref} et le motif courant dans l'image VI_c. Ces seuils, calculés pour chaque élément du vecteur de résidus, sont aussi regroupés dans un vecteur dit vecteur de seuil adaptatif VI_s. Ce vecteur VI_s est appris lors de la phase d'apprentissage pour le calcul de la matrice A.

A partir du vecteur de référence choisi $VI_{ref} = (i_{ref1}, i_{ref2}, \ldots, i_{refN})^t$ et des différents vecteurs courants $VI_c^j = (i_{c1}^j, i_{c2}^j, \ldots, i_{cN}^j)^t$ échantillonnés sur N points ($N = 373$) après M perturbations des paramètres de la transformation affine ($M = 1000$), nous estimons $VI_s = (i_{s1}, i_{s2}, \ldots, i_{sN})^t$ comme un vecteur moyen.

Un élément k du vecteur VI_s est donné par la formule suivante :

$$i_{sk} = \frac{1}{M} \sum_{p=1}^{M} (i_{refk} - i_{ck}^p) \tag{2.13}$$

Nous estimons alors le vecteur des écarts types $V\sigma_{Is} = (\sigma i_{s1}, \sigma i_{s2}, \ldots, \sigma i_{sN})^t$.

Un élément k du vecteur $V\sigma_{Is}$ est donné par la formule suivante :

$$\sigma i_{sk} = \sqrt{\frac{1}{M} \sum_{p=1}^{M} [(i_{refk} - i_{ck}^p) - i_{sk}]^2} \tag{2.14}$$

Nous traitons donc les problèmes d'occultations en supposant que pour chaque point échantillonné, la variation de niveau de gris suit une distribution gaussienne (moyenne et écart type). Dans cette méthode, les occultations détectées doivent entraîner de fortes variations de niveaux de gris dans le vecteur de résidus ΔVI dont la normalisation est fonction de la mise à jour de la matrice diagonale de masquage W pondérant les erreurs dues aux occultations. Chaque terme diagonal w_{jj} peut prendre une valeur comprise entre 0 (outlier) et 1 (inlier) calculée en comparant la valeur absolue $|\Delta i_j - i_{sj}|$ (Δi_j étant la différence de niveaux de gris $i_{refj} - i_{cj}$) avec la valeur de l'écart type σi_{sj} associée (voir figure 2.14).

L'algorithme de suivi pour un motif de référence donné traitant le problème d'occultations pour des mouvements de l'objet parallèles au plan image est illustré figure 2.13.

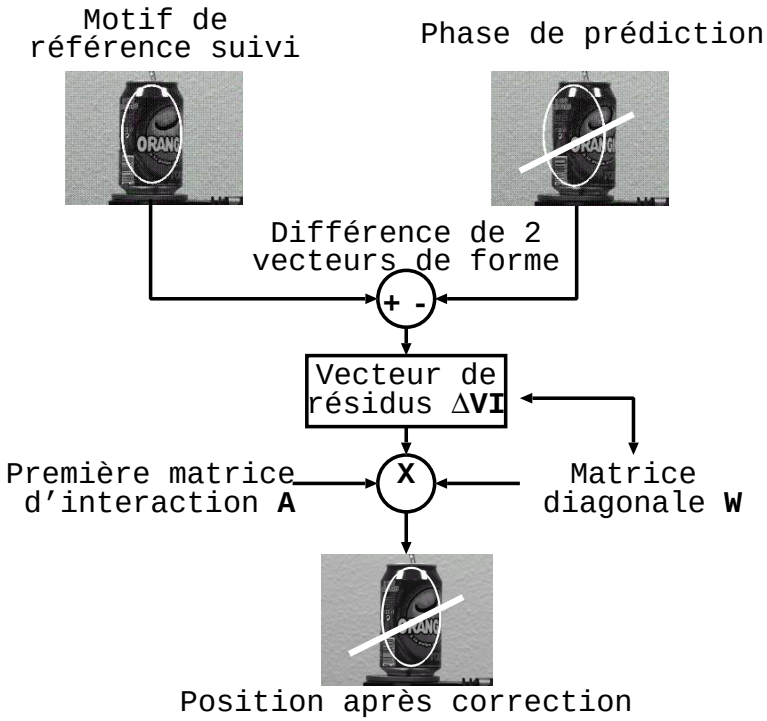


FIGURE 2.13 – principe du traitement des occultations dans la phase de suivi.

Cet algorithme peut s'écrire de la manière suivante :

DEBUT PROCEDURE TRAITEMENT OCCULTATIONS (μ' (position prédite), $Image$)

VI_c = Echantillonner $(\mu', Image)$

Pour $l = 1$ à L itérations faire

ΔVI_l = Normaliser$(VI_{ref}, VI_c, W_{l-1})$

Pour $j = 1$ à N éléments par vecteur

val.absolue = $|\Delta i_{jl} - i_{sj}|$

Si val.absolue $\leq 2\sigma i_{sj}$

alors $w_{jjl} = 1$ (Inliers)

Si $2\sigma i_{sj} <$ val.absolue $\leq 5\sigma i_{sj}$

alors $w_{jjl} = (1 - \frac{val.absolue - 2\sigma i_{sj}}{3\sigma i_{sj}})^l$

Si val.absolue $> 5\sigma i_{sj}$

alors $w_{jjl} = 0$ (Outliers)

Fin Pour

Fin Pour

$\mu = \mu' + AW_L\Delta VI_L$ (position réelle)

FIN PROCEDURE TRAITEMENT OCCULTATIONS

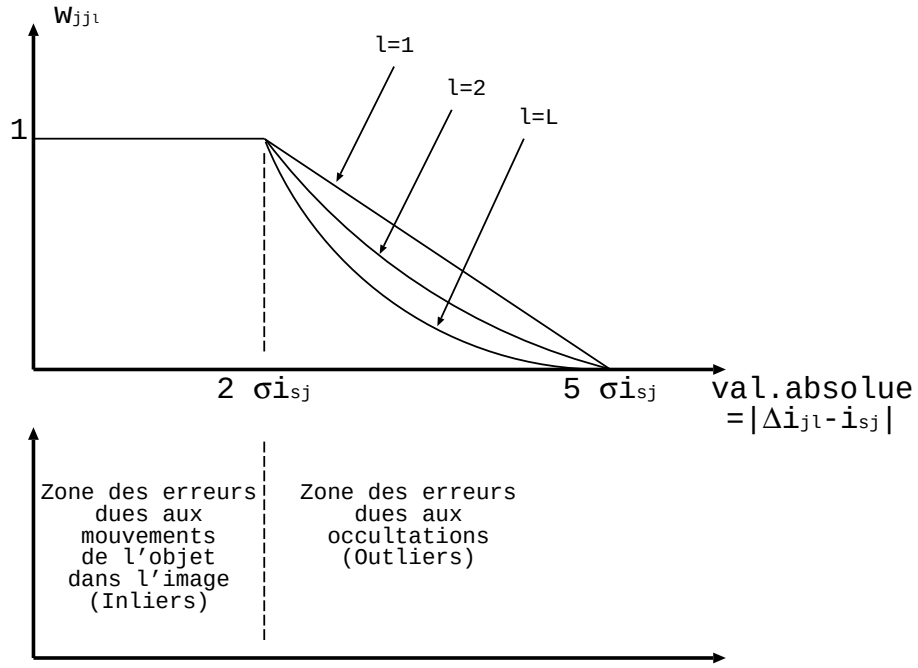


FIGURE 2.14 – recherche des Inliers et des Outliers dans le vecteur de résidus ΔVI.

La matrice diagonale W filtre les erreurs engendrées par les occultations dans le vecteur de résidus ΔVI comme le montre la figure 2.14 pour le j^{eme} élément d'un vecteur de forme à la l^{eme} itération de détection.

2.6.2 Essais

Dans la figure 2.15, nous présentons maintenant quelques images d'une séquence de suivi avec détection et suppression des occultations. Deux objets sont utilisés : une canette de soda et une figurine. Ces deux objets ont été choisis car ils sont différents :

1. par leur forme volumique : un cylindre et une tête avec des traits de caractère accentués,
2. par leur apparence : des transitions de niveaux de gris (ndg) beaucoup plus importantes pour la figurine (par exemple, passage du ndg 0 (noir) au ndg 255 (blanc) au niveau des moustaches),
3. par leur réflectance à la lumière : objet métallique (brillant) pour la canette et objet en plâtre pour la figurine (mat).

Le modèle de mouvement utilisé est une transformation affine. Cette transformation définie comme une homographie particulière est moins sensible aux déformations que peut subir le motif visuel suivi après un changement d'orientation caméra/objet. Le choix de prendre une zone elliptique pour échantillonner le motif sera justifié dans le prochain chapitre.

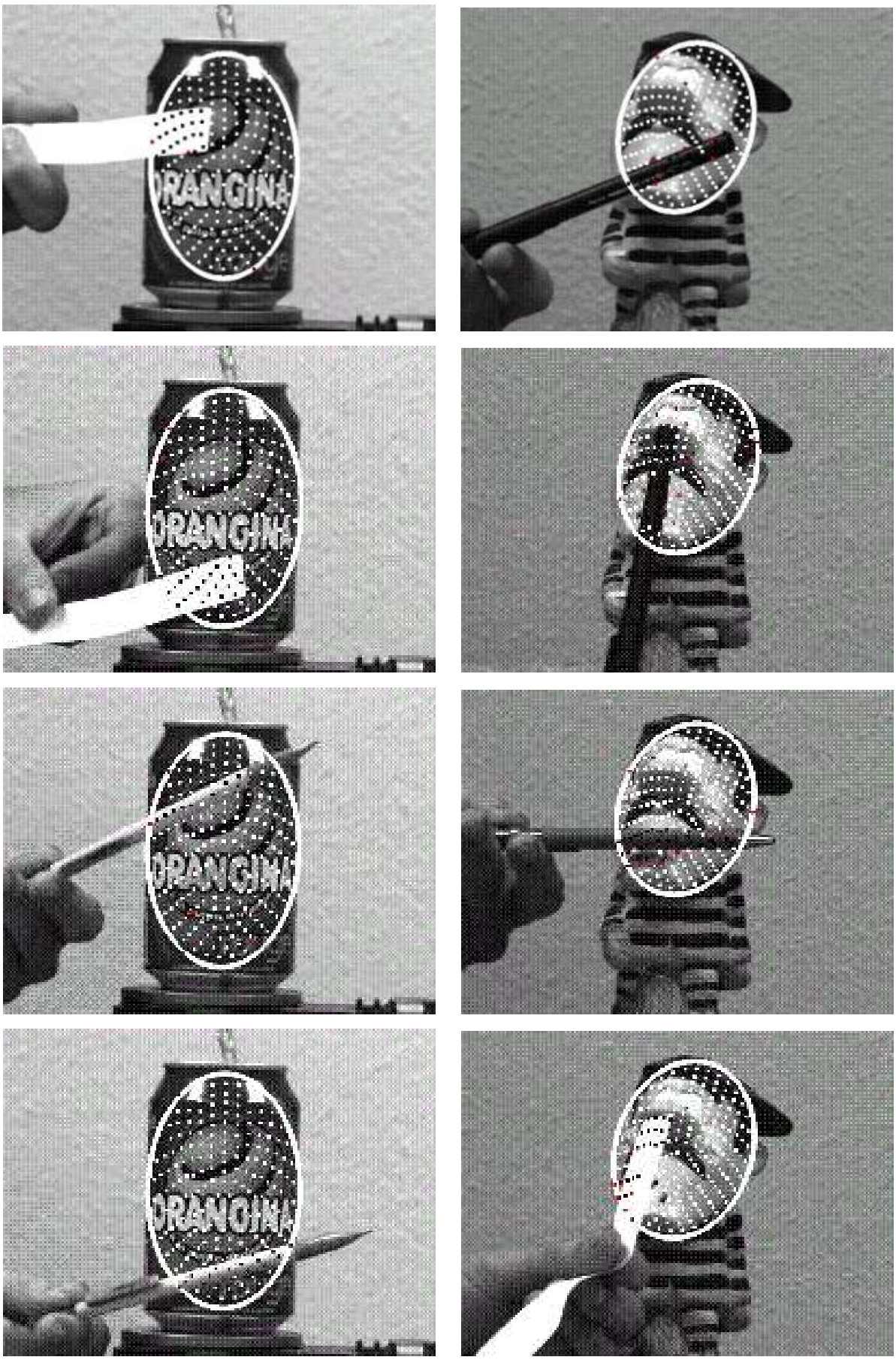


FIGURE 2.15 – quelques exemples de détection d'occultations.

Les fortes variations dans le vecteur de différence ΔVI sont bien détectées et traitées comme une apparition d'occultation (points noirs sur l'image). Le positionnement de l'ellipse sur le motif est correct. Le mérite de cette approche est de pouvoir suivre en temps réel l'objet dans la séquence d'images puisqu'elle est peu coûteuse en temps de calcul ($<$ à 20 millisecondes avec la détection des occultations) tout en minimisant l'influence des occultations. Bien entendu, comme nous travaillons avec un vecteur de résidus ΔVI pour corriger la position prédite de l'objet dans l'image, les occultations ne peuvent recouvrir que partiellement l'aspect du motif. Trop de mesures aberrantes dans le vecteur de résidus ΔVI ne nous permettraient plus d'assurer un suivi correct de l'objet dans l'image car même filtrées, la richesse des informations recueillies serait trop faible.

Suivant la valeur des termes diagonaux w_{jj} de la matrice W, la couleur des points d'échantillonnage sur les images de la figure 2.15 est différente :

- *point noir* : l'occultation est détectée. Ce point est considéré comme un *outlier* et le terme w_{jj} est égal à 0.
- *point blanc* : aucune occultation n'a été détectée. Ce point est considéré comme un *inlier* et le terme w_{jj} est égal à 1.
- *point rouge* : la valeur du terme w_{jj} est comprise entre 0 et 1.

2.7 Conclusion

Dans ce chapitre, nous avons d'abord présenté une alternative de l'algorithme de suivi de Hager *et al.* [60]. L'idée *clef* est de remplacer l'approximation Jacobienne par une approximation par hyperplans. Nous montrons comment les différents calculs réalisés au cours d'une phase d'apprentissage hors ligne nous permettent d'assurer le suivi d'un motif visuel en temps réel vidéo. Avec ces améliorations, la vitesse de convergence est multipliée par 3 ou 4, comparée à l'algorithme de suivi de Hager *et al.* [60]. Basée sur ces mêmes travaux, une méthode de seuillage adaptatif pour le traitement des occultations a été développée.

C'est une méthode généraliste car le choix du modèle de mouvement est à l'initiative de l'utilisateur. C'est une technique de suivi basée sur l'apparence qui reste sensible à l'apparition d'occultations et aux variations d'éclairement en traitant le motif dans sa globalité.

Nous nous sommes ensuite activement intéressés au problème du suivi 3D pour différents points de vue, en modélisant les objets avec un ensemble d'apparences (collection d'images 2D pour représenter un objet 3D). Cet algorithme devra être capable de suivre des objets 3D pour n'importe quel mouvement 3D. Ceci fait donc l'objet du prochain chapitre.

Chapitre 3

Extension de l'approximation par hyperplans au suivi 3D d'objets

Dans ce chapitre, nous allons développer l'approche de suivi 3D d'objets mobiles dans une séquence d'images, basée sur l'apparence et l'approximation par hyperplans (AH). Notons que ces travaux ont été positionnés dans le premier chapitre comme étant une approche globale basée sur un modèle photométrique où le problème du suivi est formulé comme un problème de recherche du meilleur ensemble de paramètres (au sens des *moindres carrés*) décrivant au mieux le mouvement de la cible au cours de la séquence.

Pour cela, l'objet 3D est représenté par une collection d'images 2D appelées *vues de référence*. Nous rappelons ici que l'une des caractéristiques de cette méthode est de ne pas utiliser des primitives visuelles (segments...) pour suivre le déplacement de l'objet dans l'image mais plutôt la différence de vecteurs de niveaux de gris entre le motif de référence suivi et le motif courant échantillonné dans une zone d'intérêt de l'image. Le problème du suivi se ramène alors à l'estimation des paramètres qui caractérisent les mouvements possibles de l'objet dans l'image par la détermination de matrices dites *"d'interaction"* apprises lors d'une phase d'apprentissage hors ligne, et cela pour chacune des *vues de référence*.
Nous montrons que l'utilisation en ligne de ces matrices pour la correction de la position prédite de l'objet dans l'image et de l'estimation des variations d'aspect du motif suivi correspond à un coût algorithmique très faible (multiplication d'une matrice par un vecteur) permettant une mise en oeuvre temps réel. Cet algorithme efficace de suivi 3D que nous allons maintenant proposer a fait l'objet de différentes publications [43] [46] [48] [45] et de démonstrations [42] [44].

3.1 Présentation de la solution de suivi 3D

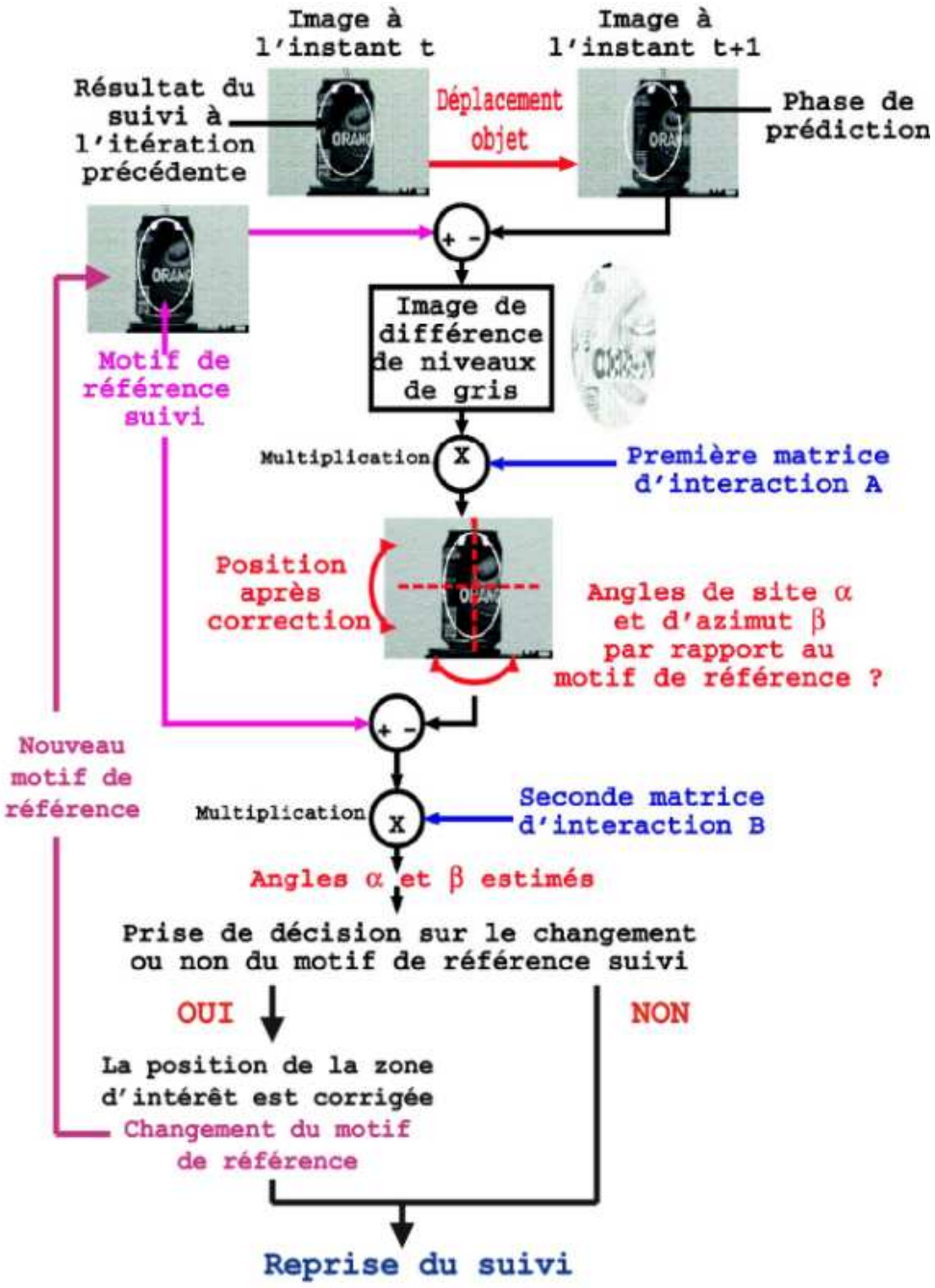


FIGURE 3.1 – principe du suivi 3D d'objets.

La solution de suivi 3D que nous allons présenter (figure 3.1) consiste à multiplier la différence de niveaux de gris entre le motif observé à l'endroit prédit et le motif de référence par une première *matrice d'interaction A* apprise lors d'une phase d'apprentissage hors ligne, pour corriger les erreurs sur les *mouvements fronto parallèles* de l'objet dans l'image. Par définition, un *mouvement fronto parallèle* est un mouvement tel que l'objet se déplace dans des plans parallèles au plan image. Sous l'hypothèse d'un tel mouvement, l'aspect apparent de l'objet suivi

ne subit pas de modification majeure ; toutefois, sa position, son orientation planaire et sa taille peuvent changer. Pour cela, nous supposons également que la taille de l'objet selon la direction d'observation reste faible par rapport à la distance objet-caméra (utilisation d'une caméra à focale longue). Ainsi, une translation 3D (même selon l'axe optique) de l'objet dans la scène correspondra à une similitude 2D dans l'image (*mouvement fronto parallèle*). Le problème du suivi du motif dans l'image se ramène alors à la correction des paramètres d'une transformation géométrique planaire par la détermination d'un vecteur d'offset.

La position prédite étant corrigée, les variations d'aspect du motif courant par rapport au motif de référence suivi dues aux orientations 3D relatives (*site* et *azimut*) de l'objet sont estimées en multipliant une seconde *matrice d'interaction* B, toujours apprise lors de la phase d'apprentissage, par la différence entre le motif courant et le motif de référence. Ces informations ainsi obtenues nous permettent de passer d'un motif de référence à l'autre tout en continuant de suivre l'objet dans l'image en temps réel vidéo (< 15 ms par itération).

Compte tenu de la rapidité des traitements (multiplication d'une matrice par un vecteur) par rapport à la vitesse de déplacement des objets dans les séquences d'images, nous n'avons pas besoin d'utiliser d'algorithme de prédiction de mouvement. En effet, l'écart de position du motif entre deux images successives reste compatible avec les variations apprises lors de la phase d'apprentissage. Quant au choix de décomposer le déplacement 3D de l'objet en plusieurs mouvements, il sera justifié par la suite, lors du développement théorique de la phase d'apprentissage hors ligne des *matrices d'interaction*.

La suite de ce chapitre se décompose en trois parties. Dans un premier temps, nous voyons comment modéliser l'objet 3D et son apparence, puis nous développons le suivi 3D en plusieurs points. Tout d'abord, nous définissons les paramètres qui caractérisent les mouvements possibles de l'objet dans l'image et leurs interprétations géométriques dans le suivi, puis nous introduisons la notion de *matrices d'interaction* calculées lors d'une phase d'apprentissage hors ligne ainsi que l'évaluation des performances de la méthode. Nous terminons par la gestion du passage d'un motif de référence à un autre et l'optimisation de la taille du motif suivi dans l'image. Dans une dernière partie, nous présentons des exemples concrets de suivi 3D d'objets volumiques sans occultation.

3.2 Modélisation de l'apparence de l'objet 3D

L'apparence d'un objet rigide dans une image dépend :

1. de sa forme et de sa réflectance [88] qui sont ses propriétés intrinsèques,
2. de sa pose et des conditions d'illumination de la scène qui varient au cours du suivi.

3.2.1 Construction du modèle d'un objet 3D

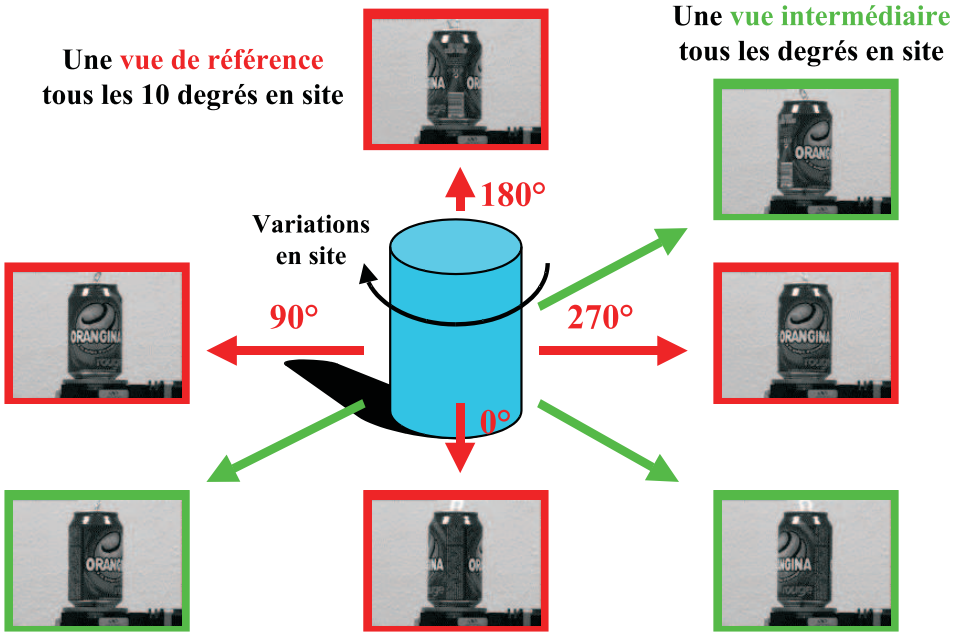


FIGURE 3.2 – exemple de construction du modèle d'un objet 3D.

Dans notre approche de suivi, un objet 3D est représenté par une collection d'images 2D appelées *vues de référence* (figure 3.2). Chacune de ces images représente un des motifs de référence de l'objet 3D à un instant du suivi pour une position caméra/objet donnée. Chaque *vue de référence* permet d'estimer une *matrice d'interaction* A lors de la phase d'apprentissage hors ligne et d'effectuer ensuite le suivi 2D en ligne d'un motif.

L'acquisition de vues intermédiaires nous permettra lors de la phase d'apprentissage hors ligne de calculer une *matrice d'interaction* B pour chacune des *vues de référence*. Durant la phase de suivi en ligne, cette matrice est alors utilisée pour estimer les variations d'aspect du motif courant dans l'image par rapport au motif de référence suivi. Ce changement d'apparence est déterminé à partir de deux valeurs angulaires en site et azimut notées respectivement α et β.

En pratique, nous utilisons une table à déplacement micrométrique (figures 3.3 et 3.4) pour photographier nos objets 3D et créer nos collections d'images 2D. Cette table, commandée à distance, permet de contrôler précisément la pose de l'objet dans l'image.

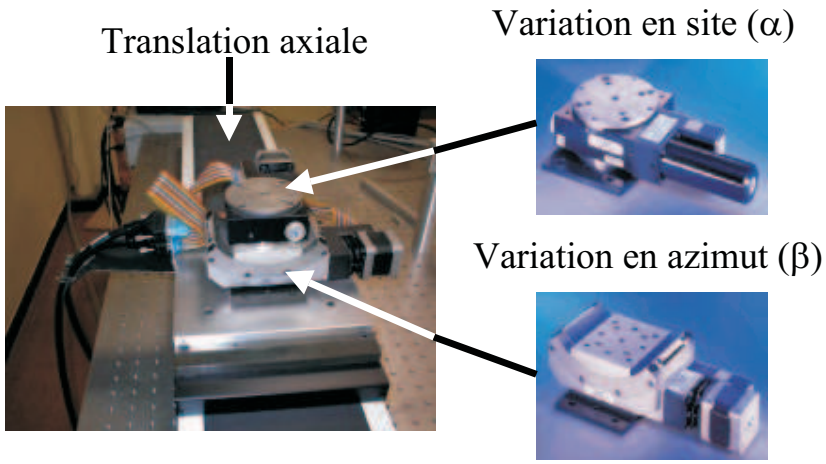


FIGURE 3.3 – mouvements autorisés par la table à déplacement micrométrique.

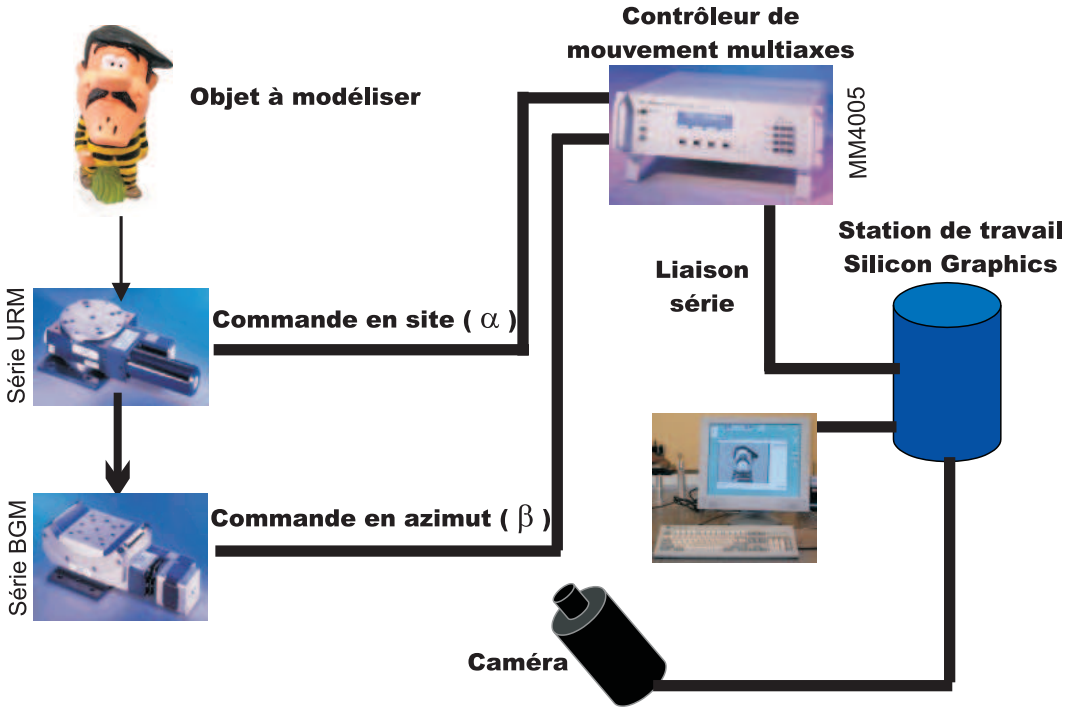
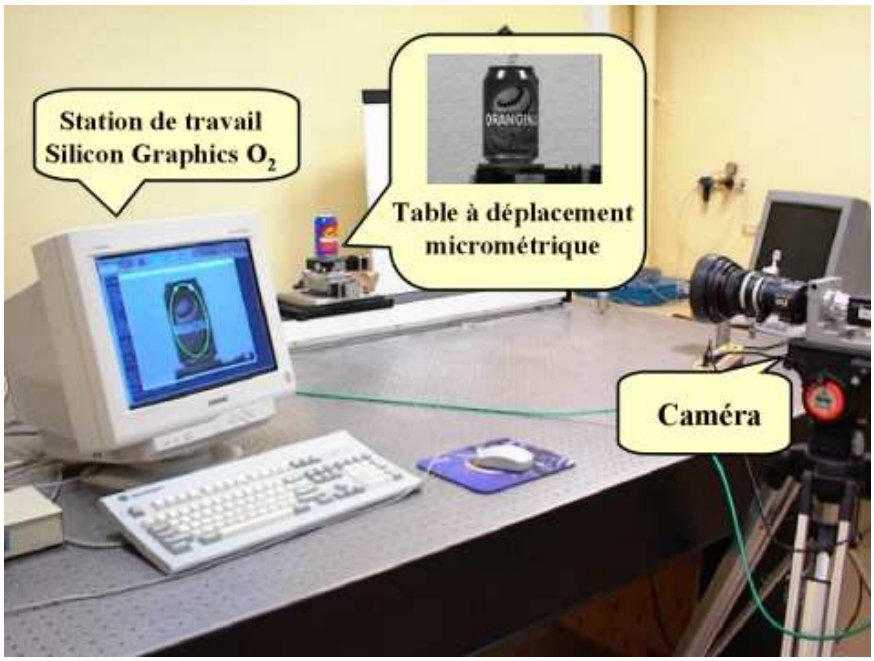


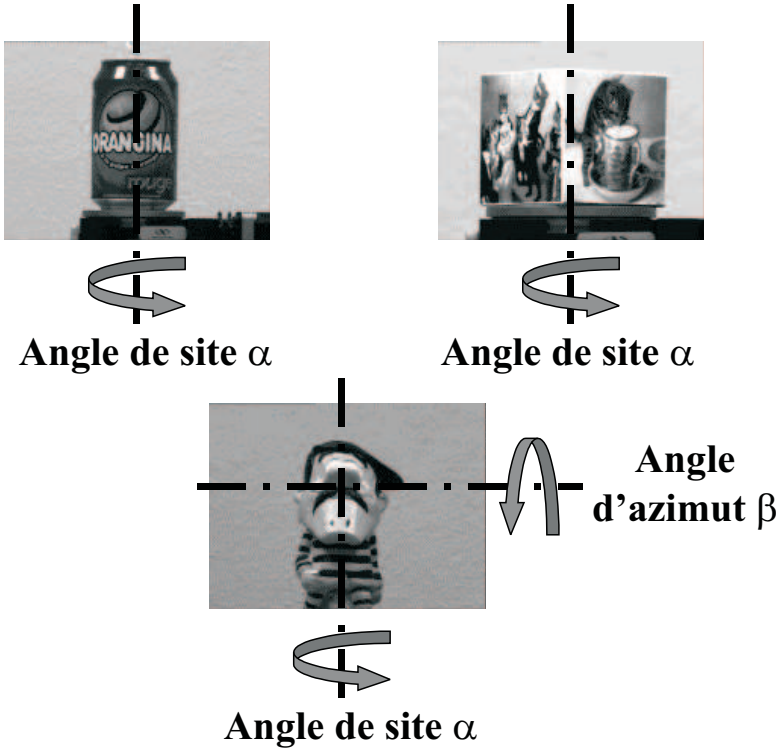
FIGURE 3.4 – présentation complète de la table à déplacement micrométrique.

Par exemple, pour les résultats présentés dans ce chapitre (figure 3.5), les *vues de référence* de la canette de soda et du cube sont acquises tous les 10 degrés en site sur 360 degrés et les vues intermédiaires tous les degrés (36 *vues de référence* pour un total de 361 images dans la base). Les *vues de référence* de la figurine sont acquises aussi tous les 10 degrés en site et azimut sur une portion de sphère (α = -40 à +40 degrés et β = -30 à +30 degrés), les vues

intermédiaires tous les 2 degrés (63 *vues de référence* pour un total de 1911 images). Ces objets ont été choisis pour montrer l'efficacité de notre méthode de suivi 3D appliquée à des objets de formes et de textures plutôt complexes et variées.



(a) phase d'acquisition.



(b) exemple de trois objets modélisés.

FIGURE 3.5 – acquisition d'une collection d'images 2D pour la modélisation d'objets.

La disposition des différentes *vues de référence* dans la collection d'images dépend fortement de la forme et du volume de l'objet entraînant une variation plus ou moins importante de l'aspect du motif courant sans pour autant perdre des points caractéristiques du motif de référence suivi. Notons ici la difficulté du choix de ces *vues de référence* qui devrait faire l'objet d'une étude complémentaire afin de les choisir automatiquement. Nous supposons également que nous sommes capables d'assurer le suivi 3D d'un motif de référence à l'un de ses plus proches voisins dans la collection d'images.

3.2.2 Représentation de l'apparence d'un motif

Nous souhaitons représenter le motif à suivre par un *vecteur de forme* (vecteur de niveaux de gris de dimension N où N est le nombre de points échantillonnés) et que cette représentation soit indépendante de la position, de l'orientation et de l'échelle du motif dans l'image. Pour cela, nous proposons d'échantillonner le motif à l'intérieur d'une zone elliptique (figure 3.6).

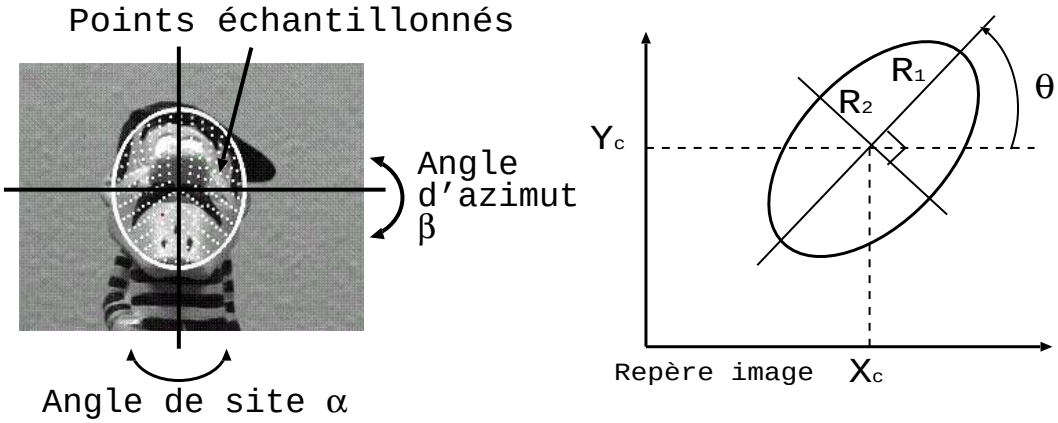


FIGURE 3.6 – échantillonnage du motif à l'intérieur d'une ellipse.

Les points où sont prélevés les niveaux de gris (points blancs sur la figure 3.6) sont répartis sur un ensemble d'ellipses concentriques échantillonnées de la plus petite à la plus grande. L'ensemble des valeurs échantillonnées (niveaux de gris toujours numérotés dans le même ordre) est alors stocké dans le *vecteur de forme*. Pour notre application, le *vecteur de forme* comprend 373 points échantillonnés sur 15 ellipses.

Ainsi quelle que soit la position, l'orientation et la taille du motif, sa représentation vectorielle après échantillonnage sera sensiblement la même, puisque les valeurs de niveaux de gris enregistrées sont positionnées dans un repère lié à l'ellipse et par conséquent au motif. Nous réalisons donc à la fois un échantillonnage local d'une région d'image et global du motif. Le

nombre de points échantillonnés dépend plus particulièrement de la méthode de sélection envisagée et de l'apparence de l'objet que l'on désire suivre.

La position et la forme de l'ellipse sont définies par un vecteur à cinq paramètres correspondant à la position du centre (X_c, Y_c), l'orientation (θ) et les longueurs du grand et du petit axe (R_1, R_2). Par la suite, nous poserons $R_2 = k * R_1$ où k est un ratio connu et fixé lors de la phase d'apprentissage pour avoir un seul facteur d'échelle. La représentation géométrique de ce vecteur est rappelée figure 3.6. Le choix de prendre une ellipse comme zone d'échantillonnage du motif suivi sera justifié dans la section suivante.

De plus, pour garantir une certaine insensibilité aux changements de conditions d'éclairement de la scène, le *vecteur de forme*, une fois échantillonné, est alors centré et normé. Ceci permet de compenser des variations affines de la luminance entre l'image de référence et l'image courante. La figure 3.7 illustre la robustesse de notre algorithme de suivi pour différentes valeurs d'ouverture et de fermeture du diaphragme de la caméra.

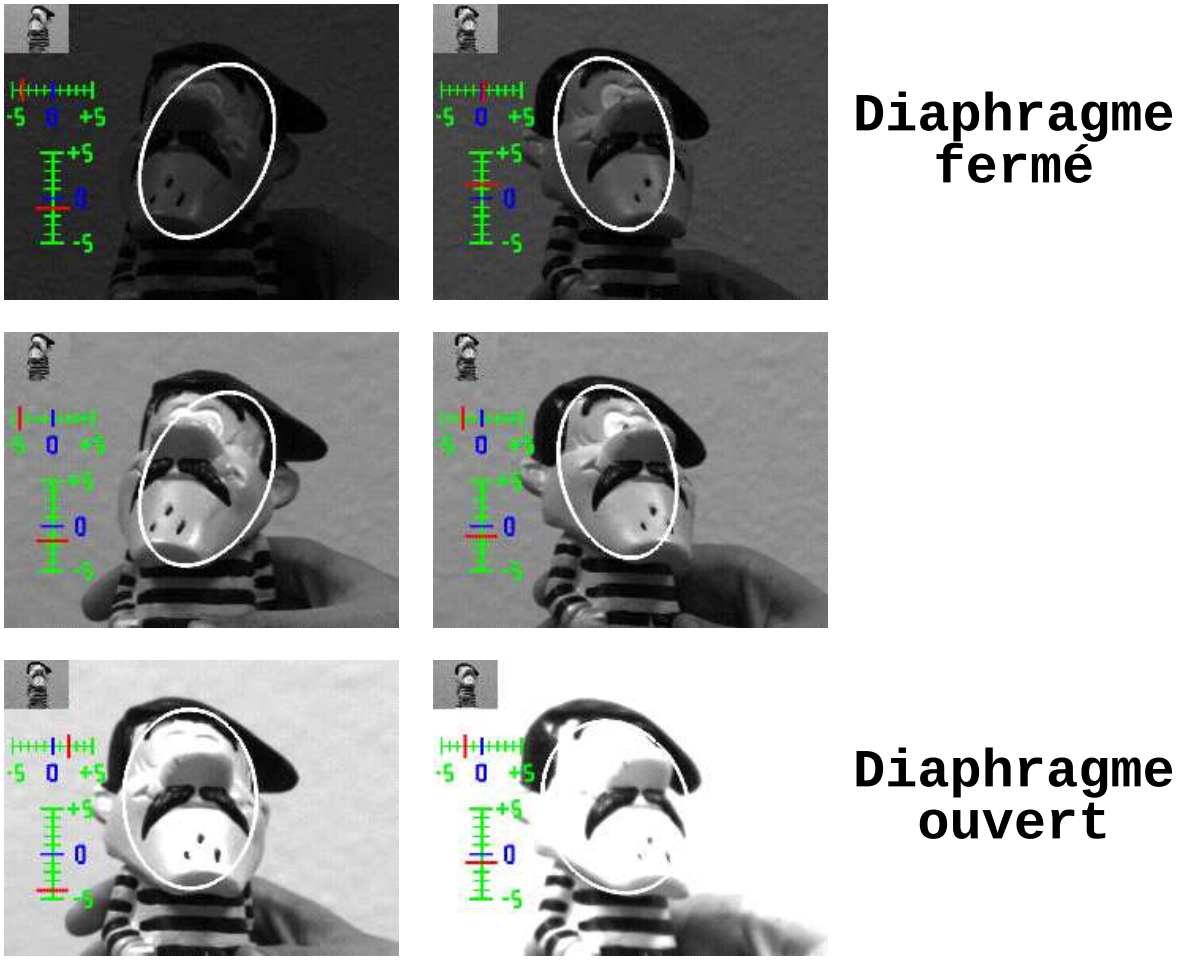


FIGURE 3.7 – robustesse de l'algorithme de suivi aux variations de luminance.

En effet, les conditions d'illumination peuvent avoir un effet sur l'apparence d'un objet. Plusieurs méthodes de reconnaissance d'objets, dans le passé, ont ignoré les variations d'éclairement ou utilisé des procédures simples de normalisation ou cherché certaines caractéristiques invariantes dans l'image comme les contours. Adini *et al.* [3] ont démontré pour la reconnaissance de visages que les variations d'illumination entraînent des variations plus importantes dans l'image que différents sujets vus sous les mêmes conditions d'illumination. Par conséquent, la reconnaissance d'objets ne peut pas donner des résultats fiables sans prendre en compte les variations d'éclairement. Farid et Adelson [51] ont démontré que les effets de la réflexion et de l'illumination peuvent être séparés par une analyse en composantes indépendantes. Toutefois, il n'est pas possible d'utiliser cette méthode pour la reconnaissance d'objets. Dans un récent papier, Chen *et al.* [19] ont montré que même pour des objets avec des surfaces Lambertiennes, il n'existe pas de fonctions discriminantes qui sont invariantes à l'illumination pour des images d'objets. Cependant, dans cet article, il a été démontré que pour une image de gradient, une mesure insensible aux variations d'éclairement peut être développée et utilisée pour une reconnaissance probabiliste d'objets. Dans un esprit similaire, Jacobs *et al.* [64] ont développé une mesure à partir d'un ratio de deux images pour comparer des images sous différentes variations d'illumination. Cette mesure peut être alors interprétée comme une simple comparaison entre des images de contours. Finalement, pour l'ensemble de ces études, nous pouvons conclure que les contours et les gradients sont des mesures usuelles pour gérer les variations d'éclairement. Cependant, dans les méthodes citées précédemment, les auteurs ne prennent pas en compte les poses multiples.

3.3 Suivi 3D d'un objet

Avant de développer l'aspect théorique du suivi 3D d'objets, nous adoptons les notations suivantes en référence au chapitre 2 qui traite du suivi d'un motif visuel 2D :

- $VI_{ref} = I_0(\mathbf{f}(\mathcal{R}; \mu_0^*))$ est le vecteur de niveaux de gris correspondant à l'échantillonnage du motif de référence que l'on souhaite suivre dans la zone d'intérêt.
- VI_c correspond soit à $\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0'))$ durant la phase d'apprentissage hors ligne (position perturbée autour de μ_0^*), soit à $\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu'))$ (position prédite) ou $\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu))$ (position corrigée ou réelle) durant la phase de suivi en ligne. En fait, VI_c est le vecteur de niveaux de gris correspondant à l'échantillonnage du motif courant dans la zone d'intérêt.
- $\mu_p = \mu'$ est le vecteur de paramètres prédit, $\mu_r = \mu$ le vecteur de paramètres correspondant à la position réelle du motif dans l'image et $\Delta\mu = \delta\mu$ la différence de ces deux vecteurs.

– $\Delta VI_p = VI_{ref} - VI_c = \delta i$ est la différence de niveaux de gris entre les deux *vecteurs de forme* pour la position prédite de l'objet dans l'image.
– $A = A_h$ est la *matrice d'interaction* apprise lors de la phase d'apprentissage hors ligne en utilisant la méthode d'approximation par hyperplans ($\delta\mu = A_h\delta i$ devient $\Delta\mu = A\Delta VI_p$).

Tout d'abord, nous devons être capable de paramétrer les mouvements possibles de l'objet pour le suivre dans l'image.

3.3.1 Paramétrisation des mouvements possibles d'un objet dans l'image

Les paramètres estimés lors du suivi doivent traiter tous les mouvements possibles de l'objet devant la caméra à savoir trois translations (T_x, T_y, T_z) et trois rotations (R_x, R_y, R_z) soit six degrés de liberté (représentation Eulérienne). Nous pouvons les classer en deux catégories :

1. ceux qui provoquent un *mouvement fronto parallèle* de l'objet par rapport au plan de l'image (un tel mouvement ne modifie pas notablement l'aspect du motif suivi). Ils sont au nombre de quatre :
   - T_x, T_y : translations axiales en x et y équivalentes aux coordonnées du centre de l'ellipse (X_c, Y_c),
   - T_z : translation axiale en z ou changement d'échelle défini par les rayons de l'ellipse $(R_1, R_2 = k * R_1)$,
   - R_z : rotation autour de l'axe z équivalente à l'orientation de l'ellipse (θ).

2. ceux qui provoquent des changements de l'apparence du motif suivi. La prise en compte de la modification d'aspect du motif se fait à l'aide de deux paramètres :
   - R_x : rotation autour de l'axe x équivalente à une variation de l'angle d'azimut β du motif courant dans l'image par rapport au motif de référence suivi,
   -R_y : rotation autour de l'axe y équivalente à une variation de l'angle de site α du motif courant dans l'image par rapport au motif de référence suivi.

Ceci est illustré par la figure 3.8 où le repère $\mathcal{R}$ correspond au repère caméra translaté vers le barycentre de l'objet (l'axe des x étant parallèle aux lignes de l'image, l'axe des y parallèle aux colonnes de l'image, l'axe des z parallèle à l'axe optique).

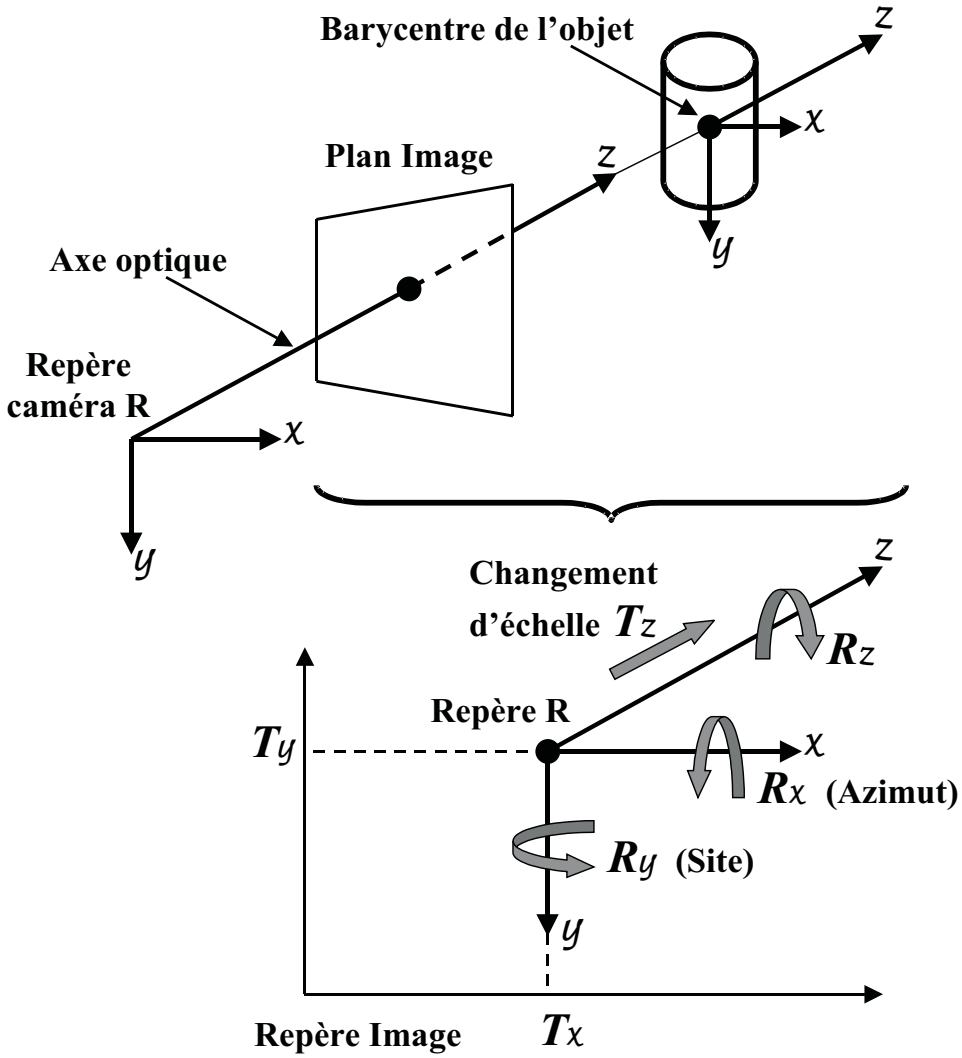


FIGURE 3.8 – mouvements possibles d'un objet dans l'image.

La combinaison de ces six paramètres ($X_c, Y_c, R_1, \theta, \alpha$ et β) nous permet de suivre un objet volumique dans une image tout en gérant ses variations d'aspect ($R_2 = k * R_1$ par définition).

3.3.2 Interprétation géométrique du suivi 3D

Nous venons de voir que le motif que l'on désire suivre est inscrit dans une ellipse dont la forme et la position dans l'image sont données par le vecteur de paramètres μ de dimension p (ici $p = 4$) avec $\mu = (X_c, Y_c, R_1, \theta)^t$ et $R_2 = k * R_1$.

Les équations déduites du chapitre 2 modélisant le suivi d'un motif visuel 2D dans une image peuvent s'écrire sous la forme :

$$\begin{cases} \mu_r = \mu_p + \Delta\mu \\ \Delta\mu = A\Delta VI_p = A(VI_{ref} - VI_c) \end{cases} \tag{3.1}$$

où :

- μ_p est le vecteur de paramètres prédit, μ_r le vecteur de paramètres correspondant à la position réelle du motif dans l'image, et $\Delta\mu$ la différence de ces deux vecteurs (c'est à dire la correction à apporter à la prédiction pour obtenir la position réelle du motif).
- VI_c est le *vecteur de forme courant* obtenu par échantillonnage du motif visuel à l'intérieur de l'ellipse prédite, VI_{ref} le *vecteur de forme* du motif de référence à suivre et ΔVI_p la différence de ces deux vecteurs de niveaux de gris.
- A est une matrice dite *"d'interaction"* $(p * N)$ correspondant au calcul d'une relation linéaire entre un ensemble de différences de niveaux de gris ΔVI_p et une correction $\Delta\mu$ des paramètres du vecteur μ lors d'une phase d'apprentissage hors ligne.

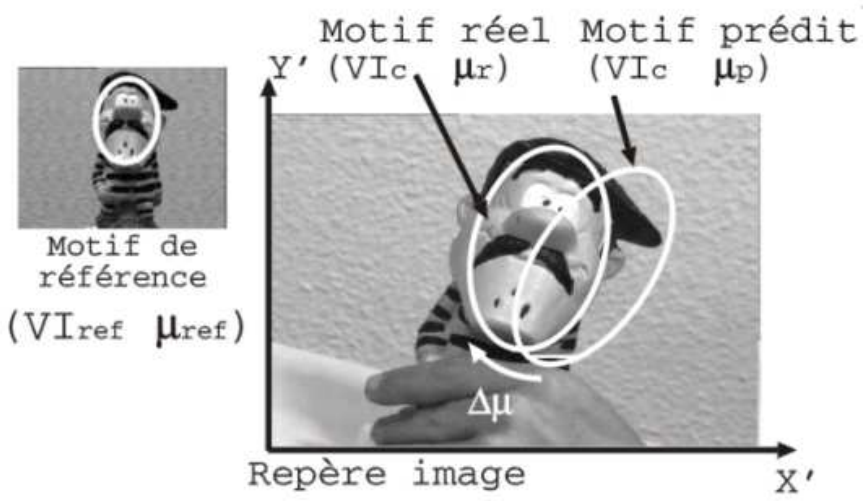


FIGURE 3.9 – principe du suivi d'un motif de référence.

Ce système d'équations (3.1) nous permet donc de repositionner le problème du suivi comme la détermination d'un vecteur d'offset $\Delta\mu$ (écrit sous une forme matricielle), en supposant que les variations de position de l'objet dans l'image correspondent aux variations des paramètres d'une transformation géométrique f (figure 3.9). Dans notre cas particulier, nous utilisons une *transformation affine* où les paramètres de l'ellipse sont les paramètres de la transformation géométrique (figure 3.10).

Un point de coordonnées (x, y) dans le référentiel région ou ellipse a pour coordonnées (x', y') dans le référentiel image par la transformation géométrique $f(\mu)$ tel que :

$$\begin{cases} x' = R_1 \cos(\theta)x - kR_1 \sin(\theta)y + X_c \\ y' = R_1 \sin(\theta)x + kR_1 \cos(\theta)y + Y_c \end{cases} \tag{3.2}$$

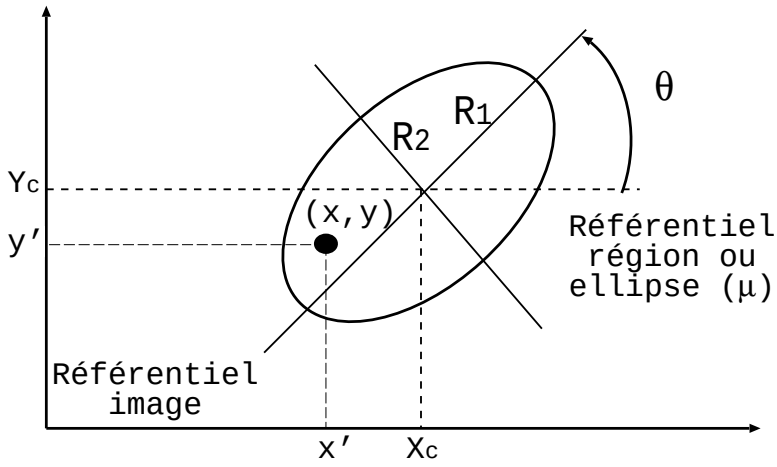


FIGURE 3.10 – interprétation géométrique des paramètres de l'ellipse dans le suivi 3D.

L'un des avantages de ce suivi est que l'on peut appliquer plusieurs types de transformation (voir chapitre 2 paragraphe *Modèles linéaires de mouvement*). Il faut simplement se définir une stratégie d'échantillonnage du motif tout en tenant compte de l'aspect et du volume de l'objet.

Nous avons vu également que les variations de l'aspect du motif courant par rapport au motif de référence suivi pouvaient être caractérisées par deux paramètres angulaires en site et azimut (α et β). Notons le vecteur $\psi = (\alpha, \beta)^t$ de dimension q (ici $q = 2$) et ΔVI_r la différence entre le vecteur de référence du motif suivi VI_{ref} et le vecteur courant VI_c dans l'ellipse prédite après correction du *mouvement fronto parallèle* estimé (motif recalé). Comme précédemment, il est alors intéressant de savoir si l'on peut déterminer ψ en connaissant ΔVI_r. Si c'est le cas, cela signifie qu'en mesurant la différence ΔVI_r, on est capable :

1. de positionner en site et azimut le motif courant dans l'image par rapport au motif de référence suivi et de ses plus proches voisins dans la collection de *vues de référence*,
2. de pouvoir décider quand changer de motif de référence.

Ceci est illustré figure 3.11, lors du suivi de figurine, pour une valeur α en site de 2 degrés par rapport au motif de référence 0 et une valeur en azimut β égale à 10 degrés. Dans cet exemple simple, le changement en site du motif de référence est décidé lorsque l'angle $\alpha \geq |\pm 6|$ pour éviter ainsi, un basculement permanent entre deux motifs pour $\alpha = \pm 5$.

1. Si ($\alpha < 6$ et $\alpha > -6$) alors motif suivi = motif actuel 0,

2. Si ($\alpha \geq 6$) alors motif suivi = motif suivant 10,
3. Si ($\alpha \leq -6$) alors motif suivi = motif précédent -10.

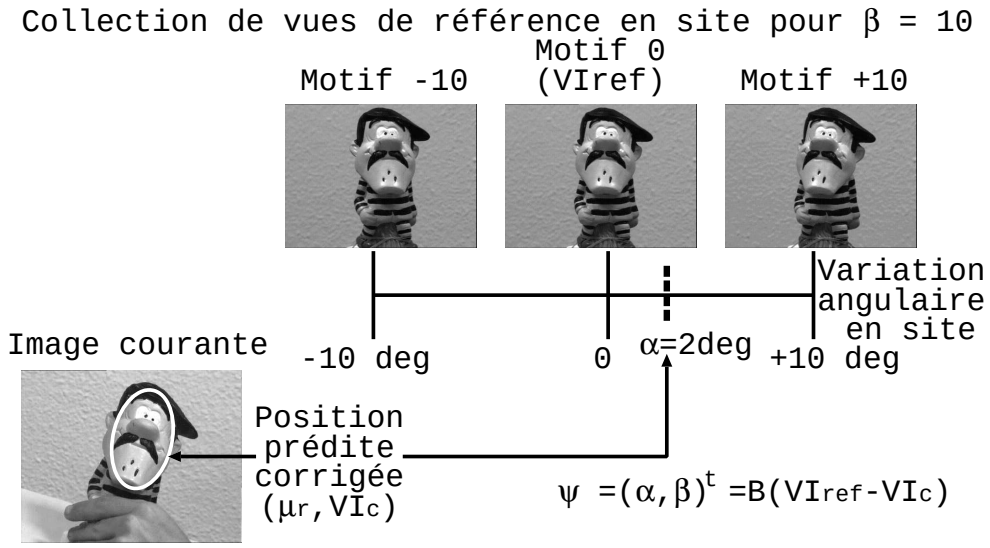


FIGURE 3.11 – positionnement du motif courant dans la collection d'images 2D suivant les variations d'aspect.

Ce calcul de ψ peut s'écrire sous la forme matricielle suivante :

$$\psi = B\Delta VI_r \tag{3.3}$$

où B est une matrice dite *"d'interaction"* $(q * N)$ correspondant au calcul d'une relation linéaire entre un ensemble de différences de niveaux de gris ΔVI_r et une estimation des paramètres angulaires du vecteur ψ par rapport au motif de référence suivi lors d'une phase d'apprentissage hors ligne.

Comme les objets 3D sont représentés par une collection d'images 2D, les *matrices d'interaction* A et B sont calculées pour chacune des *vues de référence*.

3.3.3 Estimation des matrices d'interaction pour un motif de référence donné

Le calcul des deux *matrices d'interaction* A et B se fait lors d'une phase d'apprentissage hors ligne. Une des originalités de la méthode de calcul présentée dans le chapitre 2 est que nous n'utilisons pas de matrices Jacobiennes de la *vue de référence* comme dans les travaux de Gregory D. Hager et Peter N. Belhumeur [60] ou Frank Dellaert et Robert Collins [39]. Nous estimons les matrices A et B par une minimisation *au sens des moindres carrés* en utilisant un

algorithme basé sur une décomposition en valeurs singulières (voir Annexe A sur la *résolution des systèmes d'équations linéaires*). Nous avons observé que dans ce cas, le domaine de convergence était beaucoup plus important [67] [68]. Par similitude avec la *commande référencée vision* (CRV), nous parlons ici de *matrice d'interaction*. Son estimation est une approximation linéaire correspondant à un développement limité du premier ordre d'une fonction [66]. Durant cette phase d'apprentissage, nous cherchons à minimiser un critère dans l'image (une différence de niveaux de gris). Nous n'avons donc jamais la certitude d'avoir la bonne estimation des paramètres 3D.

De plus, notre méthode par hyperplans [67] [68] ne permet pas d'estimer l'ensemble des paramètres 3D par une seule *matrice d'interaction*. L'explication est que nous pouvons obtenir, à partir de l'échantillonnage du motif visuel à l'intérieur d'une zone d'intérêt prédite, le même *vecteur de forme courant* VI_c pour différents mouvements possibles de l'objet devant la caméra (les *mouvements fronto parallèles* d'une part et les rotations en *site* et *azimut* provoquant un changement d'apparence d'autre part). Un problème d'ambiguïté (figure 3.12) se pose donc dans la reconnaissance du mouvement détecté (être capable de dissocier les deux catégories de mouvement présentées en début de section).

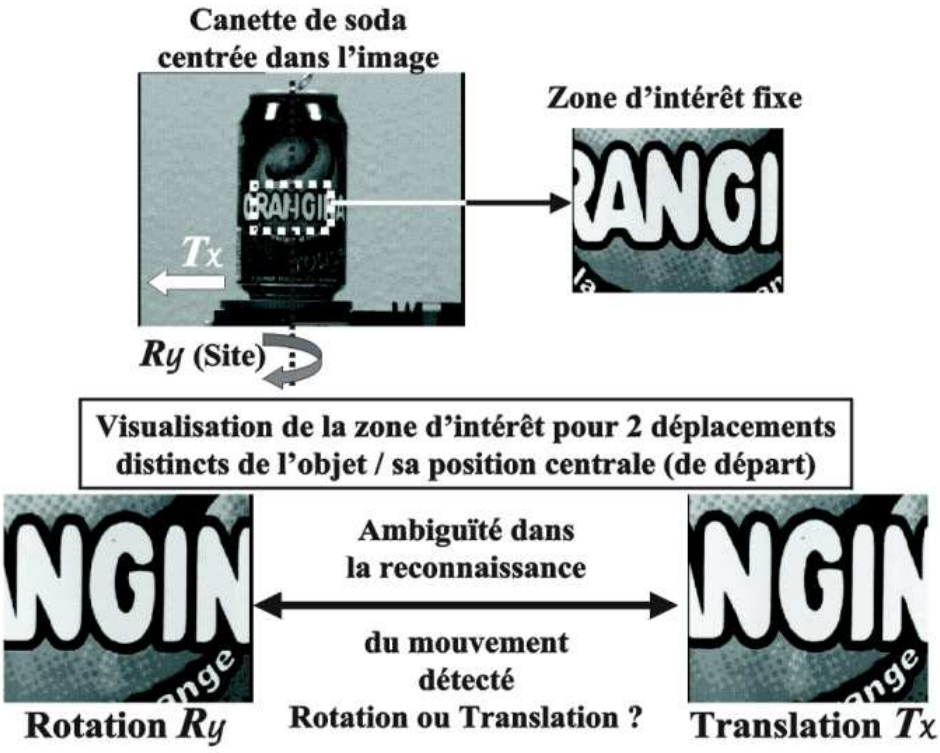


FIGURE 3.12 – ambiguïté dans la reconnaissance du mouvement détecté.

C'est pourquoi, nous avons décomposé le déplacement 3D d'un objet devant la caméra en plusieurs mouvements classés en deux catégories (avec ou sans modification de l'aspect du motif suivi).

Calcul de la matrice d'interaction A

Cette matrice permet la mise à jour des paramètres de l'ellipse ou de la *transformation affine* lors du suivi de l'objet dans une séquence d'images. Son estimation durant la phase d'apprentissage hors ligne est identique à celle présentée dans le chapitre 2. Mais ici, nous précisons les choix réalisés sur les différents paramètres de l'ellipse.

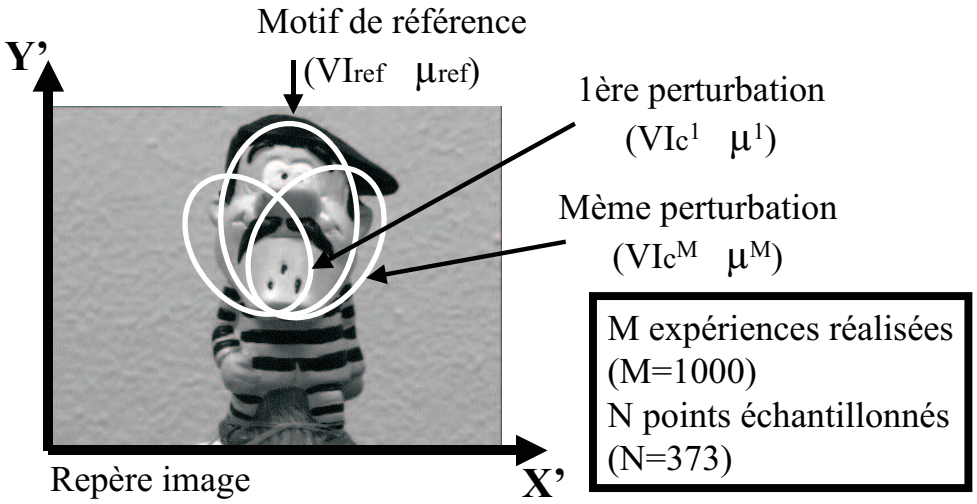


FIGURE 3.13 – perturbations des paramètres de l'ellipse pour l'estimation de la *matrice d'interaction* A.

Une ellipse est placée manuellement par l'utilisateur sur le motif de référence puis échantillonnée pour donner le *vecteur de forme de référence* VI_{ref} de dimension N ($N = 373$ points échantillonnés). Cette initialisation nous permet également de fixer le rapport k entre les deux rayons de l'ellipse ($k = R_2/R_1$). La position de l'ellipse est perturbée M fois aléatoirement autour de sa position de référence ($M = 1000$) tout en gardant le coefficient k constant (figure 3.13). Les amplitudes des variations des paramètres de l'ellipse sont de 15% de la longueur des axes pour la position du centre (rappelons que les variations de ce dernier sont réalisées le long des axes principaux de l'ellipse), de 15% sur la longueur des axes et de 15 degrés sur la rotation.

Pour chaque perturbation j, les variations des paramètres de la transformation $\Delta\mu^j = (\Delta X_c^j, \Delta Y_c^j, \Delta R_1^j, \Delta\theta^j)^t$ ainsi que le vecteur de différence $\Delta VI^j = (\Delta i_1^j, \Delta i_2^j, \ldots, \Delta i_N^j)^t$ entre le motif de référence VI_{ref} et le motif courant VI_c^j sont mémorisés. Il est alors possible d'es-

timer A si $M \geq N$. Cela revient donc à résoudre un système surdimensionné de M équations à N inconnues pour chacun des paramètres de la transformation soit quatre systèmes. En réalité, la résolution d'un seul système linéaire, ou plus exactement, le calcul d'une seule matrice pseudo-inverse est nécessaire. En notant la *matrice d'interaction* A sous la forme $A = (AX_c, AY_c, AR_1, A\theta)^t$, nous obtenons la ligne $A\theta$ de la *matrice d'interaction* relative à l'orientation de l'ellipse à l'aide du système linéaire suivant :

$$\begin{pmatrix} \Delta i_1^1 & \Delta i_2^1 & \dots & \Delta i_N^1 \\ \Delta i_1^2 & \Delta i_2^2 & \dots & \Delta i_N^2 \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \Delta i_1^M & \Delta i_2^M & \dots & \Delta i_N^M \end{pmatrix} \begin{pmatrix} A\theta_1 \\ A\theta_2 \\ \vdots \\ A\theta_N \end{pmatrix} = \begin{pmatrix} \Delta\theta^1 \\ \Delta\theta^2 \\ \vdots \\ \vdots \\ \Delta\theta^M \end{pmatrix} \tag{3.4}$$

Pouvant se mettre sous la forme matricielle suivante :

$$M_{\Delta VI} * A\theta = \Delta\theta \tag{3.5}$$

La solution est alors obtenue par :

$$A\theta = (M_{\Delta VI}^t M_{\Delta VI})^{-1} M_{\Delta VI}^t \Delta\theta = M_{\Delta VI}^+ \Delta\theta \tag{3.6}$$

La matrice $M_{\Delta VI}^+$ est la matrice pseudo-inverse de la matrice $M_{\Delta VI}$. Le calcul des trois autres lignes de la *matrice d'interaction* A utilise le produit de la même matrice avec des vecteurs de perturbations différents $(\Delta X_c, \Delta Y_c, \Delta R_1)$:

$$\begin{cases} AX_c = M_{\Delta VI}^+ \Delta X_c \\ AY_c = M_{\Delta VI}^+ \Delta Y_c \\ AR_1 = M_{\Delta VI}^+ \Delta R_1 \end{cases} \tag{3.7}$$

En pratique, durant la phase de suivi en ligne, nous utilisons une stratégie à deux niveaux d'approximation appliqués successivement pour assurer une correction dite *"grossière à fine"* de la position réelle du motif dans l'image. Pour cela, pendant la phase d'apprentissage, deux *matrices d'interaction* A_1 et A_2 sont apprises distinctement pour des niveaux différents de perturbation et ceci pour chacun des motifs de référence modélisant l'objet 3D. Les amplitudes de

ces perturbations sur les paramètres de l'ellipse ont été fixées respectivement à :

- 15 et 3% de la longueur des axes pour la position du centre (X_c, Y_c),
- 15 et 3% sur la longueur des axes (R_1 et $R_2 = k * R_1$),
- 15 et 3 degrés sur la rotation (θ).

Cette approximation *multiéchelle* permet d'augmenter d'une part les amplitudes tolérées sur les mouvements de l'objet suivi et d'autre part la précision du suivi. Ici, nous employons le terme *"multiéchelle"* puisque prendre une image et l'échantillonner pour des perturbations différentes revient à prendre plusieurs images du même objet pour des tailles différentes et des les échantillonner de la même façon. De plus, l'une des contraintes du *"multiéchelle"* est qu'il est nécessaire de filtrer l'image. En effet, une image sous-échantillonnée comportant des lignes blanches et noires doit donner une image grise et non blanche ou noire. Ceci implique donc un filtrage de toute l'image qui est une opération coûteuse en temps de calcul pour notre application. Mais le fait de réaliser un grand nombre de tirages aléatoires lors de la phase d'apprentissage et de calculer les *matrices d'interaction* par une méthode *au sens des moindres carrés* filtrent notre image.

Résultats expérimentaux sur l'estimation de la matrice A

La caractérisation des performances de la *matrice d'interaction* A, permettant d'observer comment et dans quelles limites cette matrice permet de revenir sur le motif sélectionné lorsque l'on écarte la zone d'intérêt de la position de référence pour des *mouvements fronto parallèles* (translations et rotation planaires, changement d'échelle), a été présentée et commentée au chapitre 2 dans la section *Expérimentations et Résultats*.

Calcul de la matrice d'interaction B

Cette matrice permet la mise à jour des paramètres angulaires pour traiter les changements d'aspect de l'objet lors de son suivi dans une séquence d'images.

Comme précédemment, il s'agit de déterminer puis de multiplier successivement une matrice pseudo-inverse $M^+_{\Delta VI}$ par les variations en site $\Delta\alpha$ et en azimut $\Delta\beta$ pour calculer les lignes de la *matrice d'interaction* $B = (B_\alpha, B_\beta)^t$:

$$B_\alpha = M^+_{\Delta VI}\Delta\alpha \quad \text{et} \quad B_\beta = M^+_{\Delta VI}\Delta\beta \tag{3.8}$$

Pour cela, il faut utiliser les n vues intermédiaires entourant le motif de référence dont le nombre varie selon les règles de décision prises pour changer de motif. Dans le cas de la figurine, illustrée par la figure 3.14, nous avons décidé de changer de motif de référence courant quand l'une des deux valeurs angulaires α ou β est supérieure en valeur absolue à 5 degrés.

Pour avoir un calcul fiable des valeurs angulaires, il est nécessaire d'acquérir des images intermédiaires entre chaque motif de référence et nous étendons la zone de calcul à des variations de ± 8 degrés par pas de 2 degrés autour d'une *vue de référence* soit un total de n vues de travail ($n = 81$ puisque nous avons neuf images différentes en site par azimut et neuf variations d'azimut).

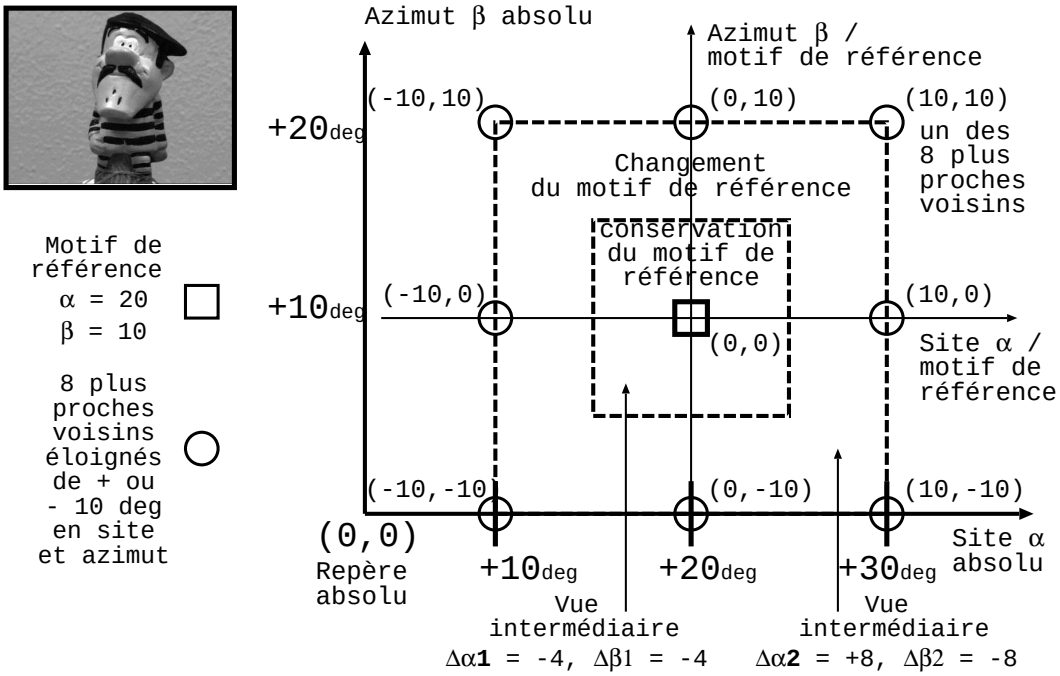


FIGURE 3.14 – sélection des vues nécessaires au calcul de la *matrice d'interaction B*.

Pour chaque vue intermédiaire de variations en site $\Delta\alpha_n$ et d'azimut $\Delta\beta_n$, nous réalisons M perturbations des paramètres de l'ellipse définie sur le motif de référence ($M = 20$), puis calculons les corrections à apporter aux paramètres de la transformation géométrique ou de l'ellipse en multipliant la *matrice d'interaction A* de la *vue de référence* par la différence entre le vecteur de référence VI_{ref} et le motif courant perturbé VI_c échantillonné dans l'image intermédiaire. L'ellipse étant corrigée et repositionnée sur le motif de référence à suivre dans l'image intermédiaire, la nouvelle mesure de différence de niveaux de gris entre le vecteur de référence VI_{ref} et le motif courant corrigé VI_c ainsi que les variations en site $\Delta\alpha_n$ et en azimut $\Delta\beta_n$ sont mémorisées pour calculer la *matrice d'interaction B* (figure 3.15). Cette estimation

de B n'est possible que si $n * M \geq N$.

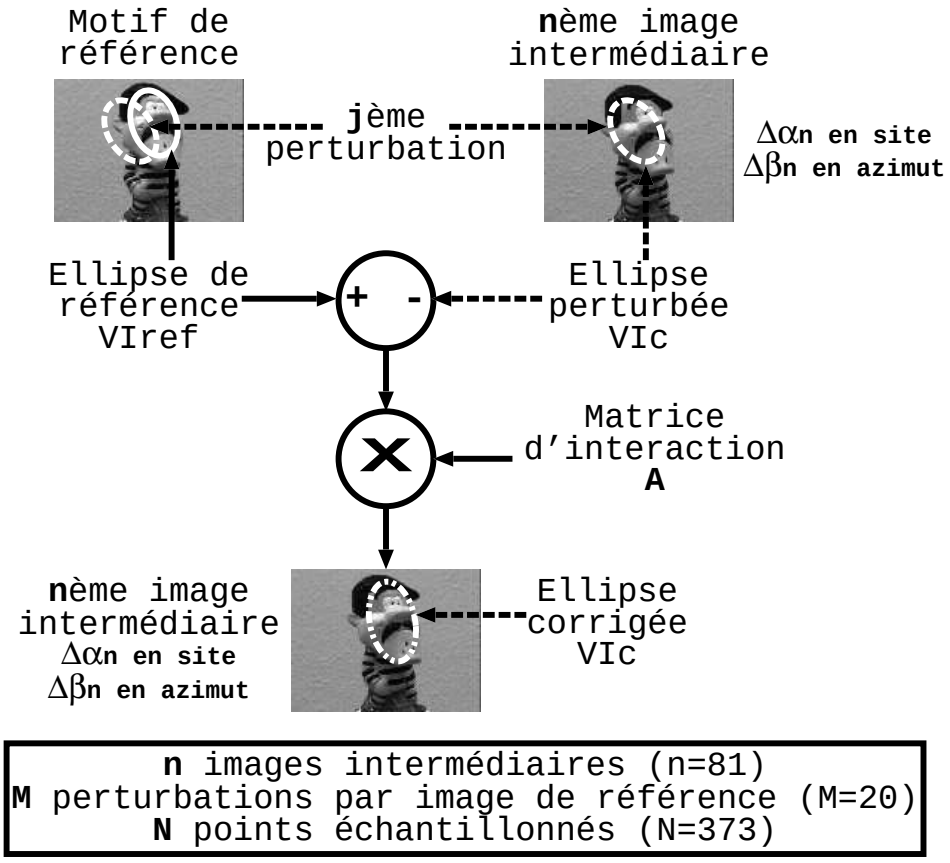


FIGURE 3.15 – perturbations et corrections des paramètres de l'ellipse pour l'estimation de la *matrice d'interaction* B.

Résultats expérimentaux sur l'estimation de la matrice B

Nous allons ici chercher à caractériser les performances de la *matrice d'interaction* B en observant ses limites. Cette matrice permet d'estimer les variations d'aspect du motif courant dans l'image par rapport au motif de référence actuellement suivi. Ces calculs sont effectués pour différentes valeurs d'angle de site α comprises dans la plage de mesures utilisée durant la phase d'apprentissage hors ligne de la *matrice d'interaction* B considérée (variation angulaire autorisée de 16 degrés pour un angle de site α variant de -8 à +8 degrés autour du motif de référence).

Nous présentons ici des résultats pour des variations en site uniquement puisque nous observons les mêmes résultats pour des variations en azimut. Cela s'explique, du fait que les deux angles en site et azimut (α et β) sont estimés à l'aide de la même *matrice d'interaction* B. De plus, nous utilisons deux objets de forme et de réflectance totalement différentes (une canette de soda et une figurine) pour voir l'incidence sur les mesures des variations d'aspect.

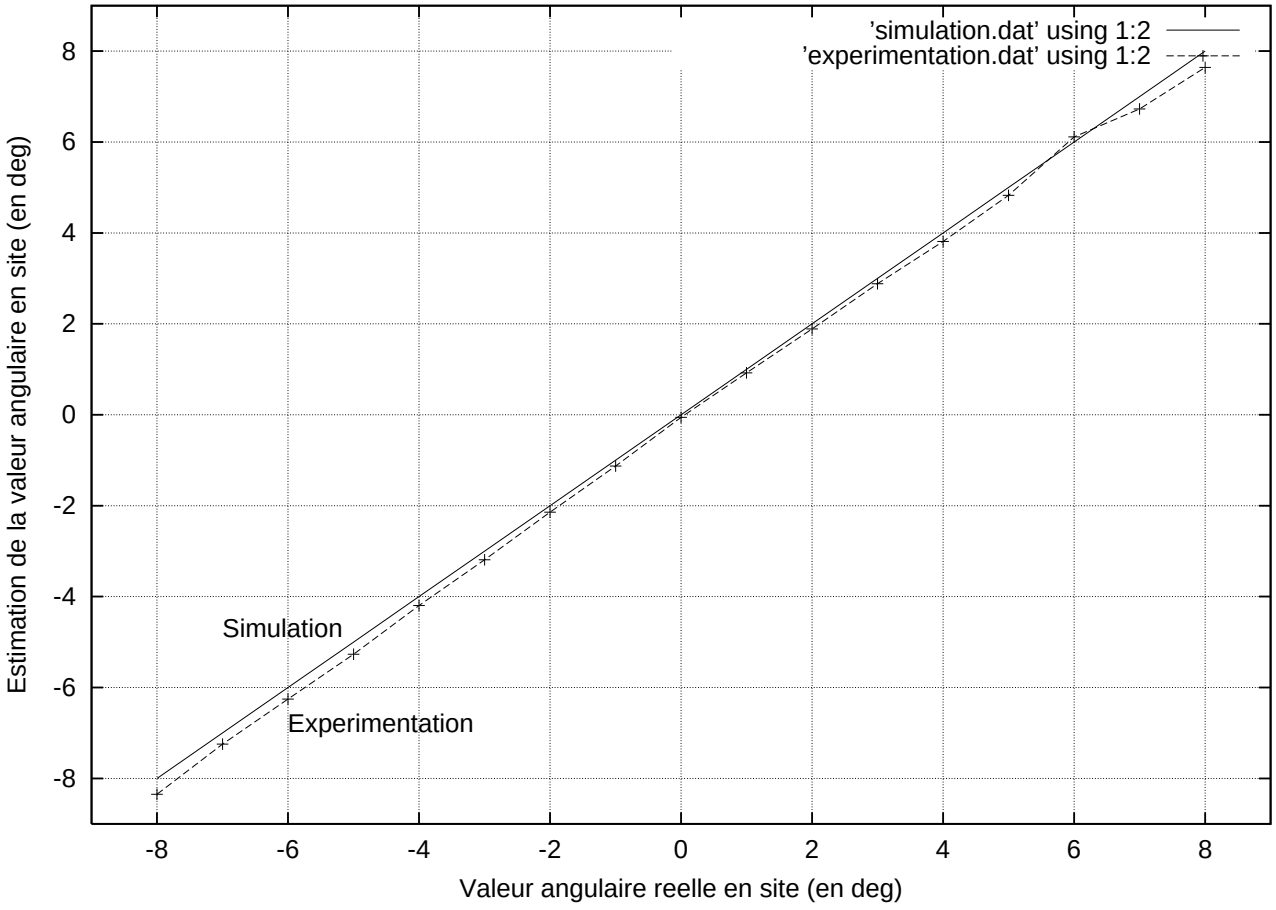
Les courbes présentées figures 3.16 et 3.17 indiquent l'estimation de l'angle de site $\alpha_{estimé} = B\Delta VI_r$ en fonction de la variation angulaire réelle en site $\alpha_{réel}$ pour un test en simulation et une expérimentation.

Dans l'exemple de la **simulation**, nous considérons l'ensemble des n vues nécessaires au calcul de la *matrice d'interaction* B lors de la phase d'apprentissage hors ligne pour le motif de référence testé. Pour chacune de ces n vues, nous corrigeons la position prédite de l'ellipse définie dans l'image de référence ($\mu_r = \mu_p + A\Delta VI_p$) et calculons la valeur estimée de l'angle de site $\alpha_{estimé} = B\Delta VI_r$. La courbe obtenue est une droite (de pente égale à 1) puisque nous utilisons les mêmes images pour estimer la valeur $\alpha_{estimé}$ que pour apprendre la *matrice d'interaction* B. En effet, chacune de ces images représente bien une variation réelle en site $\alpha_{réel}$ de l'aspect du motif suivi par rapport à sa *vue de référence* dont nous connaissons la valeur (image acquise tous les degrés). Par conséquent, nous avons $\alpha_{estimé} = \alpha_{réel}$ qui montre que l'apprentissage de la *matrice d'interaction* B à l'aide de l'approximation par hyperplans est correctement réalisé.

Dans le cas de l'**expérimentation**, nous positionnons au mieux l'objet dans l'image pour obtenir une valeur angulaire estimée en site $\alpha_{estimé}$ nulle pour le motif de référence testé, en exécutant l'algorithme de suivi et ceci, pour une position caméra/objet la plus proche possible de celle utilisée lors de la création de la base d'apprentissage. Ensuite, nous commandons le déplacement en site de la table micrométrique pour faire varier la valeur $\alpha_{réel}$ et enregistrons la valeur angulaire correspondante en site $\alpha_{estimé}$ obtenue à l'aide de l'algorithme de suivi ($\alpha_{réel}$ variant de -8 à +8 degrés par pas de 1 degré) et ceci, pour une plage de mesures correspondant à celle utilisée pour le calcul de la matrice d'interaction B. La courbe obtenue sur les figures 3.16 et 3.17 nous montre que l'estimation de l'angle de site $\alpha_{estimé}$ est une bonne approximation de la valeur angulaire réelle $\alpha_{réel}$ sur toute la plage de mesures. L'erreur angulaire $err = |\alpha_{estimé} - \alpha_{réel}|$ est inférieure à 0.2 degré pour $\alpha_{réel}$ variant entre -6 et +6 degrés (soit une variation angulaire totale de 12 degrés) et inférieure à 0.48 degré jusqu'aux limites de la plage de mesures. Les courbes théorique (ou idéale) et expérimentale sont donc pratiquement confondues et montrent que nous sommes capables d'estimer correctement les variations d'aspect du motif courant dans l'image et décider quand changer de motif de référence pour assurer le suivi 3D. Notons que l'augmentation de l'erreur angulaire err aux limites de la plage de mesures peut s'expliquer par le fait que les conditions réelles d'éclairement ont varié par rapport à la base

d'apprentissage et sont accentuées pour les variations d'aspect les plus importantes du motif actuellement suivi.

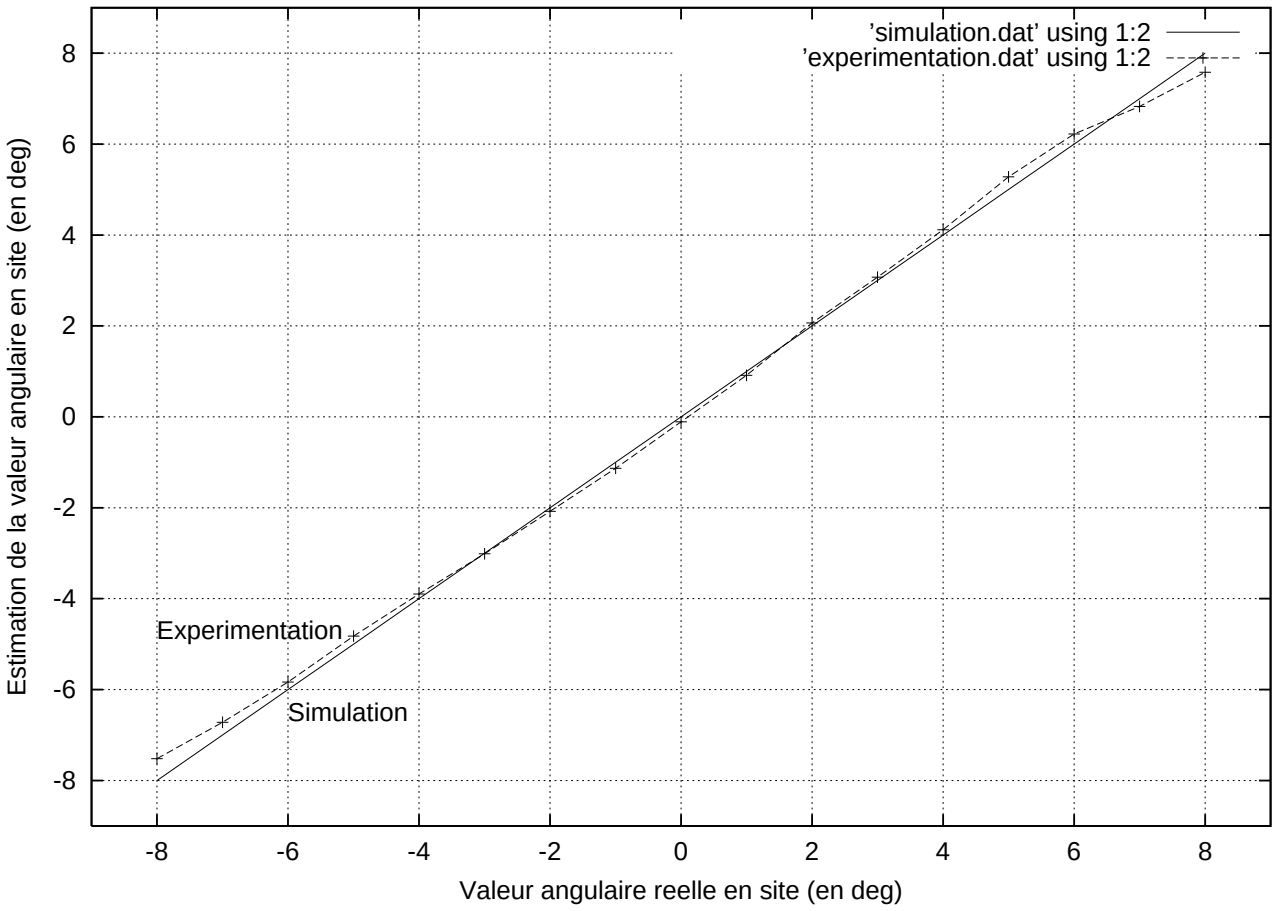
(a) images de la base d'apprentissage utilisées pour l'estimation de la matrice B.



(b) valeur angulaire estimée $\alpha_{estimé}$ en fonction de la valeur angulaire réelle $\alpha_{réel}$ pour une variation en site uniquement (en degrés).

FIGURE 3.16 – résultats sur l'estimation de l'angle de site α à partir de la *matrice d'interaction* B pour un objet de forme cylindrique (canette de soda).

(a) images de la base d'apprentissage utilisées pour l'estimation de la matrice B.



(b) valeur angulaire estimée $\alpha_{estimé}$ en fonction de la valeur angulaire réelle $\alpha_{réel}$ pour une variation en site uniquement (en degrés).

FIGURE 3.17 – résultats sur l'estimation de l'angle de site α à partir de la *matrice d'interaction* B pour un objet de forme quelconque (figurine).

3.3.4 Gestion du passage d'un motif de référence à l'autre

Nous avons vu précédemment que le calcul de la *matrice d'interaction* B dépend des règles de décision de passage d'un motif de référence à un autre, qui dépendent elles même de la

méthode d'acquisition ou de disposition des *vues de référence* les unes par rapport aux autres. En fonction de l'objet à modéliser, chacun peut se définir des règles de décision propres à son application.

Toutefois, un problème doit être pris en compte lors du changement du motif suivi : c'est la correction des paramètres de la transformation du nouveau motif en fonction de ceux de l'ancien motif. Ceci est à effectuer lorsque les paramètres de la transformation géométrique ou de l'ellipse sont différents entre les motifs de référence modélisant l'objet 3D (figure 3.18).

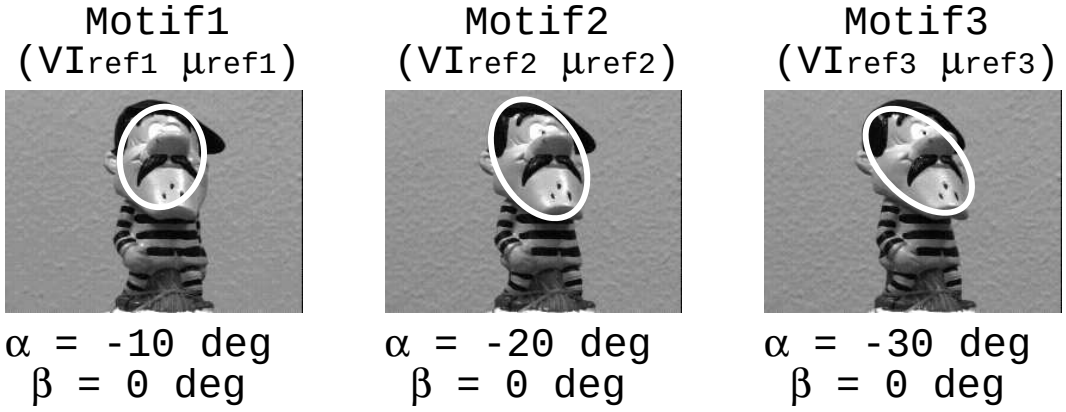


FIGURE 3.18 – exemple de 3 motifs de référence consécutifs pour des paramètres de transformation géométrique différents.

Pendant cette étape intermédiaire du suivi en ligne (figure 3.19), il s'agit de déterminer, dans l'image courante, les paramètres μ'_{su} de l'ellipse du prochain motif de référence à suivre en fonction des paramètres corrigés μ'_{ac} de l'ellipse du motif courant (motif recalé) et ceux correspondant dans la base d'apprentissage (μ_{su} et μ_{ac}). Ces calculs sont maintenant développés en prenant les notations suivantes :

1. **base d'apprentissage**

   – $\mu_{ac} = (X_{ac}, Y_{ac}, R_{1ac}, \theta_{ac})^t$ pour le motif de référence actuellement suivi,

   – $\mu_{su} = (X_{su}, Y_{su}, R_{1su}, \theta_{su})^t$ pour le prochain motif de référence à suivre.

2. **changement de motif de référence lors de la phase de suivi en ligne**

   – $\mu'_{ac} = (X'_{ac}, Y'_{ac}, R'_{1ac}, \theta'_{ac})^t$ pour la position courante du motif recalé,

– $\mu'_{su} = (X'_{su}, Y'_{su}, R'_{1su}, \theta'_{su})^t$ pour la position à estimer.

Nous rappelons également que par définition $R_2 = k * R_1$ où le coefficient k est fixé lors de la phase d'apprentissage.

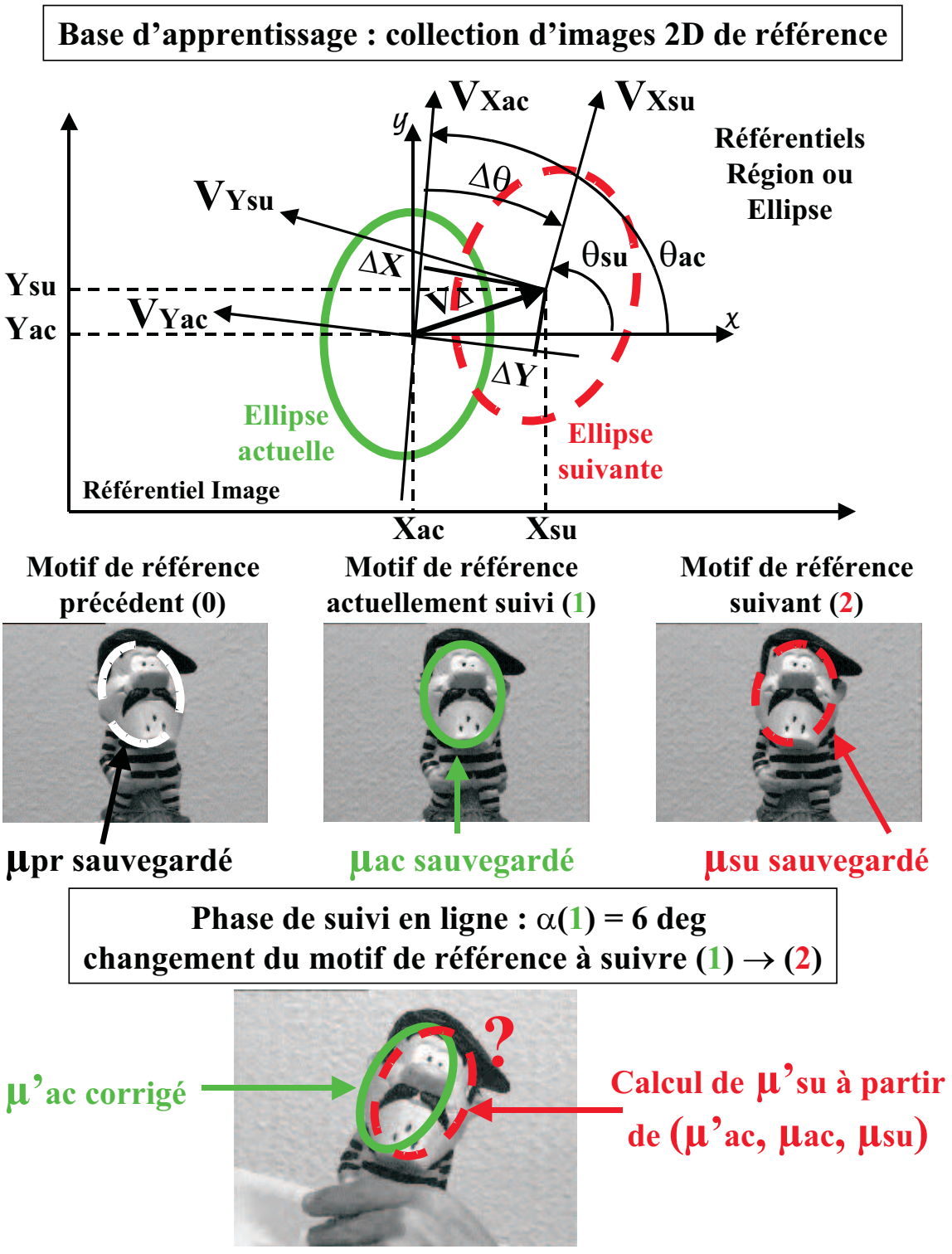


FIGURE 3.19 – correction intermédiaire des paramètres de l'ellipse lors du changement de motif de référence pendant le suivi (objet centré).

1. longueurs des axes de l'ellipse pour le changement d'échelle :

$$R'_{1su} = \frac{R_{1su}}{R_{1ac}} R'_{1ac} \quad et \quad R'_{2su} = k * R'_{1su} \tag{3.9}$$

2. orientation de l'ellipse :

$$\theta'_{su} = \theta'_{ac} + \Delta\theta = \theta'_{ac} + (\theta_{su} - \theta_{ac}) \tag{3.10}$$

3. coordonnées du centre de l'ellipse :

   Dans le référentiel (x, y) de la base d'apprentissage, les vecteurs V_{Xac}, V_{Yac} et V_Δ ont respectivement pour coordonnées $(\cos\theta_{ac}, \sin\theta_{ac})$, $(-\sin\theta_{ac}, \cos\theta_{ac})$ et $((X_{su} - X_{ac}), (Y_{su} - Y_{ac}))$. Les écarts relatifs ΔX et ΔY entre les coordonnées des centres des ellipses considérées sont estimés à partir des équations suivantes :

$$\begin{cases} \Delta X = V_\Delta \cdot V_{Xac} = (X_{su} - X_{ac})\cos\theta_{ac} + (Y_{su} - Y_{ac})\sin\theta_{ac} \\ \Delta Y = V_\Delta \cdot V_{Yac} = -(X_{su} - X_{ac})\sin\theta_{ac} + (Y_{su} - Y_{ac})\cos\theta_{ac} \end{cases} \tag{3.11}$$

   En notant $k_1 = \frac{R'_{1ac}}{R_{1ac}}$ et $k_2 = \frac{R'_{2ac}}{R_{2ac}}$ les facteurs d'échelle pour passer de la base d'apprentissage à la phase de suivi en ligne, les coordonnées de la nouvelle ellipse dans le référentiel de l'image courante sont obtenues par :

$$X'_{su} = X'_{ac} + k_1\Delta X \quad et \quad Y'_{su} = Y'_{ac} + k_2\Delta Y \tag{3.12}$$

Après cette étape intermédiaire, nous corrigeons les erreurs de position sur l'ellipse courante μ'_{su} en multipliant le vecteur de différence de niveaux de gris entre le nouveau motif de référence suivi et le motif courant par la *matrice d'interaction A* associée. Ainsi, nous recalons correctement le motif suivi et validons le changement de motif (voir l'algorithme de suivi 3D décrit dans la section *Expérimentations*).

En fait, ces calculs supplémentaires pour le changement de motif de référence ne sont valables que si l'objet 3D reste centré dans l'image pendant la phase d'acquisition de la collection d'images 2D. En effet, lors de la phase d'apprentissage, nous sauvegardons, pour chacune des *vues de référence*, le *vecteur de forme* associé ainsi que les paramètres de l'ellipse qui correspondent à des variations angulaires en site et azimut (α et β) nulles. Or, durant la phase de suivi, la correction des paramètres d'ellipse lors du changement de motif de référence s'opère pour des variations angulaires non nulles (figure 3.20).

En particulier, comme les écarts relatifs ΔX et ΔY entre les coordonnées des centres des ellipses considérées sont calculés à partir des paramètres d'ellipse de référence de la base d'apprentissage, cela engendre une erreur de position du centre de la nouvelle ellipse dans la phase de suivi. Cette erreur, restant compatible avec les variations apprises lors de la phase d'apprentissage de la *matrice d'interaction* A, est corrigée lors de l'itération suivante.

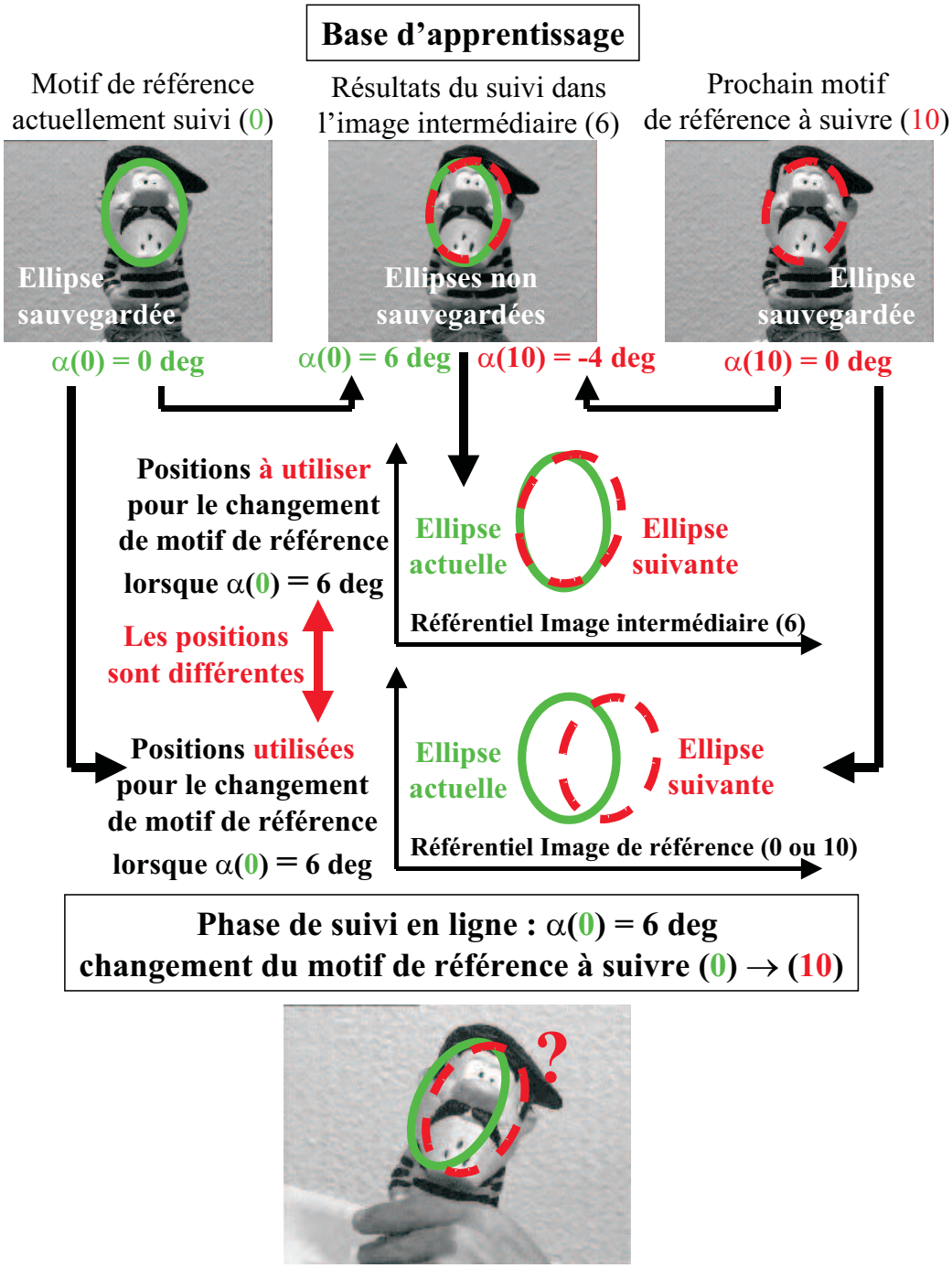


FIGURE 3.20 – erreurs sur les paramètres actualisés de l'ellipse lors du changement de motif de référence pendant le suivi (objet centré).

Lorsque l'objet 3D n'est pas centré dans l'image lors de l'acquisition de la collection d'images 2D, cette erreur de position est trop importante pour pouvoir être corrigée à l'itération suivante. Pour résoudre ce problème, il faut sauvegarder d'autres informations durant la phase d'apprentissage hors ligne. En particulier, pour chaque motif de référence, il faut repositionner et mémoriser les nouveaux paramètres de l'ellipse associée à l'aide de la *matrice d'interaction A* dans chacune des vues intermédiaires utilisées dans le calcul de la *matrice d'interaction B*. Ainsi, nous connaîtrons la forme et la position de l'ellipse de référence pour une variation angulaire non nulle. En pratique, lors de la phase de suivi, nous appliquons trois fois cette correction aux paramètres de l'ellipse du motif actuel $\mu'_{ac(actuel)}$ pour obtenir les paramètres de l'ellipse du motif suivant $\mu'_{su(suivant)}$ comme le montre la figure 3.21 pour une variation en site uniquement :

Etape 1 : à partir des paramètres d'ellipse du motif courant $\mu'_{ac(actuel)}$ pour une valeur de site α_{actuel}, la **première correction** permet de calculer les paramètres d'ellipse du motif actuel $\mu'_{ac(ref)}$ pour un angle de site α_{ref} nul.

1. **base d'apprentissage**
   - $\mu_{ac(actuel)}$ est le vecteur des paramètres de l'ellipse correspondant au résultat du suivi du motif de référence actuel (40) dans l'image intermédiaire (46) lors de la phase d'apprentissage hors ligne. Ce vecteur connu a été calculé pour une variation angulaire en site $\alpha_{actuel} = 6$ degrés (car les images intermédiaires sont acquises tous les degrés).
   - $\mu_{ac(ref)}$ est le vecteur des paramètres de l'ellipse du motif de référence actuellement suivi (40) dans la *vue de référence* (40) pour une variation en site $\alpha_{ref} = 0$. Nous rappelons ici que la forme et la position de l'ellipse dans l'image de référence sont définies manuellement par l'opérateur lors de la phase d'apprentissage. Cette ellipse est ensuite échantillonnée pour donner le *vecteur de forme* associé au motif de référence qui devra être suivi.
2. **phase de suivi en ligne**
   - $\mu'_{ac(actuel)}$ est le vecteur corrigé des paramètres de l'ellipse correspondant au motif recalé dans l'image courante pour une variation en site estimée $\alpha_{actuel} = 6$ degrés.
   - $\mu'_{ac(ref)}$ est le *vecteur de paramètres à déterminer* dans l'image courante pour une variation en site α_{ref} nulle.

Etape 2 : la **deuxième correction** assure le passage du motif actuellement suivi au prochain motif de référence suivi.

1. **base d'apprentissage**
   - $\mu_{ac(ref)}$ est le vecteur des paramètres de l'ellipse du motif actuellement suivi (40) défini dans la *vue de référence* (40) pour une variation en site $\alpha_{ref} = 0$.
   - $\mu_{su(ref)}$ est le vecteur des paramètres de l'ellipse du prochain motif suivi (50) défini dans la *vue de référence* (50) pour une variation en site α_{ref} nulle.
2. **phase de suivi en ligne**
   - $\mu'_{ac(ref)}$ est le vecteur estimé dans l'étape précédente pour le motif de référence actuellement suivi (40) et une variation en site $\alpha_{ref} = 0$.
   - $\mu'_{su(ref)}$ est le *vecteur de paramètres à déterminer* dans l'image courante pour le nouveau motif de référence à suivre (50) et une variation en site α_{ref} nulle.

Etape 3 : à partir des paramètres d'ellipse du motif suivant $\mu'_{su(ref)}$ pour une valeur de site α_{ref} nulle, la **troisième correction** permet le calcul des paramètres d'ellipse du motif suivant $\mu'_{su(suivant)}$ pour un angle de site $\alpha_{suivant}$.

1. **base d'apprentissage**
   - $\mu_{su(ref)}$ est le vecteur des paramètres de l'ellipse du nouveau motif suivi (50) défini dans la *vue de référence* (50) pour une variation en site α_{ref} nulle.
   - $\mu_{su(suivant)}$ est le vecteur des paramètres de l'ellipse correspondant au résultat du suivi du nouveau motif de référence (50) dans l'image intermédiaire (46) lors de la phase d'apprentissage hors ligne. Ce vecteur connu a été calculé pour une variation angulaire en site $\alpha_{suivant} = -4$ degrés.
2. **phase de suivi en ligne**
   - $\mu'_{su(ref)}$ est le vecteur estimé dans l'étape précédente pour le nouveau motif de référence suivi (50) et une variation en site $\alpha_{ref} = 0$.
   - $\mu'_{su(suivant)}$ est le *vecteur de paramètres à déterminer* dans l'image courante pour le nouveau motif de référence à suivre (50) et une variation en site $\alpha_{suivant} = -4$ degrés.

Comme pour le changement de motif de référence dans le cas où l'objet reste centré durant l'acquisition de la base d'apprentissage (collection d'images 2D), nous recalons le nouveau motif de référence suivi $\mu'_{su(suivant)}$ en multipliant la *matrice d'interaction* A correspondante par le vecteur courant de différence de niveaux de gris. Ceci permet de valider le changement de motif et de corriger les erreurs d'approximation se propageant de l'étape 1 à l'étape 3.

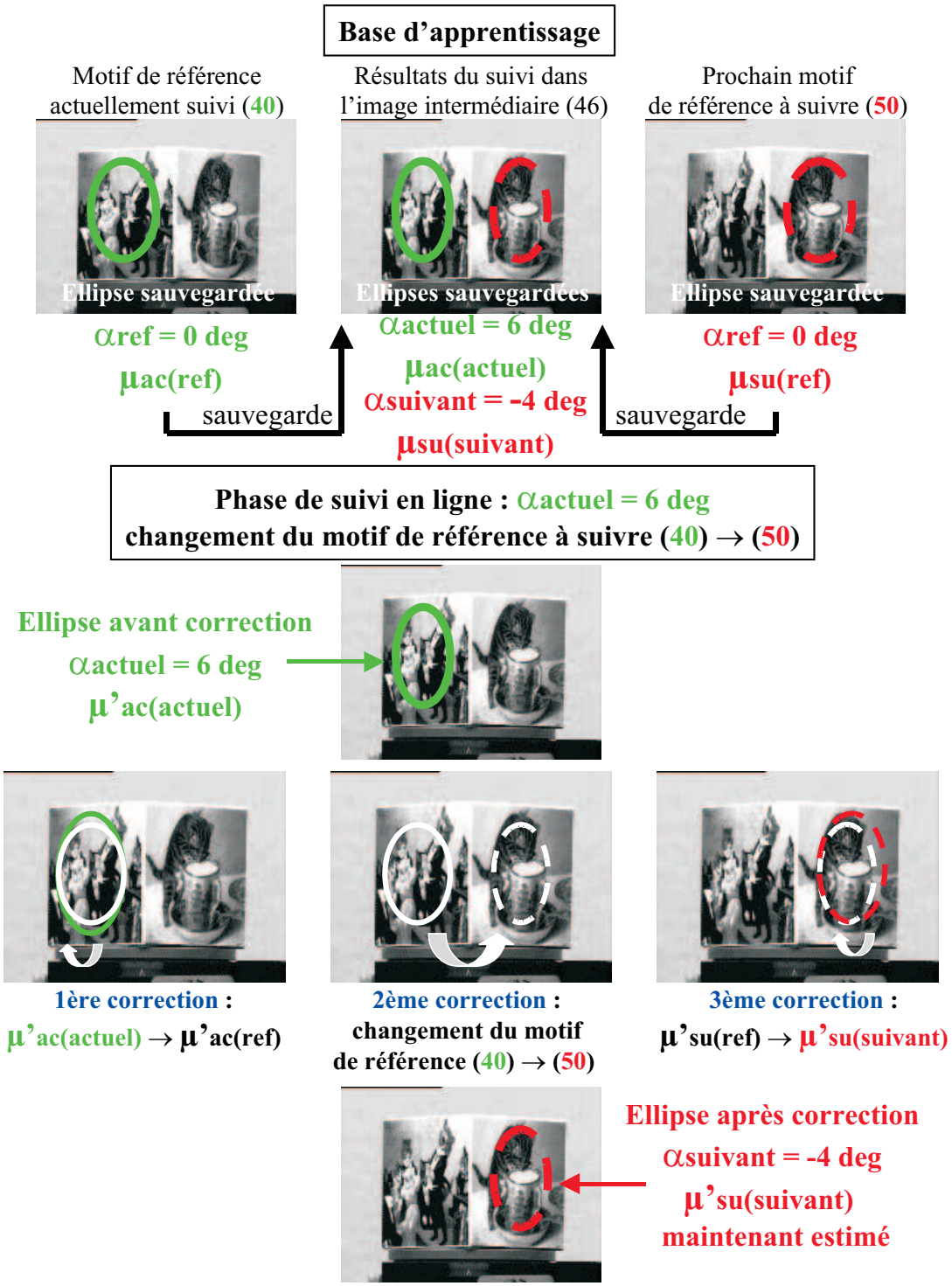


FIGURE 3.21 – corrections intermédiaires des paramètres de l'ellipse lors du changement de motif de référence pendant le suivi (objet non centré).

3.3.5 Optimisation de la taille du motif suivi dans l'image

Il s'agit ici de traiter les changements de motif de référence afin de garder le motif suivi dans une zone d'intérêt la plus grande possible dans l'image pour des translations axiales suivant l'axe optique (translation T_z sur la figure 3.8). Ces déplacements correspondent en fait au

rapprochement ou à l'éloignement de la caméra par rapport à l'objet. L'application envisagée peut être la robotique manufacturière où un bras manipulateur, équipé d'une caméra sur son effecteur, doit naviguer autour de l'objet pour assurer sa saisie.

Le problème se ramène alors à déterminer si le motif suivi après correction de sa position prédite est entièrement visible ou pas dans l'image courante avant l'estimation des variations d'apparence. Ceci représente donc un test supplémentaire et indépendant des règles de décision prises pour les variations d'aspect en site et azimut du motif courant présentées dans le paragraphe précédent. En effet, nous avons toujours supposé que lors de la phase de suivi, le motif était globalement visible dans l'image courante.

En isolant les translations axiales suivant l'axe optique des autres déplacements possibles de l'objet dans l'image, nous avons vu que les paramètres qui caractérisaient ce type de mouvement (changement d'échelle) étaient les rayons R_1 et R_2 de l'ellipse (figure 3.8). Par conséquent, lors du changement de motif de référence, nous conservons l'orientation planaire (θ) et les coordonnées du centre (X_c, Y_c) de l'ellipse actuelle pour la nouvelle zone d'intérêt. Les longueurs des axes R_1 et R_2 sont mises à jour à partir des différents facteurs d'échelle définis lors de la phase d'apprentissage hors ligne.

Pour pouvoir optimiser la taille du motif dans l'image suivant une translation axiale T_z, plusieurs collections d'images de référence sont nécessaires où chaque motif de référence d'une collection est superposé avec le motif de référence d'une autre collection (même orientation planaire (θ) et mêmes coordonnées (X_c, Y_c) pour les ellipses de référence). Durant la phase d'apprentissage hors ligne, les acquisitions de ces différentes collections sont séparées par des phases de suivi en ligne pour calculer les rapports entre les différents rayons des ellipses englobant les motifs de référence de deux collections d'images superposées (figure 3.22). Ceci nous permettra, par la suite, de passer d'une collection à une autre durant la phase de suivi. En fait, dans le cas que nous avons traité en pratique, un seul ratio est à estimer puisque pour une collection d'images donnée, une seule ellipse de référence est utilisée. Ceci s'explique par le fait que lors de l'acquisition des collections d'images 2D, l'objet reste centré dans l'image et que sa forme est cylindrique (canette de soda).

Le mode opératoire de la phase d'apprentissage hors ligne sur deux niveaux de référence superposés (i) et $(i+1)$ se résume de la manière suivante :

1. acquisition de la base d'images de référence du niveau (i) correspondant à une position caméra/objet la plus éloignée, puis définition des paramètres de l'ellipse de référence $el_{b(i)}$ (en particulier les rayons $R_{1b(i)}$ et $R_{2b(i)}$) et calcul des différentes *matrices d'interaction* $A_{b(i)}$ et $B_{b(i)}$ associées à chacun des motifs de référence.

2. tout en se rapprochant de l'objet, exécution en ligne de l'algorithme de suivi pour un des motifs de référence du niveau (i) avec une variation en site nulle et ceci tant que le motif reste entièrement visible dans l'image. Ensuite, arrêt du suivi pour sauvegarder les paramètres de l'ellipse corrigée $el'_{b(i)}$ (les rayons $R'_{1b(i)}$ et $R'_{2b(i)}$) du motif de référence du niveau (i) et définition des nouveaux paramètres de l'ellipse de référence $el_{b(i+1)}$ (les rayons $R_{1b(i+1)}$ et $R_{2b(i+1)}$) pour la collection d'images du niveau $(i+1)$.

3. acquisition de la base d'images de référence du niveau $(i+1)$ correspondant à une position caméra/objet moins éloignée et calcul des différentes *matrices d'interaction* $A_{b(i+1)}$ et $B_{b(i+1)}$ associées à chacun des motifs de référence (l'ellipse de référence $el_{b(i+1)}$ du niveau $(i+1)$ étant définie à l'étape précédente).

Notons que dans le cas où nous aurions n niveaux superposés à définir (avec $n > 0$), il suffit de réitérer $(n-1)$ fois les trois étapes du mode opératoire en commençant par les niveaux (1) et (2), puis (2) et (3) pour terminer avec les niveaux $(n-1)$ et (n).

A partir des informations $R'_{1b(i)}$, $R'_{2b(i)}$, $R_{1b(i+1)}$ et $R_{2b(i+1)}$, nous sommes maintenant capables, pendant la phase de suivi en ligne, de passer d'un motif de référence du niveau (i) à un motif de référence superposé du niveau $(i+1)$ et inversement. Si nous définissons $R_{1actuel}$ et $R_{2actuel}$ comme les rayons de l'ellipse corrigée du motif actuellement suivi dans l'image courante, les rayons de la nouvelle ellipse $R_{1nouveau}$, $R_{2nouveau}$ en cas de changement de motif de référence sont donnés par les formules suivantes :

1. *Passage du niveau* (i) *au niveau* $(i+1)$ *(on se rapproche de l'objet)*

$$R_{1nouveau} = \frac{R_{1b(i+1)}}{R'_{1b(i)}} R_{1actuel} \quad et \quad R_{2nouveau} = \frac{R_{2b(i+1)}}{R'_{2b(i)}} R_{2actuel} \tag{3.13}$$

2. *Passage du niveau* $(i+1)$ *au niveau* (i) *(on s'éloigne de l'objet)*

$$R_{1nouveau} = \frac{R'_{1b(i)}}{R_{1b(i+1)}} R_{1actuel} \quad et \quad R_{2nouveau} = \frac{R'_{2b(i)}}{R_{2b(i+1)}} R_{2actuel} \tag{3.14}$$

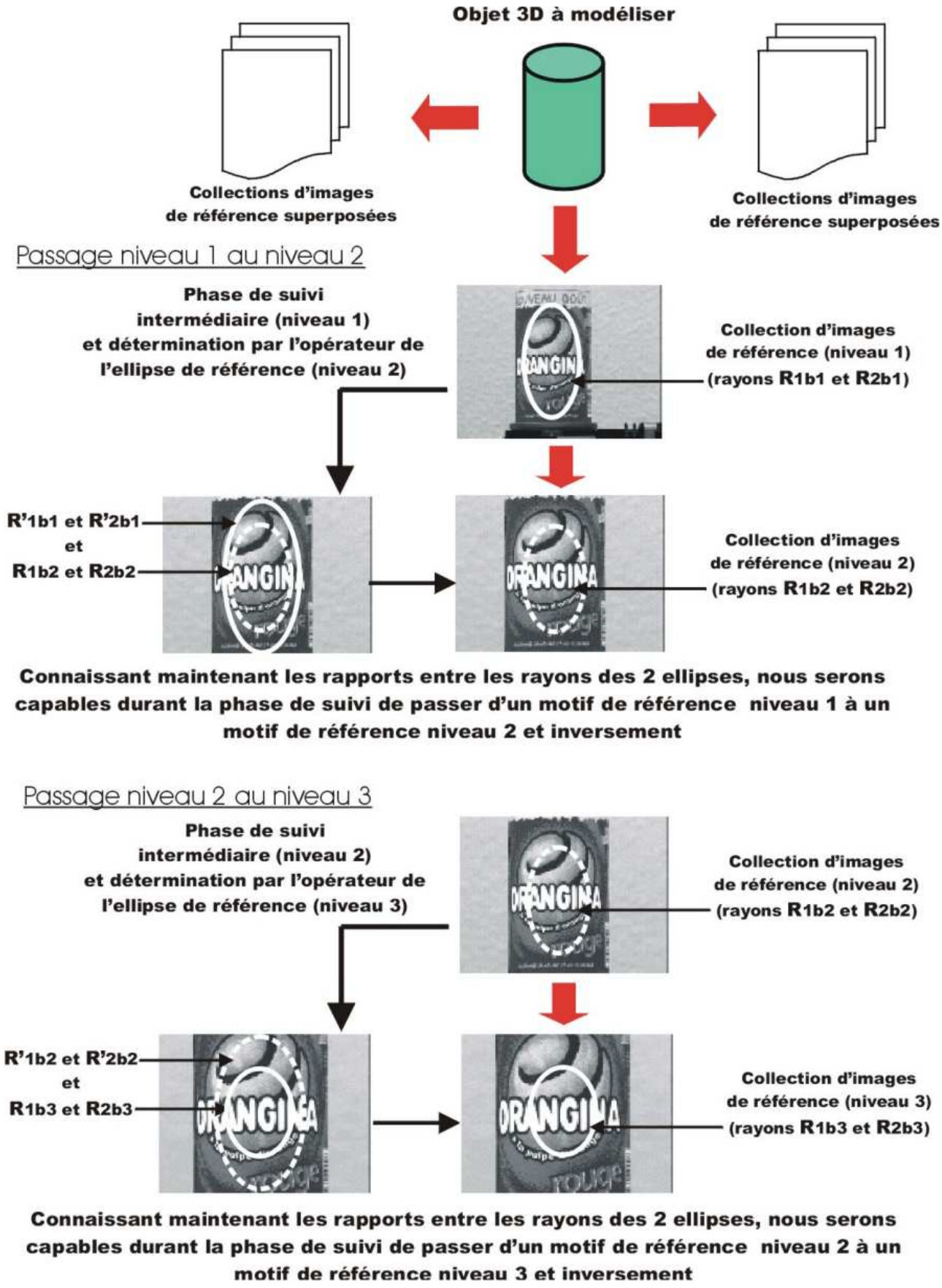


FIGURE 3.22 – principe d'acquisition sur 3 niveaux des différentes collections d'images de référence.

L'algorithme de suivi prenant en compte les variations d'apparence en site et l'optimisation du motif courant dans l'image sur trois niveaux d'apprentissage peut s'écrire de la manière suivante :

```
DEBUT PROCEDURE SUIVI : suivi actuel d'un motif du niveau (2)
 Correction de la position prédite μ_r = μ_p + AΔVI_p
 Si tous les points échantillonnés du motif sont dans l'image
    alors
     Test du passage du niveau (2) au niveau (1) (phase d'éloignement)
     Si tous les points du motif sont encore dans l'image
       alors
        Changement du motif de référence (2) vers (1)
       sinon
        Conservation du motif du niveau (2)
     Fin Si
    sinon
     Passage obligatoire du niveau (2) au niveau (3) (phase de rapprochement)
 Fin Si
 Gestion du changement d'aspect avec α = BΔVI_r
FIN PROCEDURE SUIVI
```

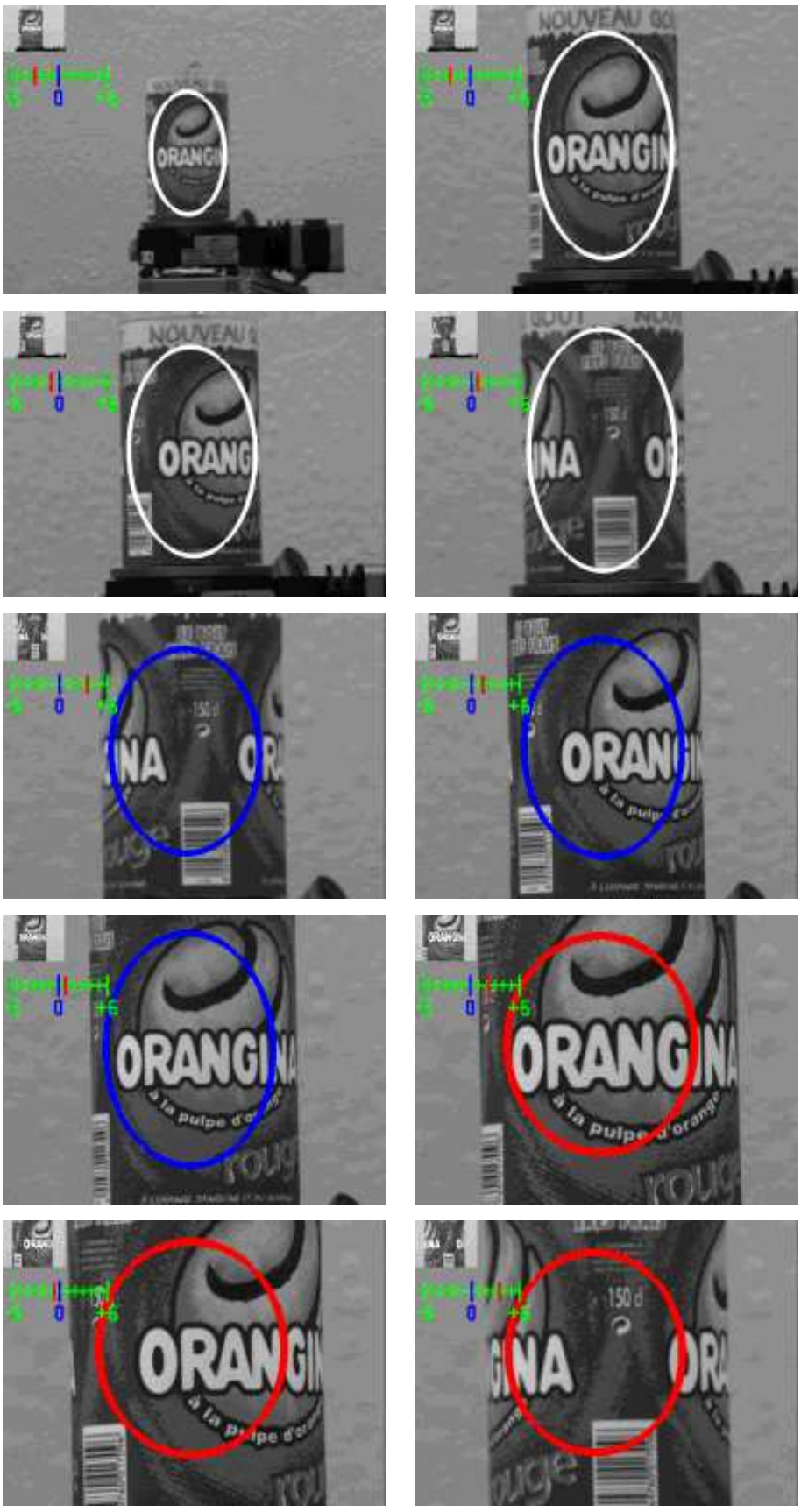


FIGURE 3.23 – exemple de suivi d'une canette de soda sur 3 niveaux d'apprentissage (ellipses blanche (niveau 1), bleue (niveau 2) et rouge (niveau 3)).

Cet algorithme est également illustré par la figure 3.23. Nous montrons ici le suivi d'un objet

cylindrique (canette de soda) sur trois niveaux d'apprentissage pour des variations en site uniquement. L'algorithme donne de bonnes performances en temps réel vidéo tout en assurant les différents changements de motif de référence. Notons également que le choix d'utiliser une étape intermédiaire de suivi entre deux acquisitions de collections d'images de référence nous permet de définir un nouveau motif de référence le mieux adapté pour assurer la continuité du suivi, tout en tenant compte des mouvements responsables des variations d'aspect (variations en site dans notre exemple). L'inconvénient majeur de cette approche est que la phase d'apprentissage hors ligne demande une mise en oeuvre conséquente et qui n'est pas automatisée.

3.4 Expérimentations

Dans ce paragraphe, nous présentons tout d'abord la phase d'initialisation du suivi ou de reconnaissance de motif, puis la phase de suivi en ligne sous forme d'un algorithme et différentes illustrations. Les programmes sont implantés sur une station de travail Silicon Graphics O_2 pour un temps d'exécution inférieur à 15 millisecondes.

3.4.1 Initialisation du suivi

Durant cette étape, nous supposons que l'objet à suivre reste immobile dans l'image. Tout d'abord, l'opérateur sélectionne à l'aide d'une ellipse le motif courant dans la première image. L'algorithme de reconnaissance d'aspect calcule alors, pour chaque image de référence de la base d'apprentissage, l'erreur quadratique de la différence de niveaux de gris entre le motif de référence testé et le motif courant échantillonné dans l'ellipse après correction de ses paramètres (c'est à dire après avoir recalé le motif à l'aide de la *matrice d'interaction* A associée).

Le motif de référence donnant alors l'erreur quadratique la plus faible sera reconnu comme le motif courant à suivre à la prochaine itération. Cette phase d'initialisation est illustrée par la figure 3.24.

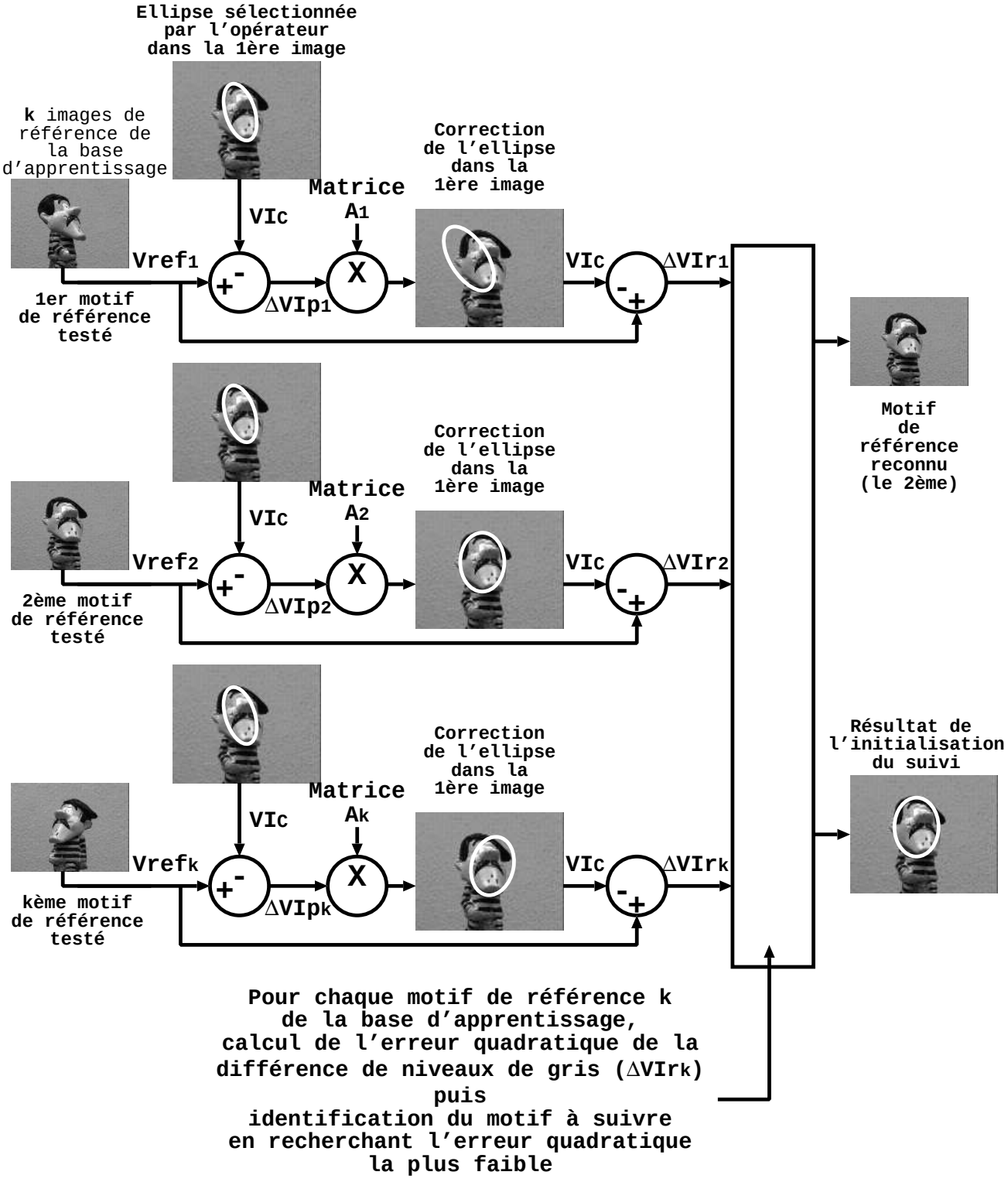


FIGURE 3.24 – principe d'initialisation de la phase de suivi.

3.4.2 Suivi d'objets texturés 3D

Compte tenu de son efficacité, l'implémentation de l'algorithme proposé permet de suivre en temps réel un objet 3D avec changement d'aspect (moins de 15 millisecondes par itération).

Cet algorithme peut s'écrire de la manière suivante :

DEBUT PROCEDURE SUIVI $(\mu, Image)$

μ_p = μ de l'itération précédente

VI_c = EchantillonnerMotif $(\mu_p, Image)$

$\mu_r^{actuel} = \mu_p + A^{actuel}(VI_{ref}^{actuel} - VI_c)$

VI_c = EchantillonnerMotif $(\mu_r^{actuel}, Image)$

$\psi^{actuel} = B^{actuel}(VI_{ref}^{actuel} - VI_c)$

$Correction$ = ChangerMotifDeRéférence (ψ^{actuel})

Si $Correction = 0$

alors pas de correction des paramètres ellipse

$\mu = \mu_r^{actuel}$, $\psi = \psi^{actuel}$

sinon correction des paramètres ellipse

μ'_{su} = CorrigerParamètresEllipse $(\mu'_{ac} = \mu_r^{actuel}, \mu_{ac}^{base}, \mu_{su}^{base})$

μ_p = μ'_{su}

VI_c = EchantillonnerMotif $(\mu_p, Image)$

$\mu_r^{suivant} = \mu_p + A^{suivant}(VI_{ref}^{suivant} - VI_c)$

VI_c = EchantillonnerMotif $(\mu_r^{suivant}, Image)$

$\psi^{suivant} = B^{suivant}(VI_{ref}^{suivant} - VI_c)$

$\mu = \mu_r^{suivant}$, $\psi = \psi^{suivant}$

Fin Si

FIN PROCEDURE SUIVI

Dans les exemples présentés maintenant (figures 3.26 à 3.30), trois objets sont utilisés : une canette de soda, un cube et une figurine. Etant donné la nature des données traitées (flot d'images vidéo), les résultats sont difficiles à représenter.

Nous avons donc choisi de visualiser les résultats du suivi à des instants différents de l'expérience. Les images retenues ont été sélectionnées de manière à montrer la robustesse du suivi pour des objets 3D possédant des propriétés intrinsèques très différentes :

- la *canette de soda* est un objet métallique de forme cylindrique et de texture plutôt complexe.
- le *cube* est un objet de forme cubique (comme son nom l'indique). Chaque face représente un motif différent imprimé sur du papier qualité photo mat.
- la *figurine* est un objet de forme quelconque réalisé en plâtre puis peint. Nous nous

intéressons ici au suivi 3D du visage aux traits accentués.

Cet algorithme de suivi d'objets 3D nous fournit en temps réel trois informations (figure 3.25) :

1. le motif de référence actuellement suivi représenté par une imagette dans le coin supérieur gauche de la fenêtre de visualisation,
2. les valeurs angulaires en site et azimut (α et β) données par deux curseurs sur des échelles graduées,
3. le résultat du suivi du motif à l'intérieur de l'ellipse.

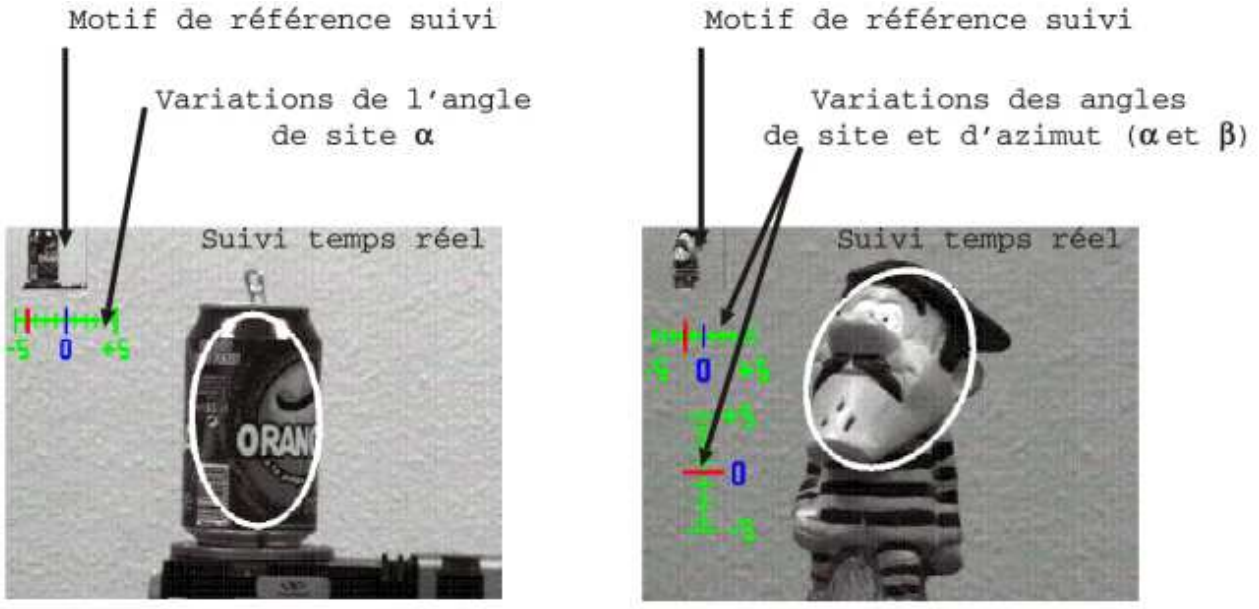


FIGURE 3.25 – différentes informations temps réel calculées par l'algorithme de suivi d'objets 3D.

Les premiers essais de suivi 3D ont donné des résultats positifs. Le passage d'un motif de référence à un autre s'effectue correctement. Le suivi est assuré quelque soit le changement d'échelle du motif dans l'image (figure 3.29) et pour des mouvements de grande amplitude entre deux images consécutives de la séquence. Cette méthode de suivi 3D reste également stable pour des positions caméra/objet entraînant de faibles variations d'aspect en azimut du motif courant dans l'image non apprises au cours de la phase d'apprentissage hors ligne (exemple du cube illustré par les figures 3.27 et 3.28).

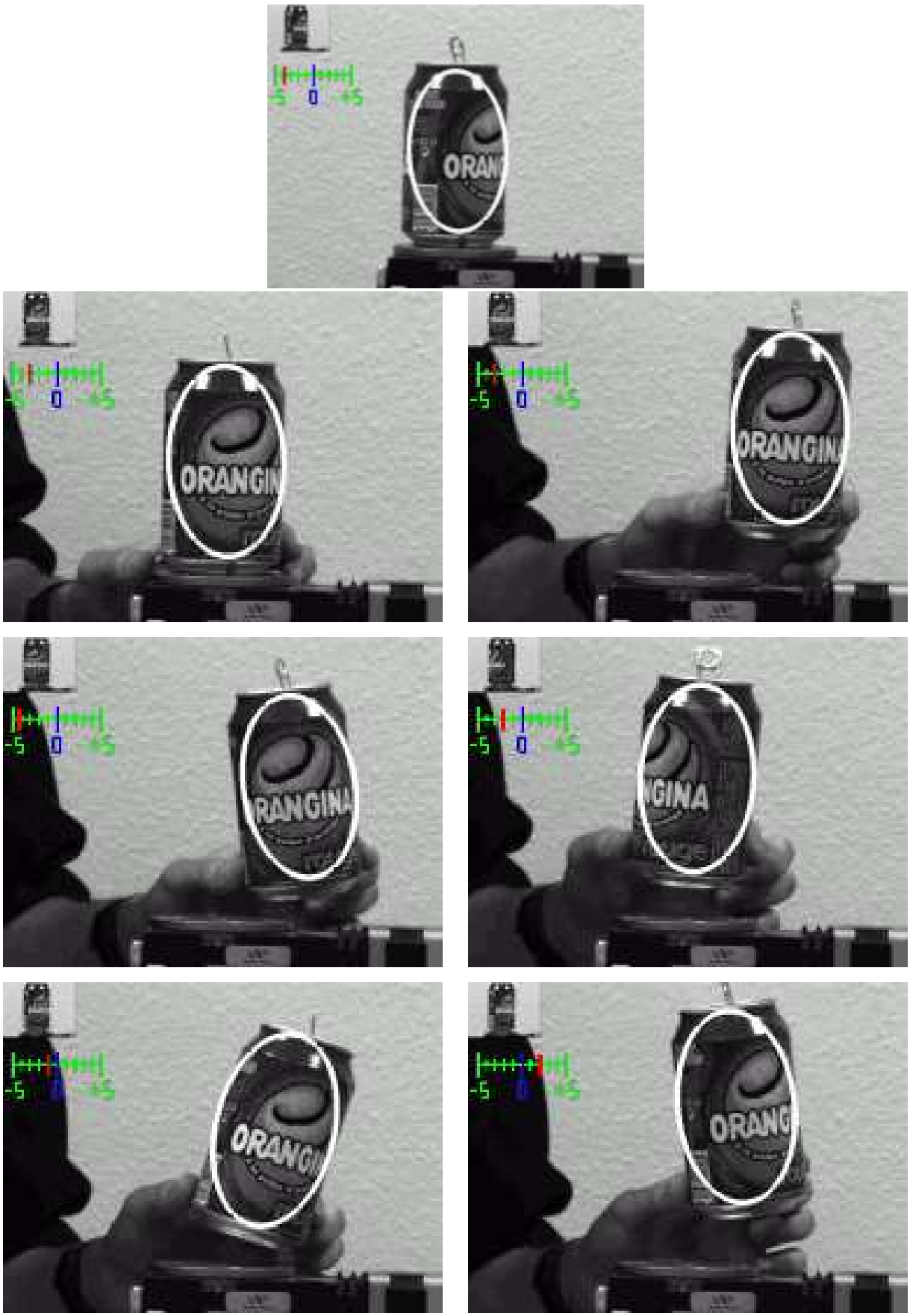


FIGURE 3.26 – suivi 3D temps réel d'une canette de soda.

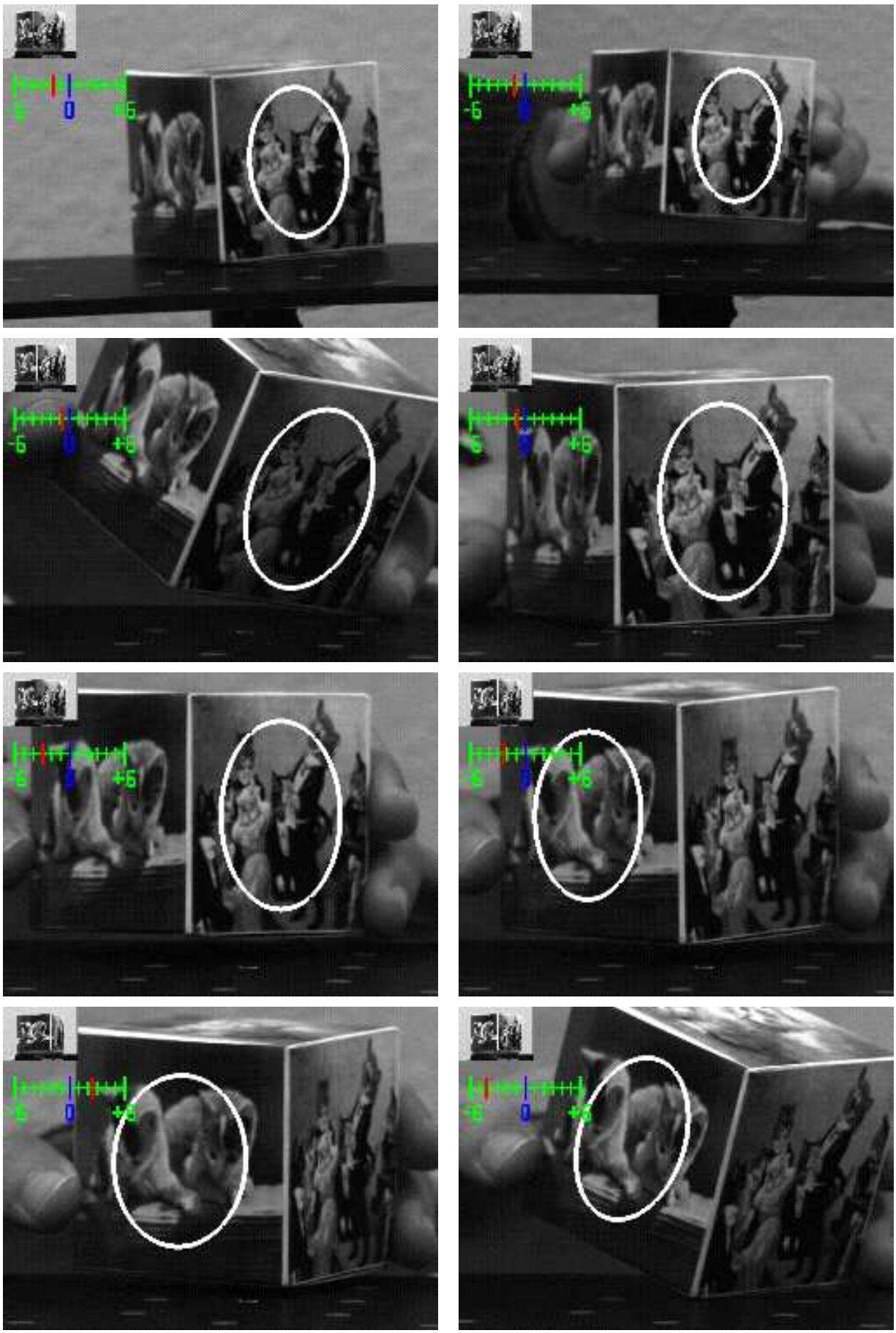


FIGURE 3.27 – suivi 3D temps réel d'un cube : séquence 1.

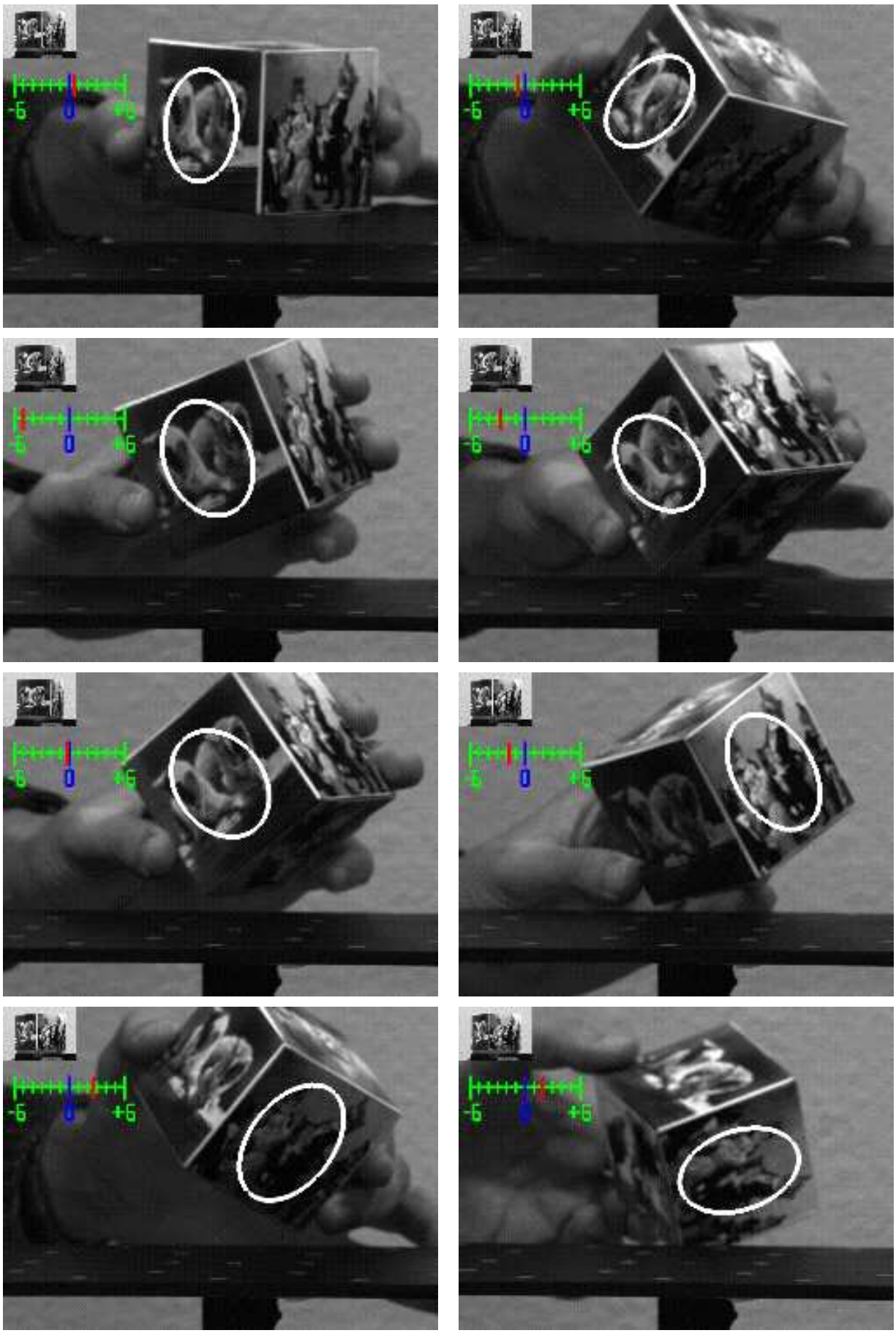


FIGURE 3.28 – suivi 3D temps réel d'un cube : séquence 2.

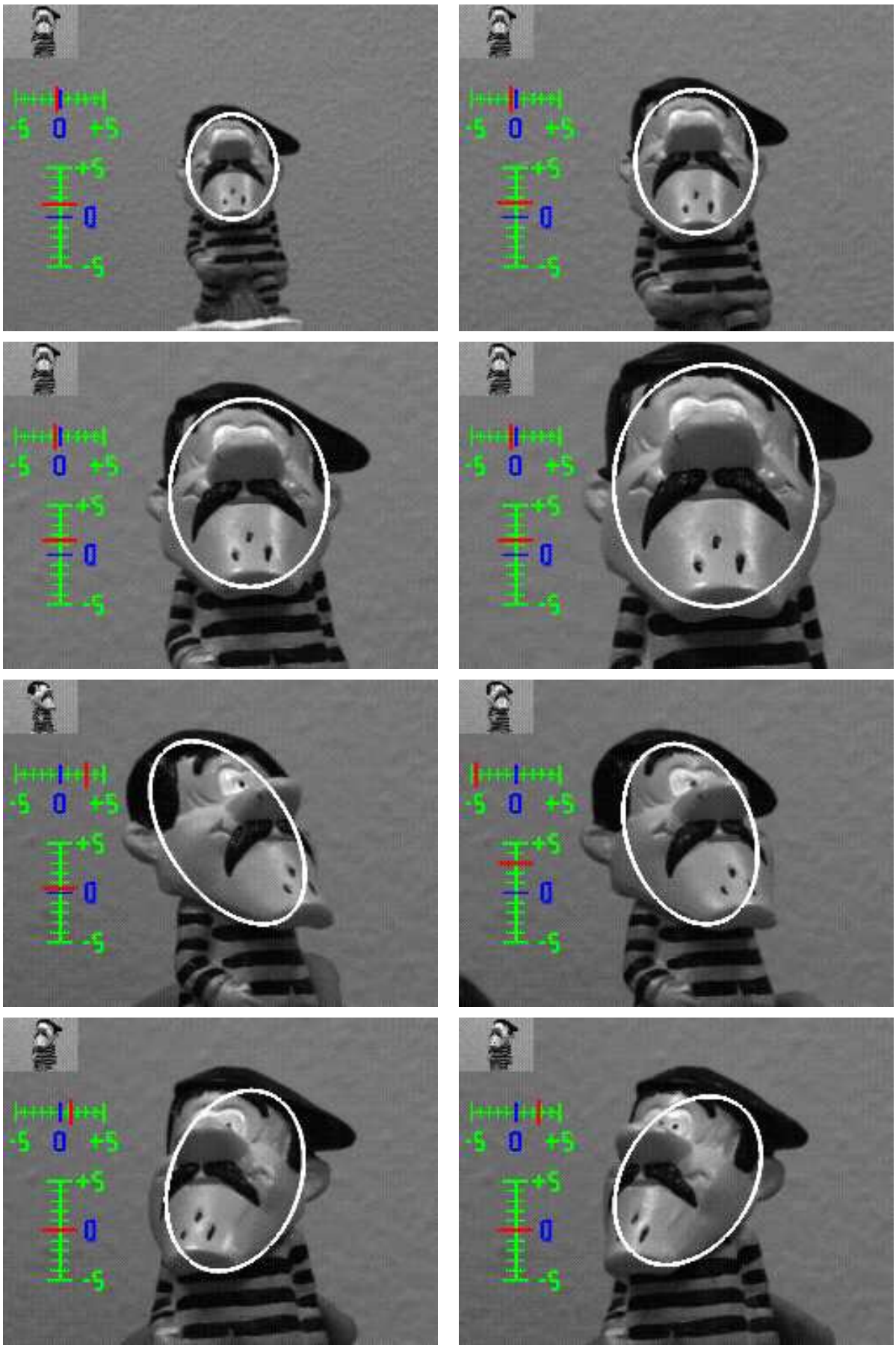


FIGURE 3.29 – suivi 3D temps réel d'une figurine : séquence 1.

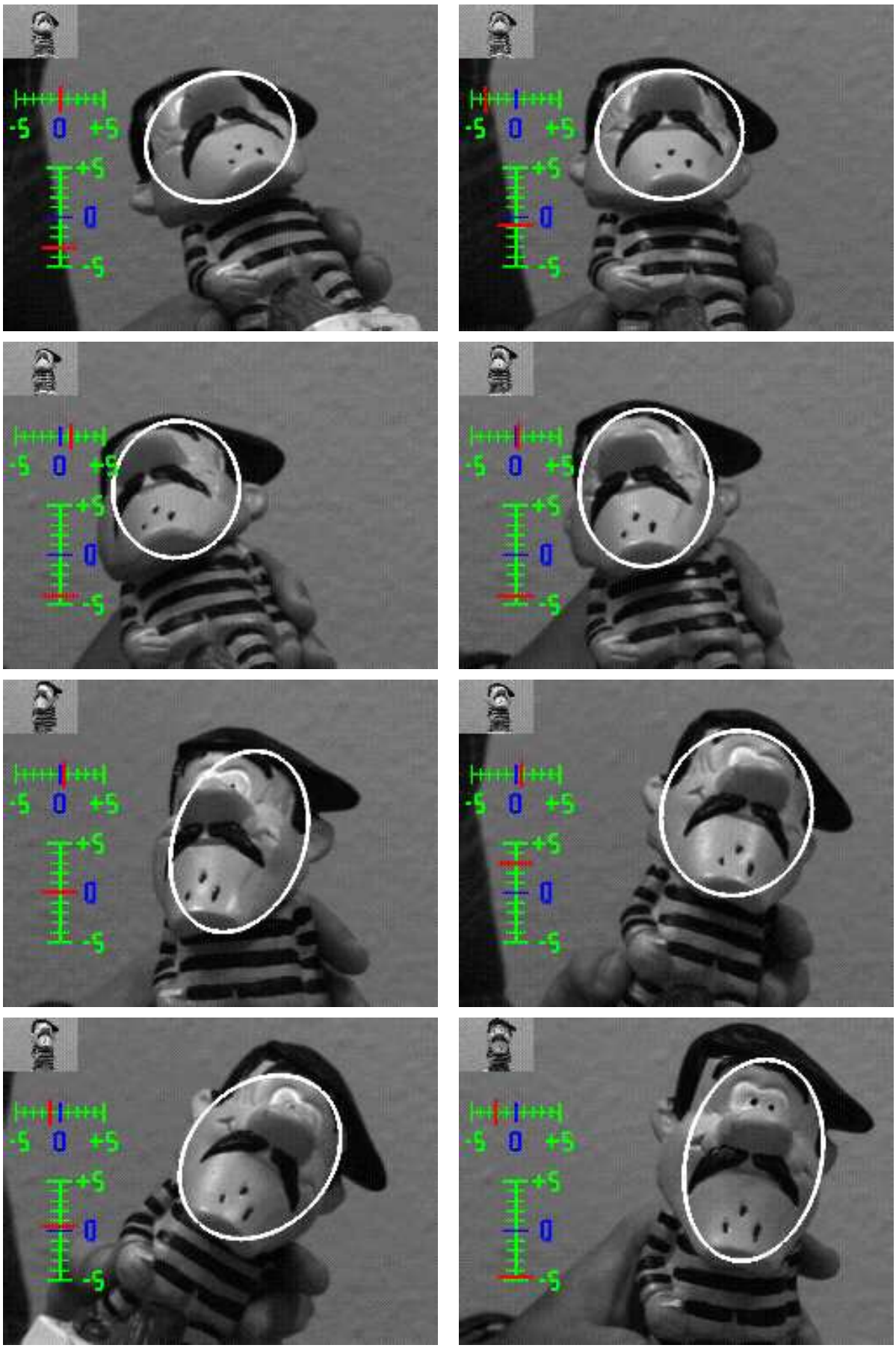


FIGURE 3.30 – suivi 3D temps réel d'une figurine : séquence 2.

Pour pouvoir être robuste aux changements de fond texturé dans l'image, il faudrait s'assurer que pendant la phase d'apprentissage hors ligne de la *matrice d'interaction A* les ellipses perturbées autour de leur position de référence restent dans l'objet. Mais ceci limiterait l'amplitude des mouvements autorisés entre deux images consécutives. De plus, comme toute différence de niveaux de gris se traduit par un déplacement, nous restons sensible à certaines perturbations pouvant se produire lors du suivi comme l'apparition d'occultations. C'est pourquoi, nous avions proposé dans le chapitre précédent une solution temps réel traitant le problème des occultations. Cette solution pourrait aussi bien considérer le changement de fond comme l'apparition d'une occultation lors de la phase de suivi. C'est pourquoi, pour la logique de cette thèse, il aurait fallu cumuler les deux approches et faire des essais de suivi en changeant de fond. Cela n'a pas été fait par manque de temps.

3.5 Conclusion et perspectives

Dans ce chapitre, nous avons présenté une solution de suivi d'objets 3D, temps réel, basée sur l'apparence et qui gère les changements d'aspect du motif. Pour cela, six paramètres sont nécessaires :

- les quatre premiers paramètres (X_c, Y_c, R_1 et θ) caractérisent les *mouvements fronto parallèles* de l'objet dans l'image où l'aspect du motif suivi n'est pas modifié. Toutefois, la position, l'orientation et la taille du motif peuvent changer.
- les deux derniers paramètres (α et β) caractérisent les variations en site et azimut de l'objet dans l'image modifiant alors l'aspect du motif suivi.

Notre méthode présente des originalités comme le calcul des *matrices d'interaction A* et B sans utiliser les matrices Jacobiennes de l'image (méthode d'approximation par hyperplans présentée dans le chapitre 2), ou une phase d'exploration autour de la prédiction supprimée lors du suivi. C'est une méthode généraliste car les variations de position de l'objet dans l'image correspondent aux variations des paramètres d'une transformation géométrique dont le choix est à l'initiative du programmeur suivant l'aspect et le volume de l'objet à suivre. De plus, c'est un algorithme très peu coûteux en temps de calcul (multiplication d'une matrice par un vecteur).

Notre souhait fut ensuite de développer des algorithmes de suivi 3D pour deux applications particulières : le suivi d'un visage et la navigation d'un bras robotique autour d'un objet 3D connu qui font maintenant l'objet du dernier chapitre.

Chapitre 4

Applications dédiées au suivi 3D

Dans ce dernier chapitre, nous allons présenter des applications de suivi 3D dédiées aux domaines de la visioconférence et de la robotique manufacturière (figure 4.1). Le premier algorithme permet la détection automatique d'un visage dans une séquence d'images pour assurer son suivi 3D. En cas de perte du motif suivi, il réinitialise automatiquement l'application en recherchant un nouveau motif de référence dans l'image.

Le deuxième algorithme permet, quant à lui, de commander en position la table à déplacement micrométrique. A partir du motif suivi dans l'image courante, nous commandons en "plus ou moins" la position de la table, par pas de un degré, pour atteindre le motif de référence désiré. Cette étape intermédiaire était importante pour valider l'intégration de notre algorithme de suivi 3D dans une boucle d'asservissement avant de travailler sur le robot portique du laboratoire.

Pour cette dernière application, deux commandes ont été développées :

- *une commande en position* : à partir du motif suivi dans l'image courante, le bras manipulateur, équipé d'une caméra sur son effecteur, doit naviguer autour d'un objet connu et fixe pour atteindre le motif de référence désiré.
- *une commande en vitesse* : le robot suit les déplacements de l'objet devant la caméra de façon à garder au centre de l'image courante le motif suivi tout en gérant ses changements d'aspect.

Notons enfin que l'algorithme de suivi 3D que nous proposons est identique pour l'ensemble des applications présentées dans ce chapitre.

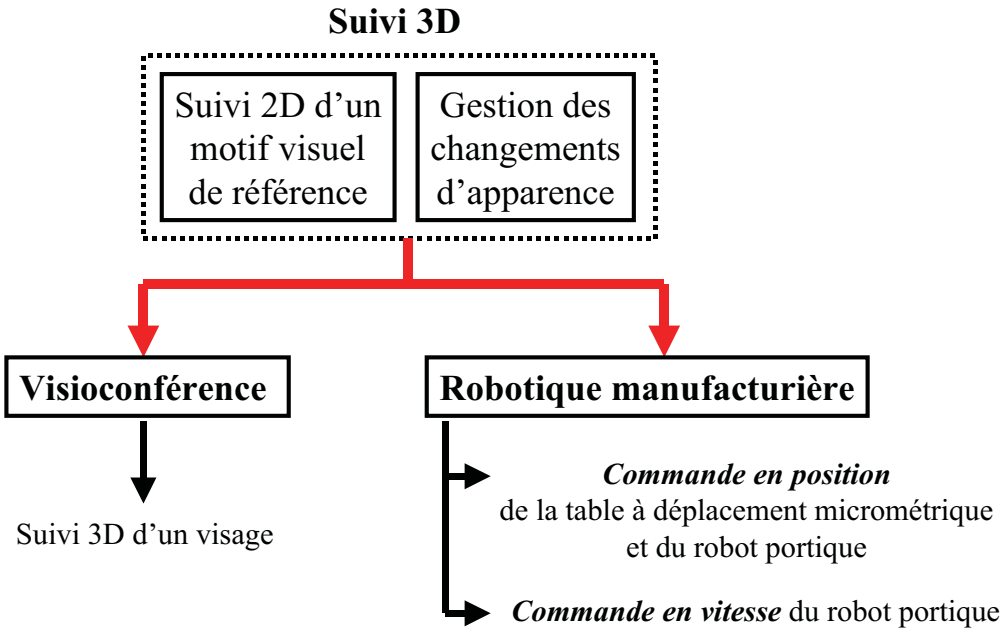


FIGURE 4.1 – présentation des applications de suivi 3D.

4.1 Détection et suivi 3D d'un visage dans une séquence vidéo

Dans cette section, nous allons développer la méthode de suivi 3D d'un visage basée sur l'apparence et les différents travaux présentés dans les chapitres précédents. L'une des applications envisagées est la visioconférence où la transmission et la compression des données sont des problèmes importants. Pour diminuer la taille des données utilisées, il serait intéressant de transmettre soit le motif suivi à l'intérieur de la zone d'intérêt soit l'image de différence entre le motif de référence et le motif courant échantillonné après correction pour reconstruire l'expression faciale de l'individu à partir de la collection d'images de référence qui aurait été transmise à l'interlocuteur à l'initialisation de l'application (lors de la première connexion).

Dans les paragraphes suivants, nous allons présenter les différents travaux déja réalisés sur le suivi de visages avant de développer notre solution. Les performances de la méthode seront estimées tout d'abord en simulation pour tester la robustesse et la fiabilité du suivi avant l'expérimentation temps réel sur un flot vidéo. Ceci nous permettra également de comparer la taille des données compressées entre l'image du motif courant suivi après correction de la position de la zone d'intérêt et l'image de différence de niveaux de gris correspondante.

Cet algorithme séquentiel a fait l'objet d'une publication [47] et une étude de parallélisation

à la demande des personnes du thème *Architectures des Machines de Perception* du laboratoire pour la thèse de Monsieur Rémi Coudarcher [27] (voir Annexe B sur la *parallélisation d'un algorithme de suivi 3D à l'aide de squelettes fonctionnels* [29] [30]).

4.1.1 Etat de l'art

Les statistiques montrent qu'approximativement, 75% des visages sur des images ne sont pas photographiés de face [75] et qu'il est important de gérer les différentes variations de pose d'un visage dans les applications de vision artificielle. C'est pourquoi, de nombreux travaux sur la reconnaissance, la modélisation et le suivi de visages ont été menés durant ces dernières années. Ces approches sont généralement basées sur des méthodes dites *"Eigenfaces"* [109], *"Elastic Graph Model"* [77], *"Linear Object Classes"* [110], *"Active Shape Models (ASMs)"* [23] et *"Active Appearance Models (AAMs)"* [25].

Modélisation des visages avec de larges variations de pose

En particulier, la modélisation des visages par une collection de vues est un véritable défi qui pose les problèmes de l'apparition d'occultations et d'ombres partielles mais aussi de la non linéarite résultante entre la forme et la texture. Pour traiter ce problème, Romdhani *et al.* [97] ont développé un modèle d'apparence basé sur différentes vues en utilisant une analyse connue sous *"Kernel Principal Component Analysis (KPCA)"*. La non linéarité de cette analyse permet au modèle de gérer de larges variations de pose. Toutefois, c'est une opération coûteuse en temps d'exécution.

Cootes *et al.* [24] ont proposé les *"Active Appearance Models (AAMs)"* pour représenter un visage. Pour cela, ils utilisent trois modèles associés aux vues de profil, de face et pour une position intermédiaire. Un *AAM* est généré en combinant un modèle de variation de forme avec un modèle de variation d'apparence. Il est défini à partir d'un ensemble de points caractéristiques représentant au mieux les traits et l'expression du visage à modéliser (points situés autour des yeux, de la bouche, du nez et délimitant le contour du visage). La phase d'apprentissage hors ligne consiste à déterminer la relation entre les variations des paramètres du modèle et l'erreur résiduelle de niveaux de gris entre l'image d'apprentissage et l'image synthétisée par le modèle. Ceci permettra, par la suite, de corriger en ligne la prédiction des paramètres courants du modèle lors de sa mise en correspondance dans une image.

Une autre approche consiste à utiliser les *"Active Shape Models (ASMs)"*. Ce modèle utilise les caractéristiques locales de l'image (contours, luminosité) pour se déformer progressivement

et prendre la forme de la caractéristique souhaitée. Il existe trois types d'*ASMs* :

1. le premier utilise un contour générique actif appelé *"snake"* introduit par Kass *et al.* en 1987 [71]. Afin de localiser correctement le visage dans l'image, la première initialisation du *snake* doit être proche de la solution c'est à dire à proximité des contours du visage. Son évolution est obtenue ensuite à partir de la minimisation d'une fonction d'énergie.
2. le deuxième est basé sur la notion de modèle déformable *"deformable template"* introduit par Yuille *et al.* [115]. Ce modèle a été développé pour palier aux problèmes de détection rencontrés avec le *snake* pour un contraste de luminosité faible.
3. la dernière solution a été proposée plus tard par Cootes *et al.* en utilisant un modèle générique flexible appelé *"Point Distributed Model (PDM)"* [78] qui permet de donner une description paramétrée et compacte de la forme et de l'apparence du visage. Ce dernier modèle permet d'obtenir une interprétation efficace du visage humain.

D'autre part, des méthodes ont été présentées par Moghaddam et Pentland [84] à l'aide des *"Eigenspaces"* ou par Li *et al.* [79] en introduisant la notion de *"Support Vector Machine (SVM)"* pour modéliser le visage dans un espace. Mais la division de cet espace dans ces méthodes est généralement arbitraire et souvent imprécise. Une approche intermédiaire est d'utiliser des modèles 3D plutôt qu'une collection d'images 2D pour modéliser le visage. DeCarlo et Metaxas [37] ont présenté un modèle 3D déformable d'un visage dans lequel le flot optique et l'information de contour sont combinés. Leur modèle a suivi avec succès des visages dans des séquences avec des changements significatifs de pose et d'expression.

Modélisation dynamique des visages sur des séquences

En parallèle de la modélisation des visages à travers plusieurs vues, l'exploitation des dynamiques du visage en utilisant des informations spatio-temporelles des séquences vidéos a été alors reçue avec un grand intérêt. Dans ces séquences vidéos, non seulement l'information sur des objets visuels peut être acquise mais la continuité temporelle et la constance du sujet peuvent alors fournir une représentation plus robuste [56]. Gong *et al.* [57] ont introduit une approche qui utilise des réseaux de neurones pour reconnaître des signatures temporelles des visages. Yamaguchi *et al.* [113] ont présenté une méthode de reconnaissance de visages qui consiste à construire un sous-espace pour les visages détectés dans une séquence donnée et de mettre ensuite en correspondance ce sous-espace avec les sous-espaces prototypes.

La solution de suivi 3D que nous proposons maintenant permet de suivre un visage dans une

image avec de fortes variations de pose (plus particulièrement des variations en site) entraînant des changements d'aspect plus ou moins importants. Ces variations d'apparence sont gérées à partir d'une collection d'images 2D de référence modélisant le visage 3D pour différentes poses données. Pour présenter notre algorithme, nous conservons les notations prises dans le chapitre précédent.

4.1.2 Solution retenue pour le suivi 3D d'un visage

Principe du suivi 3D

Cet algorithme permet un suivi 3D temps réel d'un visage dans une séquence vidéo tout en gérant les variations d'aspect, principalement pour des variations en site. Pour cela, le visage 3D est modélisé par une collection d'images 2D de référence. Nous rappelons ici que par définition, un *motif* est une région de l'image à l'intérieur d'une zone d'intérêt et que son échantillonnage donne un vecteur de niveaux de gris appelé aussi *vecteur de forme*.

Pour chacun de ces motifs de référence, une *matrice d'interaction* A est apprise lors d'une phase d'apprentissage hors ligne basée sur l'*approximation par hyperplans* développée dans le chapitre 2. Comme nous l'avons vu, cette matrice lie les différences de niveaux de gris entre le motif de référence actuellement suivi et le motif courant échantillonné dans la zone d'intérêt après correction à son déplacement fronto parallèle. En pratique, deux matrices d'interaction A_1 et A_2 sont apprises distinctement pour des niveaux différents de perturbation pour assurer une correction dite *"grossière à fine"* de la position réelle du motif dans l'image.

Pour gérer les variations d'aspect et assurer le changement de motif de référence durant la phase de suivi en ligne, nous n'utilisons plus la matrice d'interaction B. En effet, la matrice B ne peut pas donner des résultats fiables pour l'estimation des variations d'aspect en site du fait que le visage est un objet déformable avec des variations d'expression. Ces changements vont entraîner obligatoirement une erreur dans le vecteur de différence de niveaux de gris et par conséquent une erreur dans le calcul de l'*angle de site* α. De plus, nous verrons par la suite que la méthode d'acquisition de la collection d'images 2D n'aurait pas permis d'estimer correctement la matrice d'interaction B. La solution retenue alors pour traiter le problème des variations d'aspect en site sera de tester simultanément durant la phase de suivi en ligne les résultats de suivi obtenus à partir du motif de référence actuellement suivi et de ses plus proches voisins dans la collection d'images 2D.

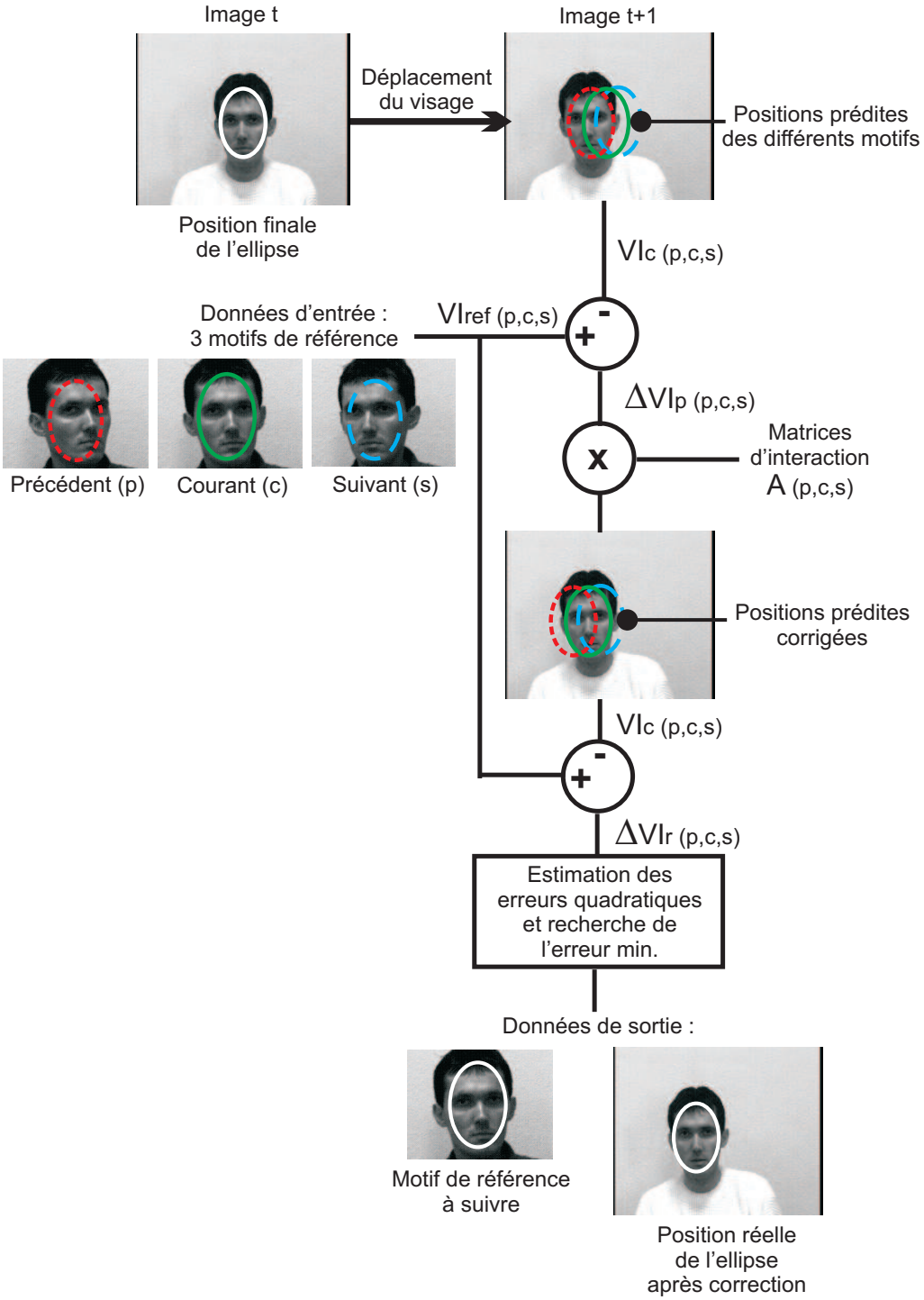


FIGURE 4.2 – principe du suivi 3D de visages.

L'étape de suivi en ligne (figure 4.2) consiste à prédire la position du motif dans l'image $\mu_{p(p,c,s)}$ (en position, échelle et orientation) et à estimer les corrections $\Delta\mu_{(p,c,s)}$ à apporter pour chacun des motifs de référence testés (motifs courant (c), précédent (p) et suivant (s)). Cette correction est calculée en multipliant la différence de niveaux de gris $\Delta VI_{p(p,c,s)}$ entre le motif échantillonné à l'endroit prédit $VI_{c(p,c,s)}$ et le motif de référence testé $VI_{ref(p,c,s)}$ par la

matrice d'interaction associée $A_{(p,c,s)}$. Le problème du suivi du motif dans l'image se ramène alors à la correction des paramètres d'une transformation affine (les 5 paramètres de l'ellipse $(X_c, Y_c, R_1, R_2, \theta)$) par la détermination d'un vecteur d'offset $\Delta\mu_{(p,c,s)}$.

Pour chacun des motifs de référence testés, nous obtenons une position prédite corrigée qui est supposée la position réelle $\mu_{r(p,c,s)}$ du motif dans l'image. Pour chacune de ces positions, nous calculons l'erreur quadratique de la différence de niveaux de gris $\Delta VI_{r(p,c,s)}$ entre le motif courant échantillonné dans cette nouvelle zone d'intérêt $VI_{c(p,c,s)}$ et le motif de référence associé $VI_{ref(p,c,s)}$. Le motif de référence donnant l'erreur quadratique la plus faible sera considéré comme le nouveau motif de référence à suivre dans la prochaine image.

Ce test comparatif entre plusieurs motifs de référence nous permet donc :

- de gérer les variations d'apparence du motif suivi (variations majeure d'aspect en site et modérée en azimut) suite à un changement d'orientation du visage par rapport au capteur vision,
- de changer de motif de référence tout en continuant d'assurer le suivi 3D en temps réel vidéo (moins de 20ms par itération).

Construction du modèle 3D du visage basée sur l'apparence

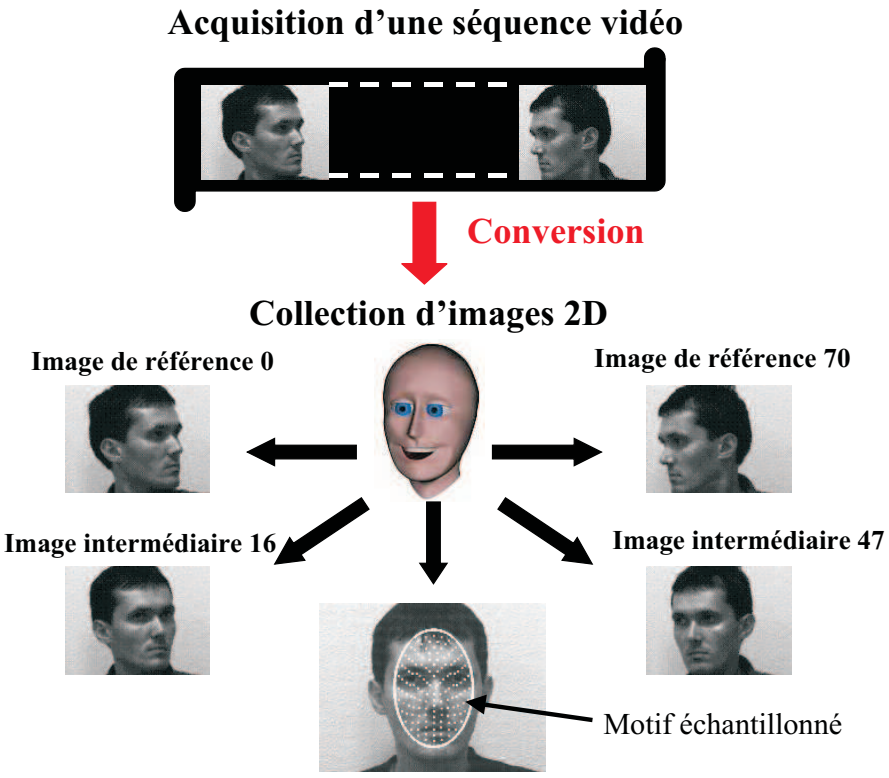


FIGURE 4.3 – modélisation 3D du visage et échantillonnage du motif dans une ellipse.

Par définition, un visage est un objet variable et déformable. Ses apparences dans une image dépendent fortement des conditions d'éclairement et bien entendu de la pose, de l'expression et de l'identité de la personne. Dans notre approche de suivi 3D basée sur l'apparence, un visage est modélisé par une collection d'images 2D. Pour acquérir cette base d'apprentissage, l'individu, assis devant la caméra, positionne son visage au milieu de l'image et exécute une rotation de la tête de la gauche vers la droite ou inversement pour avoir une variation d'aspect en site du visage sur 180 degrés (du profil gauche au profil droit du visage). Ce mouvement enregistré sous forme d'une séquence vidéo est décomposé par la suite en une collection d'images 2D où nous allons définir les vues de référence et les vues intermédiaires (figure 4.3).

Chacune de ces vues de référence représente un des motifs de référence du visage à un instant du suivi pour une position caméra/visage donnée. Ces images, comme nous le savons, permettent d'assurer le suivi 2D du motif. De plus, comme nous avons supposé que le visage reste centré lors de l'acquisition vidéo pour la création de la collection d'images, une seule ellipse de référence est nécessaire pour l'ensemble des vues de référence.

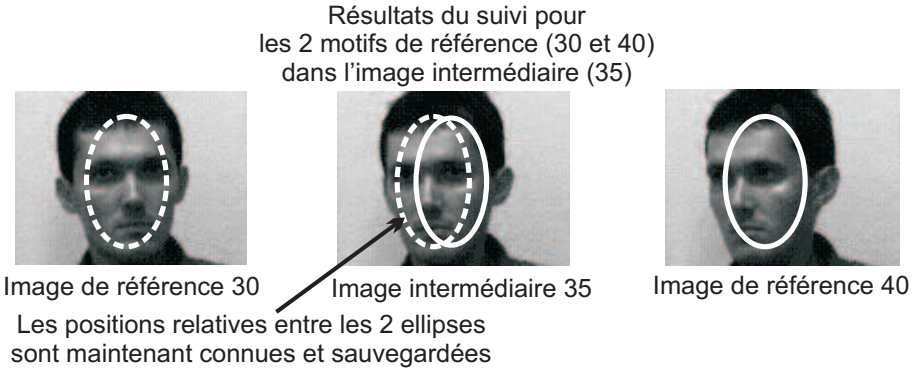


FIGURE 4.4 – sauvegarde des résultats du suivi de 2 motifs de référence consécutifs dans une image intermédiaire de la base d'apprentissage.

Les vues intermédiaires, quant à elles, nous permettront de sauvegarder durant la phase d'apprentissage, les différents résultats de suivi (les positions corrigées de l'ellipse) pour chacun des motifs de référence les plus proches dans la base d'images (figure 4.4). Lors de la phase de suivi, certains de ces résultats, en particulier ceux obtenus pour les images intermédiaires situées au milieu (à *"mi-chemin"*) de deux motifs de référence consécutifs, seront utilisés avec la position prédite du motif actuellement suivi (c) pour estimer les positions avant correction des zones d'intérêt correspondant aux motifs de référence précédent (p) et suivant (s) dans l'image courante (figure 4.5). Cette étape intermédiaire est nécessaire pour assurer un passage efficace

d'un motif de référence à l'autre.

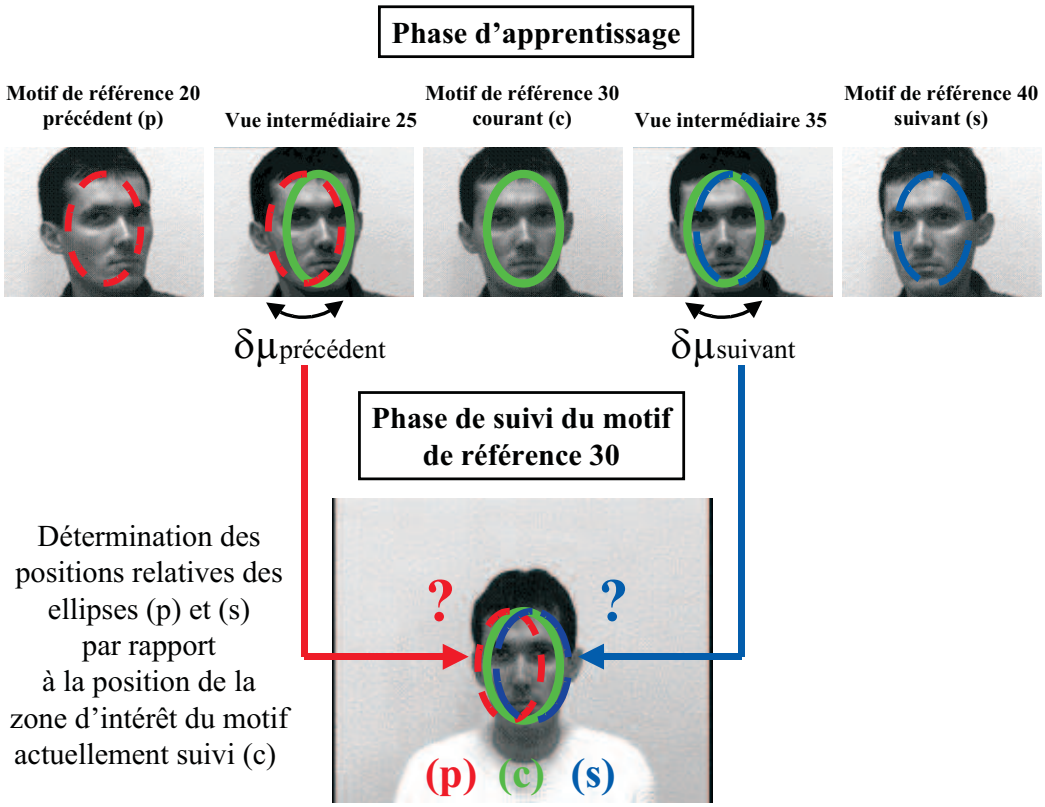


FIGURE 4.5 – estimation des positions relatives des ellipses (p) et (s) par rapport à la position prédite de l'ellipse (c).

Comme l'amplitude du mouvement en rotation du visage peut varier entre deux images consécutives de la base, il n'apparaissait pas judicieux de calculer une matrice d'interaction B pour estimer de façon efficace les variations d'apparence en site. De plus, les changements d'expression du visage peuvent varier au cours du suivi et entraîner des erreurs dans le vecteur de différence de niveaux de gris et par conséquent dans l'estimation de la variation d'aspect en site du motif actuellement suivi. Ainsi, le test simultané de trois motifs de référence consécutifs lors de la phase de suivi filtre au mieux les changements d'expression du visage et les variations d'aspect dues à des mouvements non modélisés durant l'acquisition de la collection d'images 2D (en particulier, les mouvements de faibles amplitudes en azimut).

Pour les résultats présentés dans cette section, la base d'apprentissage comprend 71 images numérotées de 0 à 70 pour une variation privilégiée en site sur 180 degrés. Les images, dont l'indice est un multiple de 10, sont les vues de référence (soit un total de 8 images). Nous supposons, par principe, que nous sommes capables d'assurer le suivi d'un motif de référence

donné dans les vues intermédiaires jusqu'à l'un de ses plus proches voisins.

Comme dans le chapitre précédent, l'échantillonnage du motif à l'intérieur d'une zone d'intérêt (ici une ellipse) reste identique (figure 4.3). Le vecteur de forme obtenu de taille N comprend non plus 373 points mais seulement 170 points répartis sur un ensemble de 10 ellipses concentriques échantillonnées de la plus petite à la plus grande. Cet échantillonnage régulier accompagné d'une diminution du nombre de points permet de limiter les influences des changements d'expression du visage sans pour autant perdre de la robustesse dans le suivi en ligne. La position et la forme de l'ellipse sont définies par un vecteur μ à cinq paramètres correspondant à la position de son centre (X_c, Y_c), son orientation (θ) et les longueurs de ses axes (R_1, R_2) (figure 4.6).

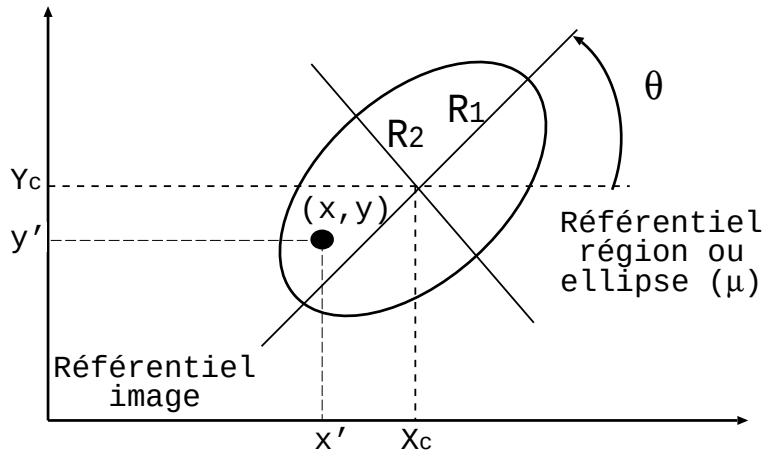


FIGURE 4.6 – rappel sur la définition des paramètres caractérisant l'ellipse.

Présentation de l'algorithme de suivi 3D

Dans ce paragraphe, nous allons rappeler brièvement les idées principales qui caractérisent notre solution de suivi 3D utilisée pour l'ensemble des applications présentées dans ce chapitre. Notre algorithme de suivi 3D peut se décomposer en deux modules qui assurent les fonctions suivantes :

1. le suivi 2D d'un motif visuel de référence,
2. la gestion des changements d'apparence du motif suivi.

Le module de *suivi 2D d'un motif de référence* dans une séquence d'images (figure 4.7) se résume par l'équation suivante :

$$\mu_r = \mu_p + \Delta\mu = \mu_p + A\Delta VI_p = \mu_p + A(VI_{ref} - VI_c) \tag{4.1}$$

où A est la matrice d'interaction permettant la mise à jour des paramètres d'une transformation affine définis par les paramètres de l'ellipse englobant le motif à suivre. Cette matrice est apprise durant une phase d'apprentissage hors ligne.

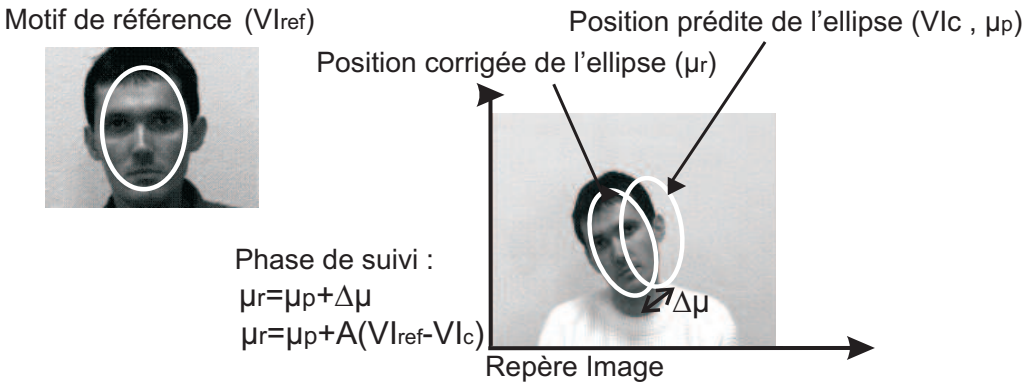


FIGURE 4.7 – rappel du principe de suivi 2D avec la matrice d'interaction A.

Le module de *gestion des changements d'apparence du motif suivi* doit permettre, quant à lui, un enchaînement efficace des motifs de référence tout en assurant le suivi 3D du visage. Pour cela, nous comparons en permanence les résultats du suivi obtenus à partir du motif de référence actuellement suivi (ou courant (c)) et de ses plus proches voisins (motifs précédent (p) et suivant (s)) dans la base d'apprentissage en terme d'erreur quadratique. Le motif de référence donnant la plus faible erreur quadratique de la différence de niveaux de gris $\Delta VI_{r(p,c,s)}$ entre son vecteur de forme $VI_{ref(p,c,s)}$ et le vecteur $VI_{c(p,c,s)}$ du motif échantillonné dans l'ellipse corrigée sera considéré comme le nouveau motif de référence à suivre dans la prochaine image.

Mais pour pouvoir calculer les corrections à appliquer à la position prédite de l'ellipse dans l'image pour chacun des motifs de référence voisins du motif de référence actuellement suivi, il faut rajouter une étape intermédiaire dans la phase de suivi en ligne. En effet, la position prédite de la zone d'intérêt dans l'image est définie à partir du motif de référence courant (c) et non par rapport aux motifs de référence les plus proches dans la base d'apprentissage (motifs précédent (p) et suivant (s)). Il faut donc calculer pour chacun des motifs de référence voisins une position prédite de l'ellipse dans l'image par rapport à la position prédite actuellement définie par le motif de référence suivi avant d'estimer la position réelle de l'ellipse corrigée et de calculer l'erreur quadratique associée. Ainsi, pour les vues intermédiaires situées à *"mi-chemin"* des motifs de référence les plus proches dans la base d'apprentissage, nous calculons et sauvegardons les différentes positions des ellipses correspondant aux résultats du suivi pour chacun des motifs de référence testés (figure 4.8). Nous choisissons en particulier ces images

intermédiaires car nous supposons en pratique que le changement de motif de référence se situe autour de ces variations d'apparence.

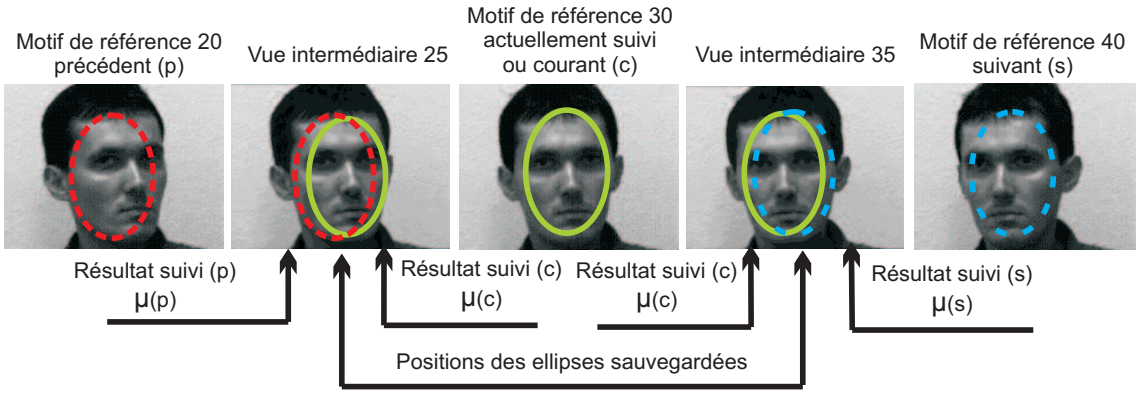


FIGURE 4.8 – utilisation des vues intermédiaires de la base d'apprentissage pour le positionnement des ellipses prédites dans la phase de suivi.

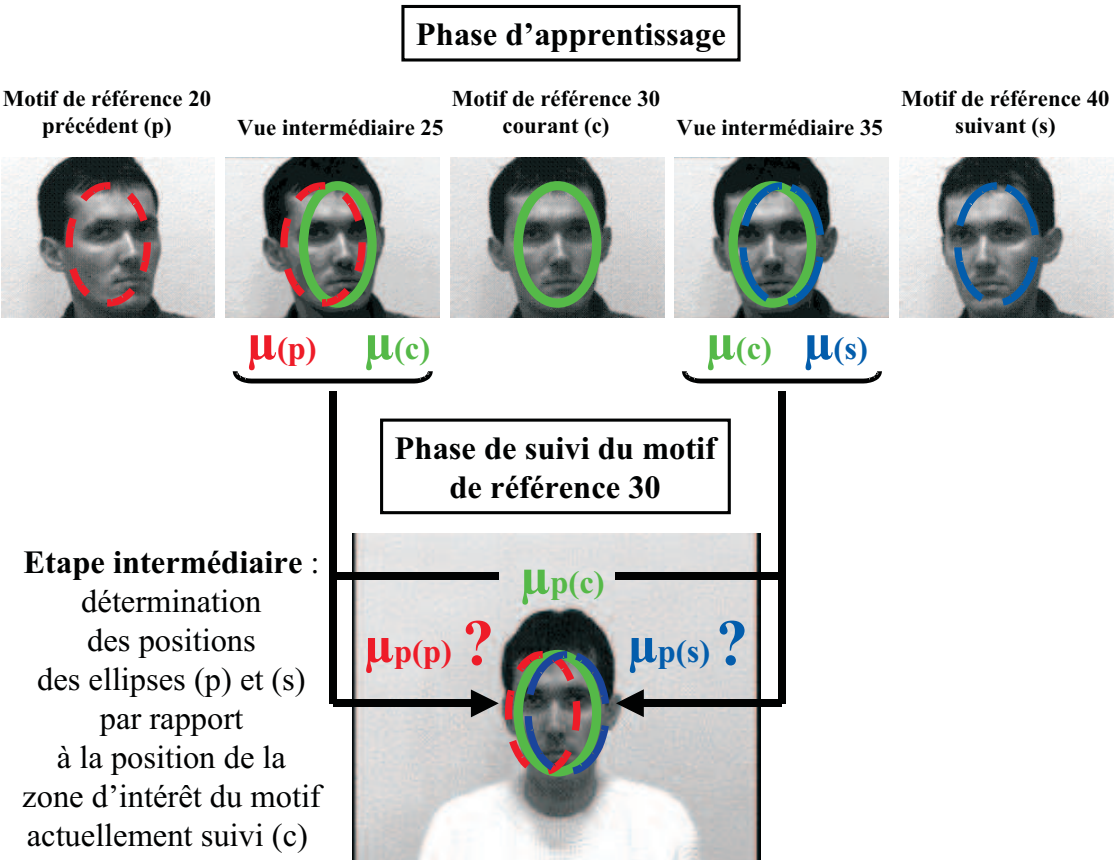


FIGURE 4.9 – informations nécessaires pour le calcul du positionnement des ellipses prédites dans la phase de suivi.

Pendant cette étape intermédiaire du suivi (figure 4.9), il s'agit d'exprimer les paramètres des ellipses prédites $\mu_{p(p)}$ et $\mu_{p(s)}$ des motifs de référence précédent (p) et suivant (s) en fonction des paramètres de l'ellipse prédite $\mu_{p(c)}$ du motif de référence actuellement suivi (c) et ceux

correspondant dans la base d'apprentissage ($\mu_{(c)}$, $\mu_{(p)}$ et $\mu_{(s)}$) avant de calculer les corrections pour obtenir les différentes positions réelles des ellipses $\mu_{r(p,c,s)}$ dans l'image. Ces calculs ont été développés dans le chapitre précédent au moment de la gestion du changement de motif de référence pour le suivi 3D d'un objet. Ils correspondent en fait à des changements d'échelle, d'orientation et des écarts relatifs de position que nous allons brièvement rappeler ici en traitant le cas du motif de référence suivant (s) (la méthode restant identique pour le motif de référence précédent (p)).

En prenant les notations suivantes pour les paramètres de l'ellipse considérée :

1. **résultats du suivi dans l'image intermédiaire (35) de la base d'apprentissage :**
   - $\mu_{(c)} = (X_{c(c)}, Y_{c(c)}, R_{1(c)}, \theta_{(c)})^t$ pour le motif de référence suivi (c),
   - $\mu_{(s)} = (X_{c(s)}, Y_{c(s)}, R_{1(s)}, \theta_{(s)})^t$ pour le motif de référence suivant (s) dans la base d'apprentissage.

2. **calculs de la position prédite lors de la phase de suivi en ligne :**
   - $\mu_{p(c)} = (X_{cp(c)}, Y_{cp(c)}, R_{1p(c)}, \theta_{p(c)})^t$ pour la position prédite du motif actuellement suivi (c),
   - $\mu_{p(s)} = (X_{cp(s)}, Y_{cp(s)}, R_{1p(s)}, \theta_{p(s)})^t$ pour la position prédite du motif de référence suivant (s) à estimer.

3. **par définition** $R_2 = k * R_1$ où le coefficient k est fixé lors de la phase d'apprentissage.

Nous obtenons alors les relations suivantes :

1. longueurs des axes de l'ellipse pour le changement d'échelle :

$$R_{1p(s)} = \frac{R_{1(s)}}{R_{1(c)}} R_{1p(c)} \quad et \quad R_{2p(s)} = k * R_{1p(s)} \tag{4.2}$$

2. orientation de l'ellipse :

$$\theta_{p(s)} = \theta_{p(c)} + \Delta\theta = \theta_{p(c)} + (\theta_{(s)} - \theta_{(c)}) \tag{4.3}$$

3. coordonnées du centre de l'ellipse :
   En notant $k_{1(c)} = \frac{R_{1p(c)}}{R_{1(c)}}$ et $k_{2(c)} = \frac{R_{2p(c)}}{R_{2(c)}}$ les facteurs d'échelle pour passer de la base d'apprentissage à la phase de suivi en ligne, les coordonnées de la nouvelle ellipse prédite dans le référentiel de l'image courante sont obtenues par :

$$X_{cp(s)} = X_{cp(c)} + k_{1(c)}\Delta X \quad et \quad Y_{cp(s)} = Y_{cp(c)} + k_{2(c)}\Delta Y \tag{4.4}$$

avec :

$$\begin{cases} \Delta X = (X_{c(s)} - X_{c(c)})\cos\theta_{(c)} + (Y_{c(s)} - Y_{c(c)})\sin\theta_{(c)} \\ \Delta Y = -(X_{c(s)} - X_{c(c)})\sin\theta_{(c)} + (Y_{c(s)} - Y_{c(c)})\cos\theta_{(c)} \end{cases} \quad (4.5)$$

Nous supposons également que les erreurs de calcul sur les positions prédites pour les motifs de référence les plus proches dans la base d'images restent compatibles avec les variations d'amplitude des paramètres des ellipses apprises lors de la phase d'apprentissage hors ligne des matrices d'interaction A associées.

4.1.3 Simulation et Expérimentation temps réel

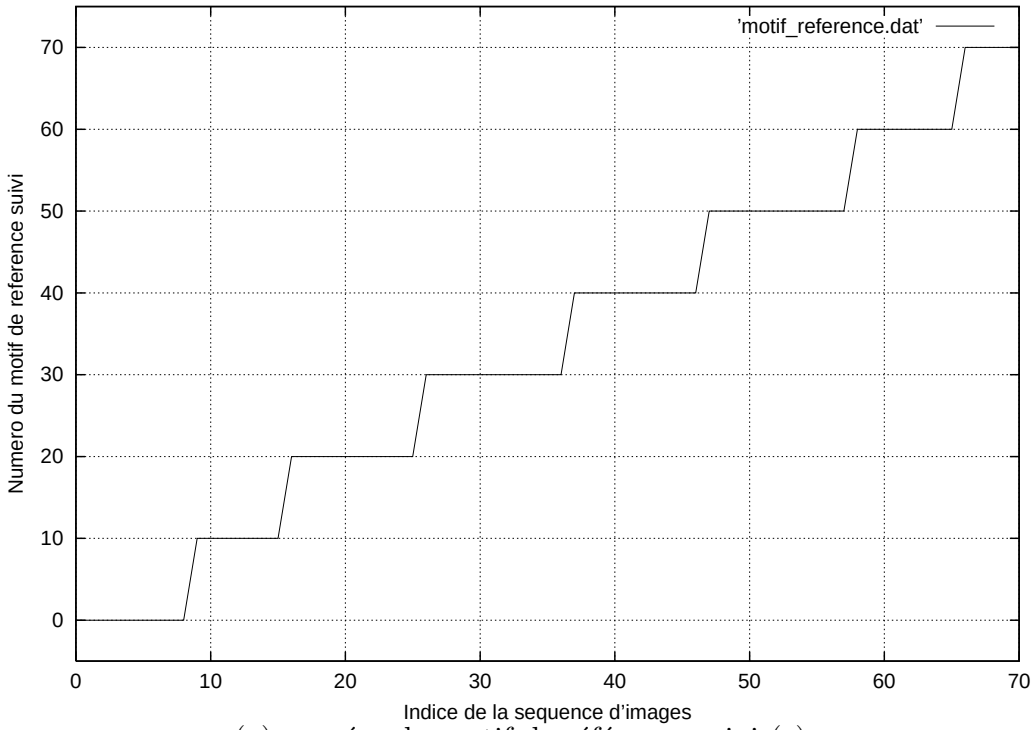
Dans ce paragraphe, nous allons présenter les performances de notre algorithme de suivi implémenté sur une station de travail Silicon Graphics O_2. Cet algorithme permet de suivre en temps réel vidéo un visage avec gestion des changements d'aspect ($<$ 20ms par itération). Pour valider l'efficacité de notre approche de suivi 3D, l'algorithme a d'abord été testé en simulation.

Résultats de l'algorithme de suivi de visages en simulation

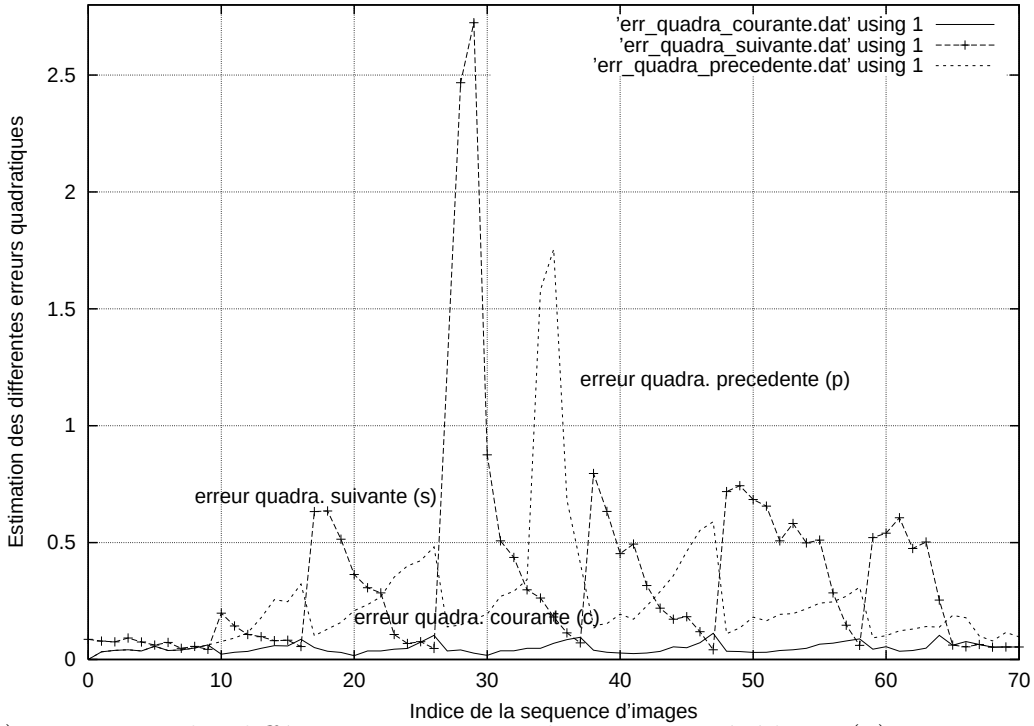
Durant cette étape de simulation, nous testons l'algorithme de suivi sur l'ensemble des images de la base d'apprentissage. Pour chacune de ces images, nous enregistrons les différentes erreurs quadratiques de la différence de niveaux de gris entre le motif de référence testé et le motif échantillonné dans l'ellipse corrigée ainsi que le numéro du motif de référence donnant l'erreur quadratique la plus faible et qui sera le nouveau motif à suivre dans la prochaine image.

Les figures 4.10 et 4.11 montrent les résultats obtenus par l'algorithme de suivi sur l'ensemble de la collection d'images. L'initialisation du suivi est réalisée soit sur la première image de la base d'apprentissage soit sur la dernière. Pour ces deux essais, l'enchaînement des motifs de référence à suivre s'effectue correctement par paliers successifs. La largeur d'un palier dépend principalement des variations d'apparence qui sont beaucoup plus importantes pour un visage vu de face que de profil lors d'un mouvement en rotation de la tête. De plus, l'amplitude du déplacement du visage n'est pas constante entre les différentes images constituant la base d'apprentissage du fait de la méthode d'acquisition de ces images (conversion d'une séquence vidéo). Ce changement de motif de référence s'effectue pour des images généralement situées à *"mi-chemin"* des motifs de référence les plus proches. Nous constatons également que l'erreur quadratique du motif de référence suivi reste relativement faible ($<$ à 0.26) devant les erreurs quadratiques des motifs de référence précédent et suivant de la base. Ceci permet donc d'avoir

aucune ambiguïté sur le motif de référence à suivre dans la prochaine image.

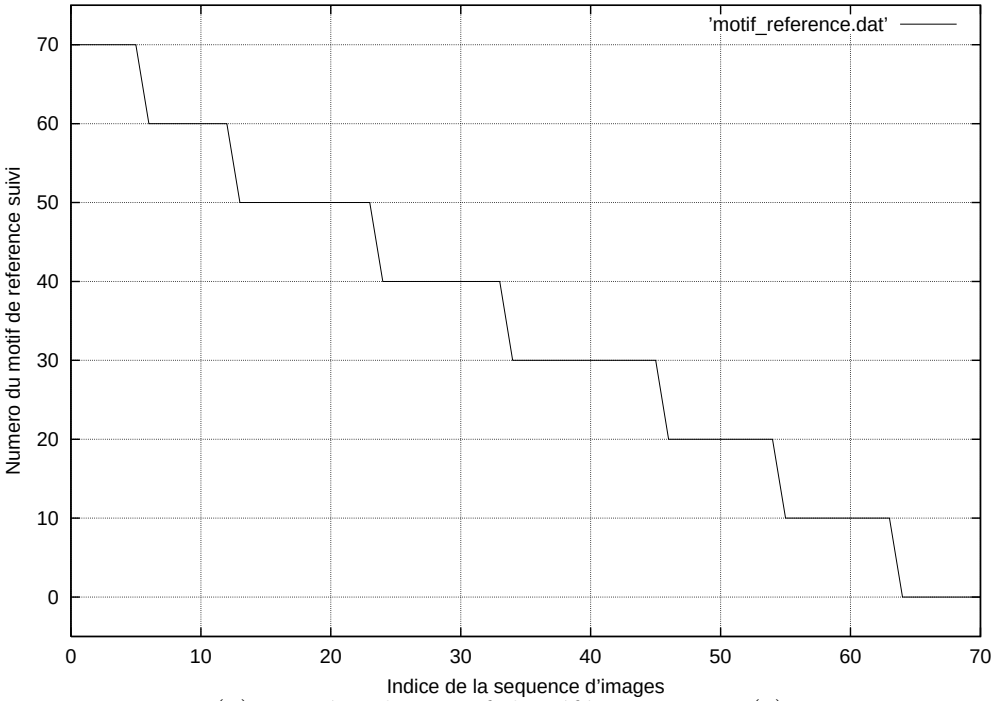


(a) numéro du motif de référence suivi (c).

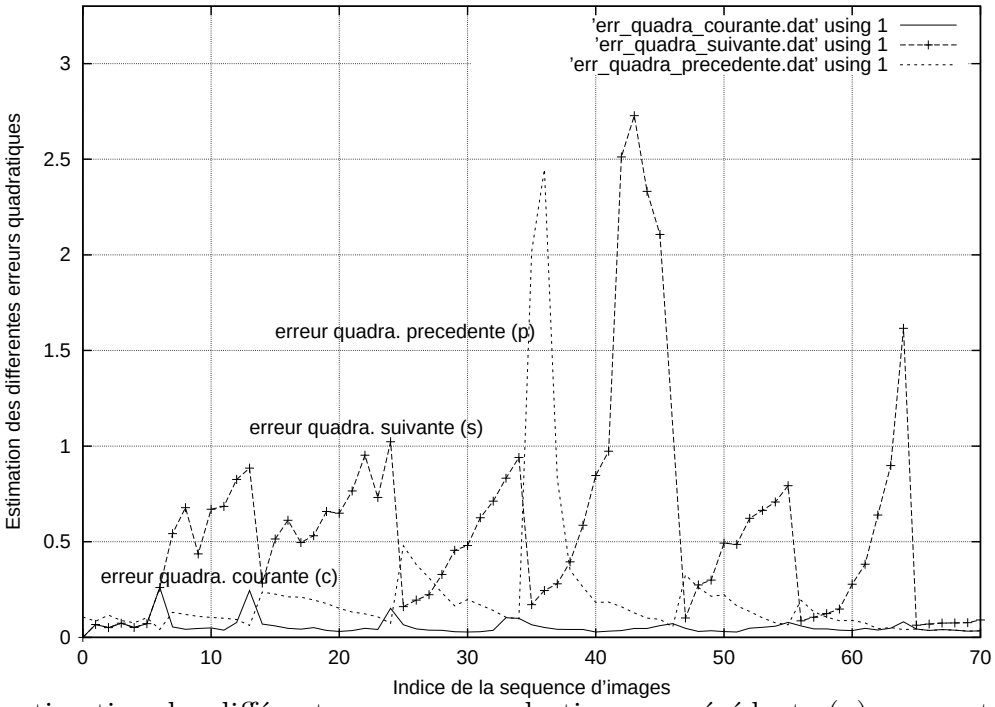


(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.10 – résultats de l'algorithme de suivi sur l'ensemble des 71 vues de la base d'apprentissage (de la première image vers la dernière image).



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.11 – résultats de l'algorithme de suivi sur l'ensemble des 71 vues de la base d'apprentissage (de la dernière image vers la première image).

Ces premiers résultats nous ont vivement encouragé à tester notre algorithme de suivi 3D en situation réelle.

Expérimentation de l'algorithme de suivi et compression des données

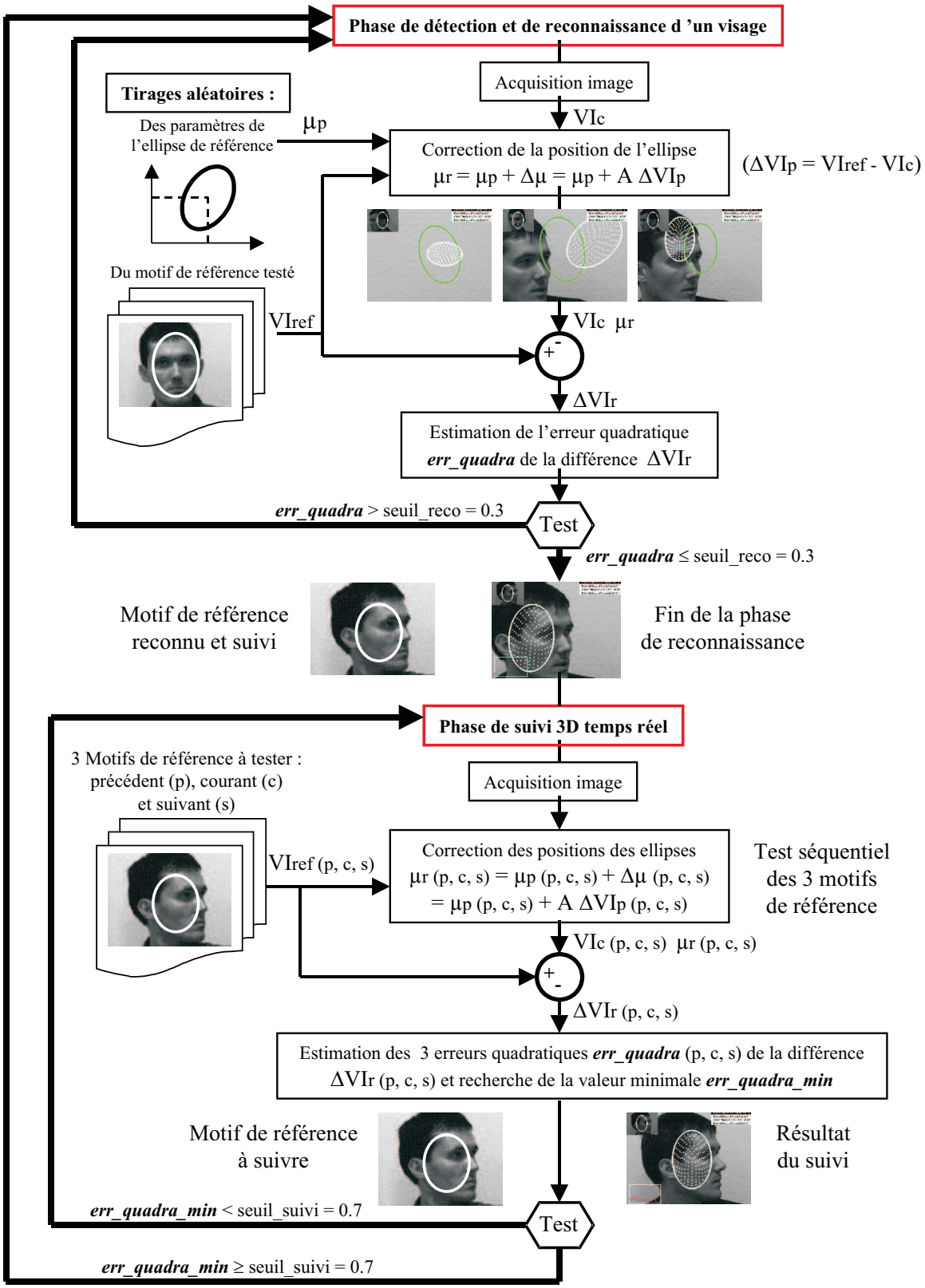


FIGURE 4.12 – présentation de l'application automatique de suivi 3D d'un visage.

Notre premier souhait était de réaliser une application entièrement autonome c'est à dire que l'algorithme de suivi doit permettre une détection automatique d'un visage dans l'image

pour en assurer son suivi 3D et être capable de revenir en mode d'initialisation en cas de perte de suivi en ligne du motif. Le principe de cette application de suivi 3D est illustré figure 4.12.

Pour réaliser cette initialisation automatique, nous tirons aléatoirement un motif de référence dans la base d'apprentissage et une position de l'ellipse dans l'image courante. Nous cherchons alors à corriger cette position d'ellipse μ_p pour essayer de se recaler sur le motif du visage détecté μ_r et ceci, tant que l'erreur quadratique *err-quadra* de la différence de niveaux de gris ΔVI_r entre le motif de référence testé VI_{ref} et le motif courant échantillonné VI_c dans l'ellipse après correction reste supérieure à un seuil *seuil-reco* égal à 0.30.

La détection de la perte de suivi d'un motif se fait également en comparant l'erreur quadratique la plus faible en sortie du processus de suivi à un seuil *seuil-suivi* égal à 0.70. Tant que cette erreur reste inférieure à ce seuil, la phase de suivi en ligne se poursuit sinon une nouvelle phase d'initialisation de suivi commence.

Les valeurs de ces différents seuils n'ont pas été déduites des résultats obtenus à partir de la simulation mais plutôt expérimentalement pour pouvoir tenir compte des variations d'éclairement suivant la pose du visage.

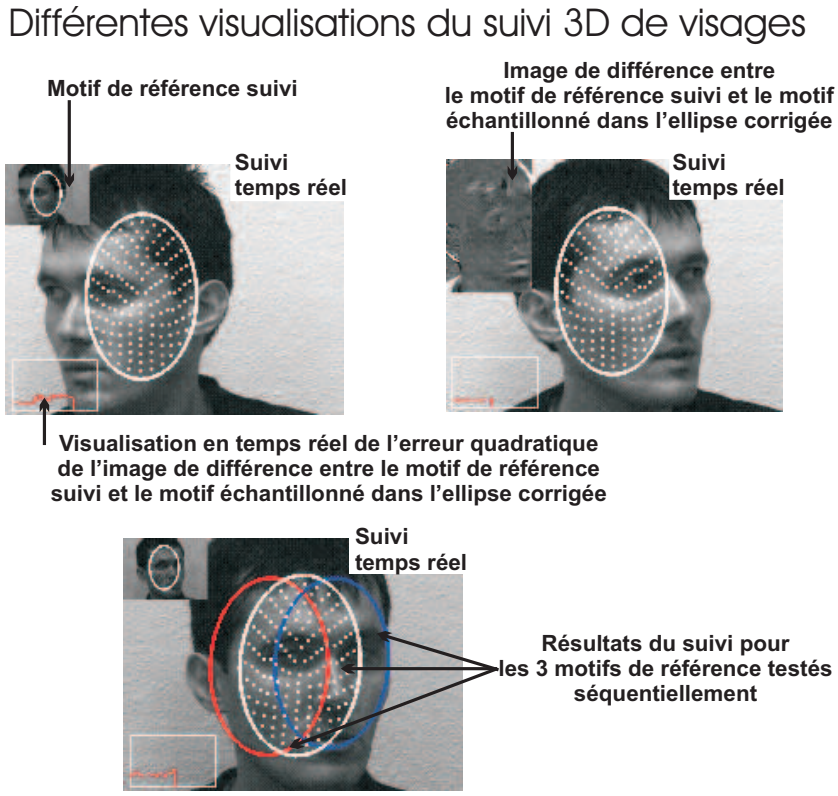


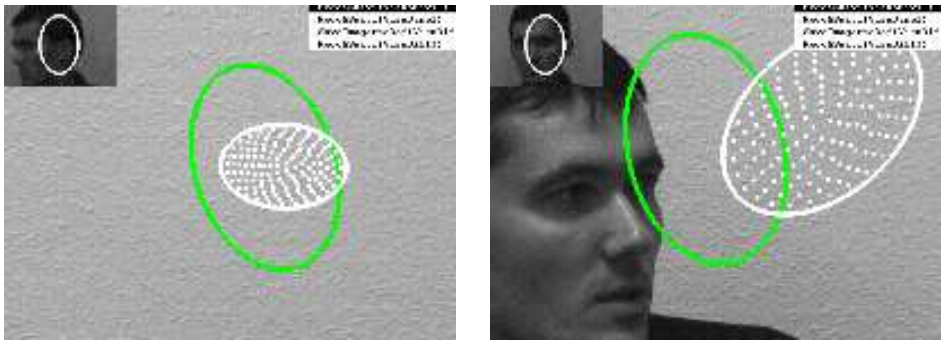
FIGURE 4.13 – visualisation des différents résultats issus de l'algorithme de suivi.

Cet algorithme de suivi 3D de visages nous fournit en temps réel trois informations (figure 4.13) :

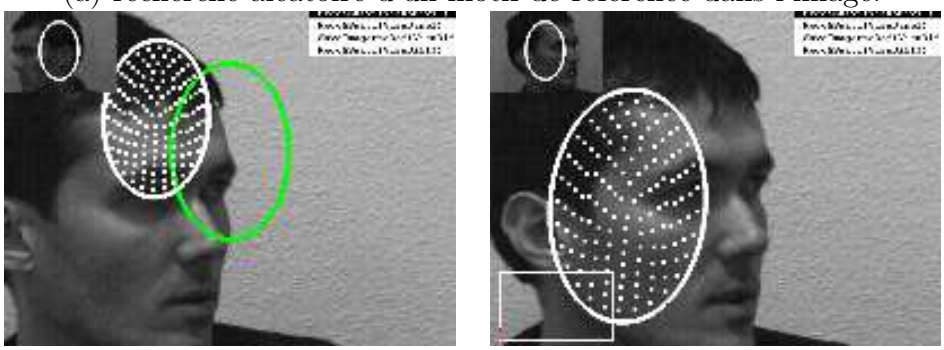
1. le motif de référence actuellement suivi représenté par une imagette dans le coin supérieur gauche de la fenêtre de visualisation ou l'image de différence de niveaux de gris entre le motif de référence suivi et le motif échantillonné dans l'ellipse courante après correction,
2. la mesure de l'erreur quadratique de la différence de niveaux de gris résultante du suivi à l'aide d'un graphique ajouté en surimpression sur l'image courante (coin inférieur gauche),
3. le résultat du suivi pour chacun des motifs de référence à l'intérieur d'une ellipse de couleur différente (ellipses rouge (motif précédent), verte (motif courant), bleue (motif suivant) et blanche (motif de référence donnant l'erreur quadratique la plus faible et qui sera suivi dans la prochaine image)).

Les illustrations présentées maintenant montrent les résultats obtenus à des instants différents de l'expérience :

– *la phase d'initialisation du processus* : l'algorithme recherche de manière aléatoire la position d'un motif de référence dans l'image et ceci tant qu'un des motifs de référence testés n'est pas reconnu (figure 4.14). En moyenne, nous réalisons quinze tirages aléatoires pour un temps d'initialisation inférieur à la seconde.



(a) recherche aléatoire d'un motif de référence dans l'image.



(b) détection et correction de l'ellipse courante avec le motif de référence testé.

FIGURE 4.14 – détection et reconnaissance d'un motif de référence dans la base d'apprentissage.

– *la phase de suivi en ligne* : les images retenues ont été sélectionnées de manière à montrer la robustesse de la méthode aux variations d'éclairement, de position du visage par rapport à la caméra et de ses expressions (figures 4.15, 4.16 et 4.17).

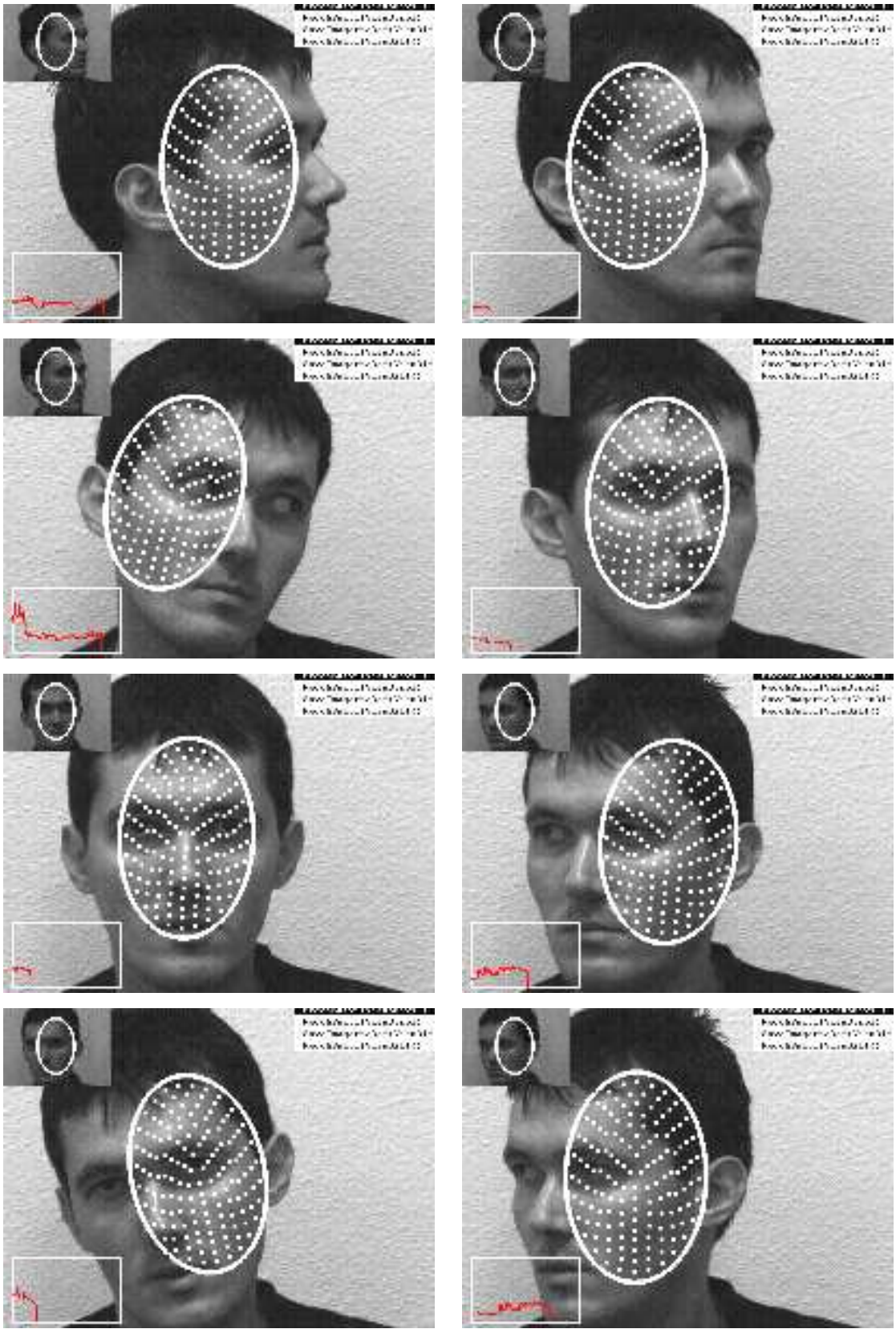


FIGURE 4.15 – suivi 3D d'un visage : séquence 1.

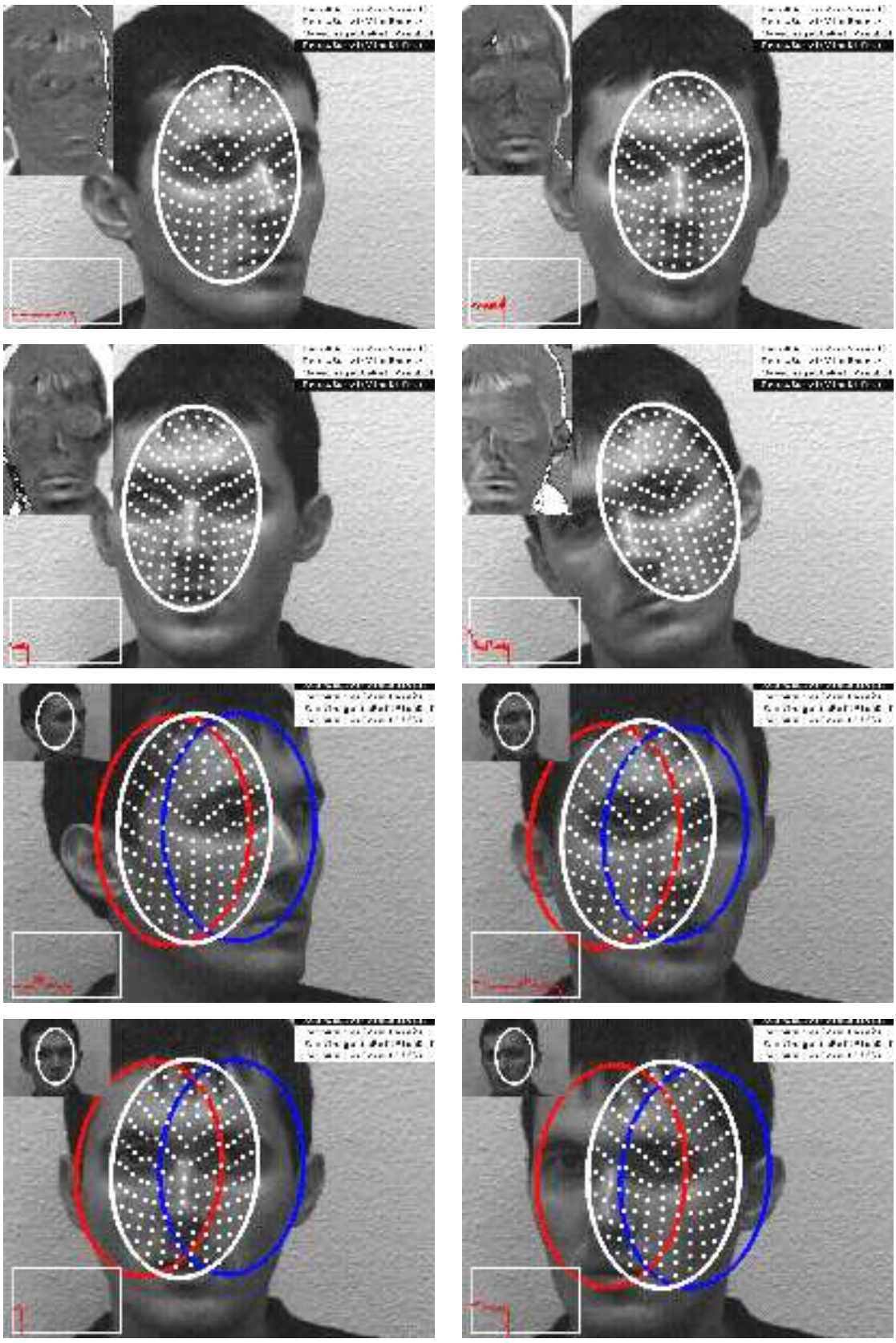


FIGURE 4.16 – suivi 3D d'un visage : séquence 2.

Les premiers essais réels de suivi 3D d'un visage ont donné des résultats plutôt encourageants. Le passage d'un motif de référence à un autre s'effectue toujours correctement. Les variations d'éclairement de la scène ainsi que certains mouvements du visage devant la caméra

(en particulier pour des variations en azimut) augmentent la valeur finale de l'erreur quadratique de la différence de niveaux de gris sans pour autant entraîner la perte de suivi du motif courant dans l'image. C'est pourquoi, la valeur du seuil détectant la perte de suivi n'a pas été définie d'après les résultats de la simulation mais en fonction des conditions réelles d'expérimentation.

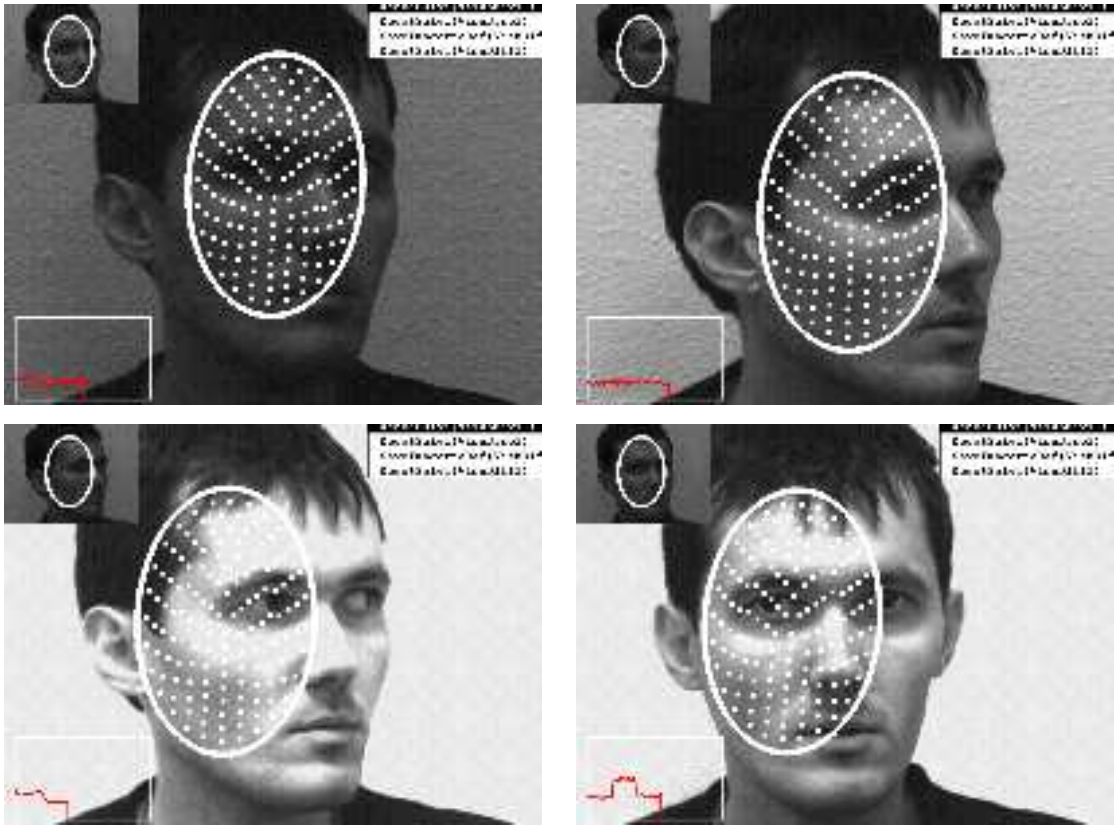


FIGURE 4.17 – comportement de l'algorithme aux variations d'éclairement.

Pour améliorer les performances de notre algorithme de suivi, il serait intéressant de définir une zone d'intérêt par image de référence et ne pas garder qu'une seule ellipse pour l'ensemble de ces vues. Ceci permettrait de sélectionner un motif de référence de manière plus rigoureuse. Une autre amélioration possible serait une méthode indépendante de la méthode retenue qui permettrait à partir d'une collection d'images 2D de choisir les meilleures vues de référence pour modéliser au mieux un visage ou un objet 3D.

Dans cette application de suivi 3D de visages, nous nous sommes également intéressés au problème de la compression des informations qui seraient transmises dans le cas de la visio-conférence par exemple. En particulier, pour diminuer la taille des données utilisées, il serait intéressant de transmettre soit le motif suivi à l'intérieur de la zone d'intérêt soit l'image de différence entre le motif de référence et le motif courant échantillonné après correction pour reconstruire l'expression faciale de l'individu à partir de la collection d'images de référence

qui aurait été transmise à l'interlocuteur à l'initialisation de l'application (lors de la première connexion).

Les tableaux ci-dessous nous fournissent, à titre indicatif, les différentes tailles des fichiers de sortie (au format .avi) construits à partir d'une collection de 71 images (au format .gif) avec ou sans compression des images. La taille originale de chaque image est de 120x160 pixels. Dans le premier tableau, l'image représente le motif courant à l'intérieur de la zone d'intérêt après correction de sa position. Dans le second tableau, elle représente l'image de différence entre le motif de référence actuellement suivi et le motif courant échantillonné dans l'ellipse dont la position a été corrigée. De plus, les taux utilisés pour compresser l'information ne doivent pas dégrader de manière importante la qualité du rendu de l'image à l'interlocuteur dans le cas de la visioconférence (figure 4.18). Le taux de compression des données associé à un fichier est calculé en divisant sa taille par la taille du fichier original non compressé.

Paramètres de sortie vidéo	Qualité / Taux de compression (%)	Taille du fichier de sortie (bytes)	Taille moyenne par image (bytes)
Pas de compression		5 454 832	76 829
Compression JPEG	1 (Most) / 13.29	724 800	10 208
	0.75 (High) / 3.35	182 712	2 573
	0.50 (Normal) / 2.49	135 880	1 914

Paramètres de sortie vidéo	Qualité / Taux de compression (%)	Taille du fichier de sortie (bytes)	Taille moyenne par image (bytes)
Pas de compression		5 454 832	76 829
Compression JPEG	1 (Most) / 13.99	763 160	10 748
	0.75 (High) / 3.28	178 880	2 519
	0.50 (Normal) / 2.35	128 088	1 804

Les différentes images utilisées pour construire le fichier vidéo de sortie sont les résultats fournis par l'algorithme de suivi sur l'ensemble des 71 images constituant la base d'apprentissage lors du test en simulation.

1 0.75 0.50 0.25 0

FIGURE 4.18 – images compressées pour différentes qualités de rendu.

En comparant les taux de compression pour une même qualité, nous nous apercevons que l'image de différence fournit autant d'information dans son contenu que l'image courante. Nous pouvons donc aussi bien transmettre à l'interlocuteur l'une des deux images. La meilleure solution bien entendu est de transmettre directement la zone de l'image courante définie autour de la position corrigée de l'ellipse pour éviter tout traitement de reconstruction du visage après réception.

4.2 Commande en position "plus ou moins" de la table à déplacement micrométrique

Dans cette section, nous souhaitons utiliser notre algorithme de suivi 3D pour contrôler les mouvements de la table à déplacement micrométrique, en particulier, sa position angulaire en site. Cette application a été menée pour tester les possibilités d'intégration de notre solution de suivi 3D dans une boucle d'asservissement visuel avant de poursuivre nos développements sur le robot portique du laboratoire. Pour présenter nos résultats, la base d'apprentissage modélisant l'apparence de l'objet 3D (la figurine) comprend 181 images numérotées de 10 à 190 (soit un total de 19 vues de référence dont l'indice est un multiple de 10). Ces images sont acquises tous les degrés pour une variation en site de 180 degrés (figure 4.19). Chaque motif de référence est défini à partir d'une ellipse différente sélectionnée manuellement par l'opérateur.

Dans les paragraphes suivants, nous présentons la commande en position que nous avons développée avant de tester en simulation puis en condition réelle ses performances qui reposent principalement sur la fiabilité et la robustesse de notre algorithme de suivi 3D. Pour cette application, la caméra est positionnée de façon que l'objet posé sur la table à déplacement micrométrique reste centré dans l'image.

objet 3D à modéliser | vue de référence 10 | vue de référence 100 | vue de référence 190

FIGURE 4.19 – modélisation de la figurine pour une variation en site de 180 degrés.

4.2.1 Solution retenue pour la commande en position "plus ou moins"

Pour initialiser le processus, l'algorithme de suivi 3D détecte l'apparition d'un motif à l'intérieur de la zone d'intérêt (l'ellipse courante) et recherche, dans la base d'apprentissage, le motif de référence donnant la plus faible erreur quadratique de la différence de niveaux de gris entre son vecteur de forme et le motif échantillonné dans l'ellipse après correction de sa position dans l'image. Le motif courant d'indice *num_motif_suivi* étant maintenant reconnu, nous déterminons une consigne S^* définie comme le nouveau motif de référence à atteindre d'indice *num_motif_atteindre*. La phase d'initialisation terminée, nous pouvons asservir la position angulaire en site de la table dont le principe est illustré ici par la figure 4.20.

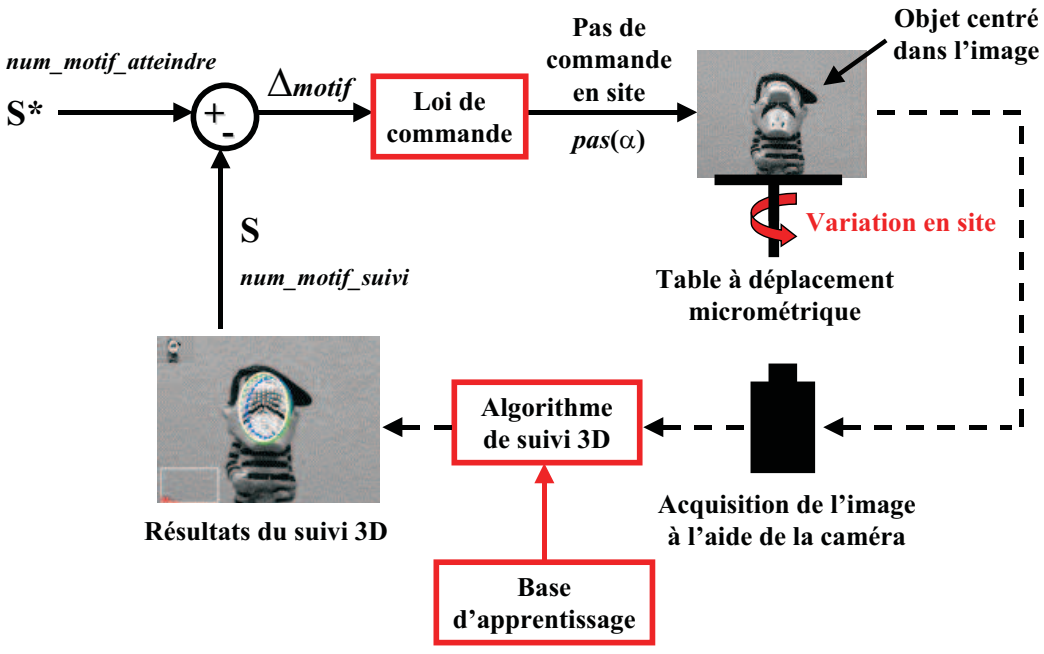


FIGURE 4.20 – commande en position "plus ou moins" de la table à déplacement micrométrique.

Pour chaque nouvelle acquisition d'image, la *loi de commande en position* élabore le pas de commande en site $pas(\alpha)$ en comparant la consigne S^* (le motif de référence à atteindre) avec la mesure courante S (le motif de référence actuellement suivi) issue de notre algorithme de suivi 3D. Ce pas de commande appliqué à la table correspond à la variation en site entre deux images consécutives de la base d'apprentissage soit 1 degré. Lorsque la différence $\Delta motif$ entre les deux indices des motifs de référence devient nulle, la consigne est atteinte et le déplacement de la table est stoppé.

En posant $\Delta motif$ = *num_motif_atteindre* - *num_motif_suivi*, cette loi de commande en position "plus ou moins" peut être décrite de la manière suivante :

1. si $\Delta motif > 0$ alors $pas(\alpha) = +1$ deg / position courante (le motif courant se situe avant le motif à atteindre dans la collection d'images de référence),
2. si $\Delta motif < 0$ alors $pas(\alpha) =$ -1 deg / position courante (le motif courant se situe après le motif à atteindre dans la collection d'images de référence),
3. si $\Delta motif = 0$ alors $pas(\alpha) = 0$ deg / position courante (le motif courant correspondant au motif à atteindre, on arrête le déplacement de la table).

Bien entendu, au cours de l'asservissement visuel, l'échelle et l'orientation du motif dans l'image courante peuvent changer. Par contre, sa position doit rester autour du centre de l'image pour éviter de perdre son suivi (la caméra restant fixe devant l'objet au cours du déplacement en site de la table (figure 4.21)).

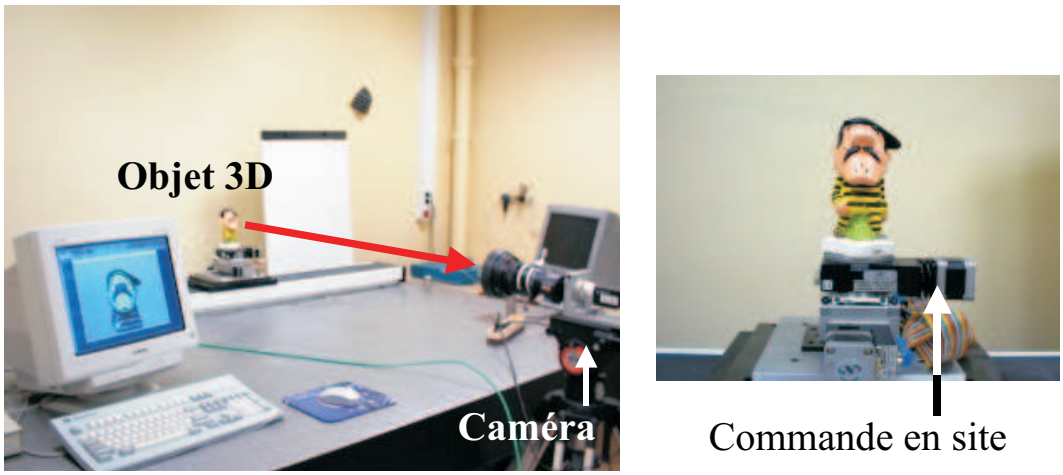
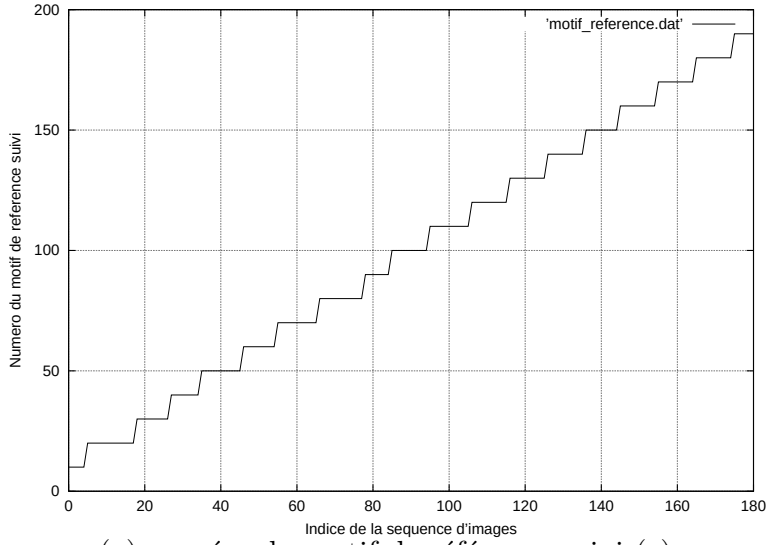


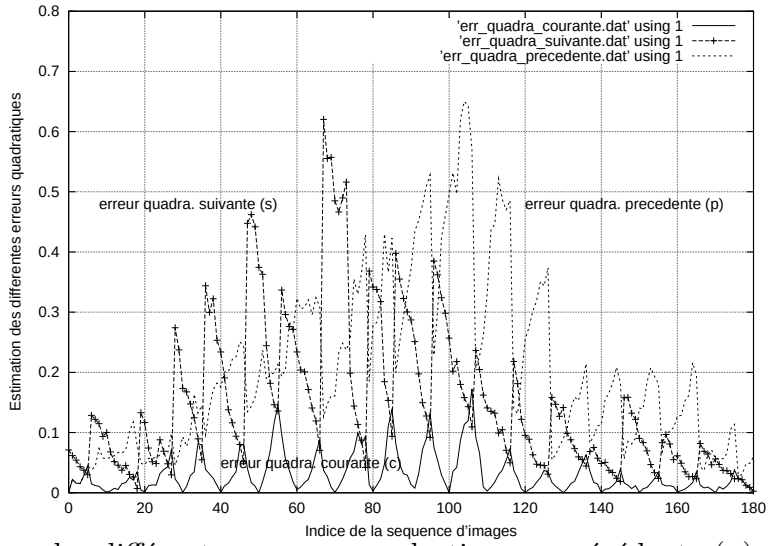
FIGURE 4.21 – position caméra/objet lors de la commande en site de la table.

4.2.2 Simulation de l'algorithme de suivi 3D pour la commande en position "plus ou moins"

Comme les performances de l'asservissement visuel en position dépendent principalement des résultats issus de notre algorithme de suivi 3D, nous simulons le comportement de notre approche sur l'ensemble des images de la base d'apprentissage. Ainsi le passage entre deux images consécutives de la base, correspondant à une variation en site de 1 degré de l'objet par rapport à la caméra, sera équivalent au pas de commande défini par la loi de commande en position pour contrôler le déplacement de la table.



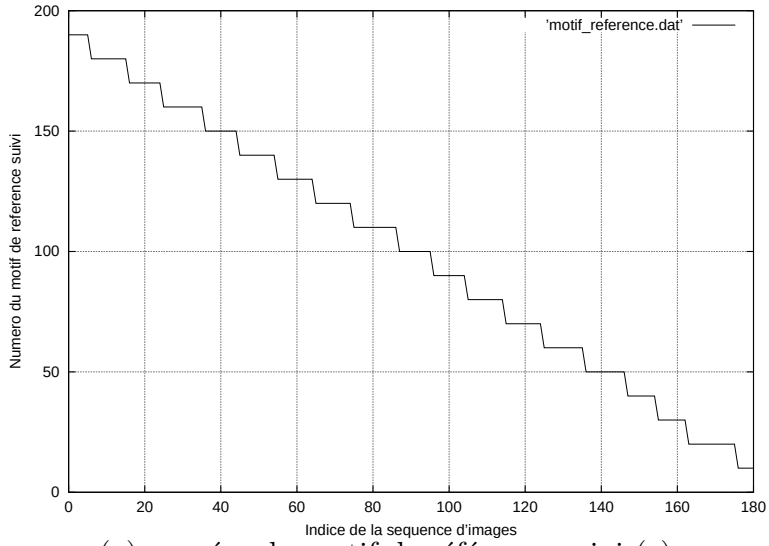
(a) numéro du motif de référence suivi (c).



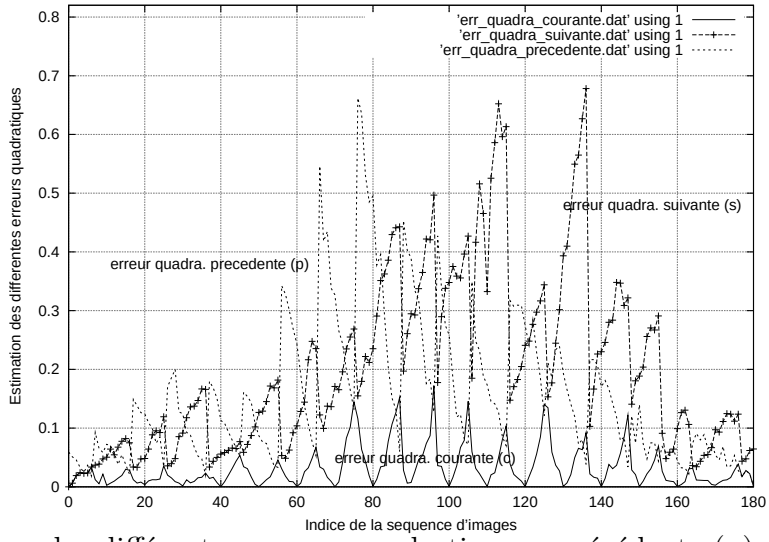
(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.22 – résultats de l'algorithme de suivi sur l'ensemble des 181 vues de la base d'apprentissage (de la première image vers la dernière image).

Pour chacune de ces images, nous enregistrons l'erreur quadratique de la différence de niveaux de gris associée à chacun des motifs de référence testés après correction des paramètres de l'ellipse ainsi que l'indice du nouveau motif de référence à suivre dans la prochaine image.



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.23 – résultats de l'algorithme de suivi sur l'ensemble des 181 vues de la base d'apprentissage (de la dernière image vers la première image).

Les figures 4.22 et 4.23 montrent les résultats obtenus au cours de la simulation sur l'ensemble de la collection d'images. L'initialisation de l'algorithme de suivi est réalisée soit sur la première image de la base d'apprentissage (vue de référence 10) soit sur la dernière (vue de référence 190).

Comme nous utilisons le même algorithme de suivi pour l'ensemble des applications présentées dans ce chapitre, les résultats de simulation sont similaires à ceux observés pour le suivi de visages. Nous constatons que l'enchaînement des différents motifs de référence s'effectue régulièrement par paliers successifs. Ceci s'explique par le fait que la variation de pose de l'objet entre deux images consécutives de la base reste identique (variation en site de 1 degré). Notons cependant que pour une même variation en site, les variations d'apparence du motif dans l'image courante peuvent être plus ou moins importantes selon que la figurine se trouve de profil ou de face. D'une manière générale, le changement de motif de référence s'effectue pour des images intermédiaires situées à *"mi-chemin"* des motifs de référence les plus proches.

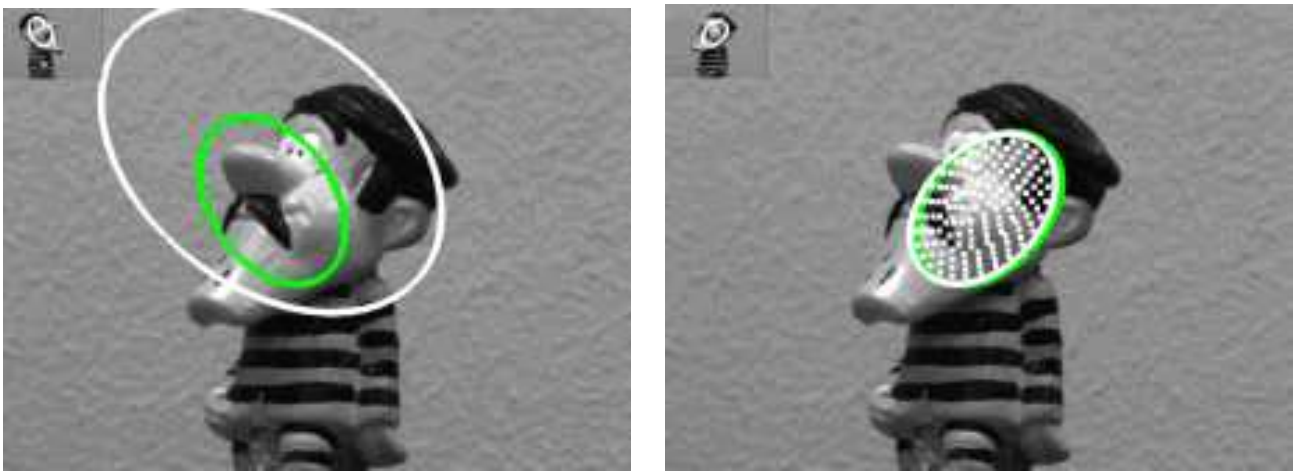
A la différence du suivi de visages, les valeurs des trois erreurs quadratiques sont plus faibles. Ici, les erreurs quadratiques précédente (p) et suivante (s) restent inférieures à 0.70 alors que dans la première application, elles pouvaient atteindre la valeur 2.70. Par contre, la valeur de l'erreur quadratique du motif de référence suivi (c) reste dans le même ordre de grandeur (valeurs inférieures à 0.26 pour la première et à 0.18 pour la seconde). Cette variation d'amplitude observée pour chacune des grandeurs estimées peut se justifier selon plusieurs points :

1. la nature de l'objet 3D (ses propriétés intrinsèques : apparence et réflectance),
2. la méthode d'acquisition de la base d'apprentissage,
3. le choix des motifs de référence,
4. le choix des mouvements autorisés lors de la phase de suivi,
5. les conditions d'éclairement (éclairages artificiel et/ou naturel) qui peuvent varier entre l'instant où l'on enregistre la base d'apprentissage et le moment où l'on assure le suivi en ligne.

Ces différents résultats nous ont permis de vérifier que nous étions capables d'assurer le suivi sans ambiguïté sur l'ensemble de la séquence d'images. Nous pourrons, par conséquent, initialiser correctement notre application quel que soit le motif détecté dans l'image courante et donner par la suite des informations fiables à la loi de commande pour contrôler le déplacement de la table en condition réelle d'expérimentation.

4.2.3 Expérimentation de l'algorithme de commande en position "plus ou moins" de la table à déplacement micrométrique

L'objet étant posé sur la table devant la caméra, l'algorithme de détection automatique d'un motif dans l'image est exécuté pour initialiser l'application (figure 4.24). Pour cela, nous tirons aléatoirement un motif de référence dans la base d'apprentissage et une position de l'ellipse dans l'image courante. Nous cherchons alors à corriger les paramètres de cette ellipse pour essayer de se recaler sur le motif détecté et ceci, tant que l'erreur quadratique entre le motif de référence testé et le motif courant échantillonné dans l'ellipse corrigée reste supérieure à un seuil de valeur 0.40. En moyenne, nous réalisons vingt tirages aléatoires pour initialiser l'application (soit un temps total d'exécution de l'ordre de la seconde).



(a) recherche aléatoire d'un motif dans l'image.

(b) détection et reconnaissance du motif dans l'image.

FIGURE 4.24 – phase de détection automatique d'un motif dans l'image.

Le motif étant maintenant reconnu ($num\_motif\_suivi = 150$ dans notre exemple), nous tirons aléatoirement un motif de référence que nous souhaitons atteindre en contrôlant le déplacement angulaire en site de la table ($num\_motif\_atteindre = 70$).

L'initialisation étant terminée, l'algorithme de suivi 3D détermine le motif de référence suivi dans l'image courante après chaque déplacement de la table. Cette information est ensuite utilisée par la loi de commande qui élabore le pas de commande $pas(\alpha)$ à envoyer à la table pour son prochain déplacement et ceci, tant que le motif de référence suivi d'indice num_motif_suivi ne correspond pas au motif de référence à atteindre d'indice $num\_motif\_atteindre$.

Notons qu'entre deux acquisitions d'image successives, la table se déplace d'un pas de 1 degré en site par rapport à sa position courante correspondant à la variation de position de l'objet 3D devant la caméra entre deux images consécutives de la base d'apprentissage.

Cet algorithme, développé sur une station de travail Silicon Graphics O_2, a un temps d'exécution inférieur à 20 ms. Il fournit en temps réel les informations associées à l'erreur quadratique courante, le motif de référence suivi et la position de l'ellipse pour chacun des trois motifs de référence testés (figure 4.25).

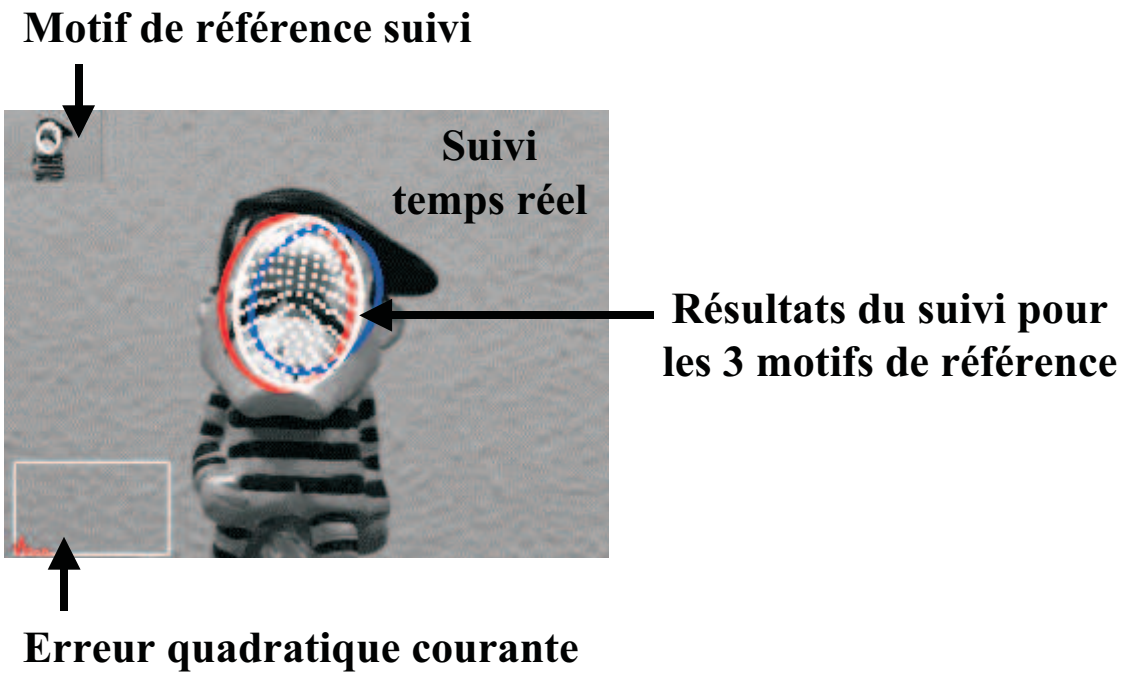


FIGURE 4.25 – visualisation des résultats lors de la commande en position de la table.

La figure 4.26 illustre les résultats de notre algorithme obtenus à des instants différents de l'expérimentation. Au cours de l'asservissement visuel, nous sauvegardons, après chaque nouvelle acquisition d'image, les valeurs des différentes erreurs quadratiques associées aux trois motifs de référence testés séquentiellement et l'indice du motif de référence à suivre dans la prochaine image.

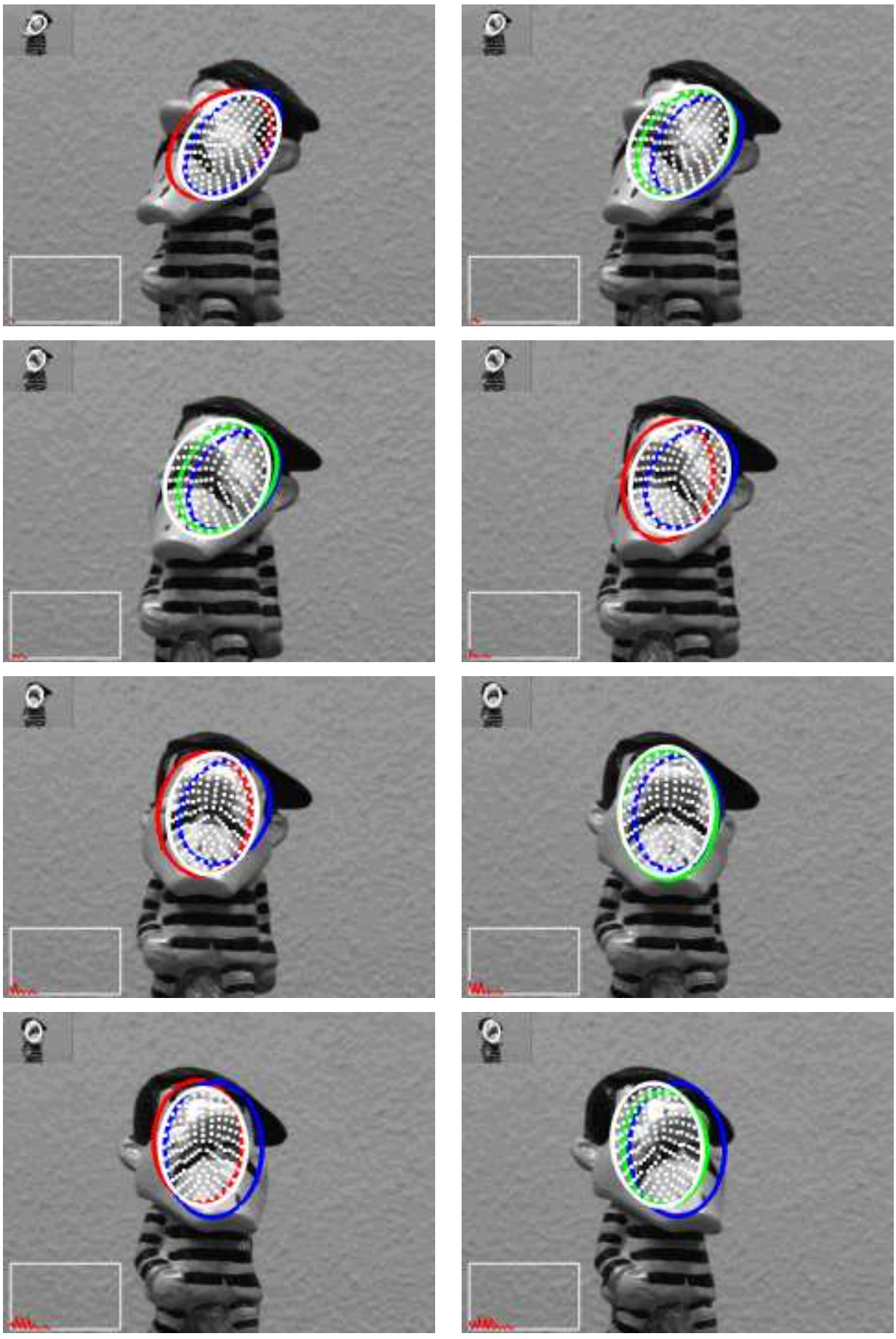
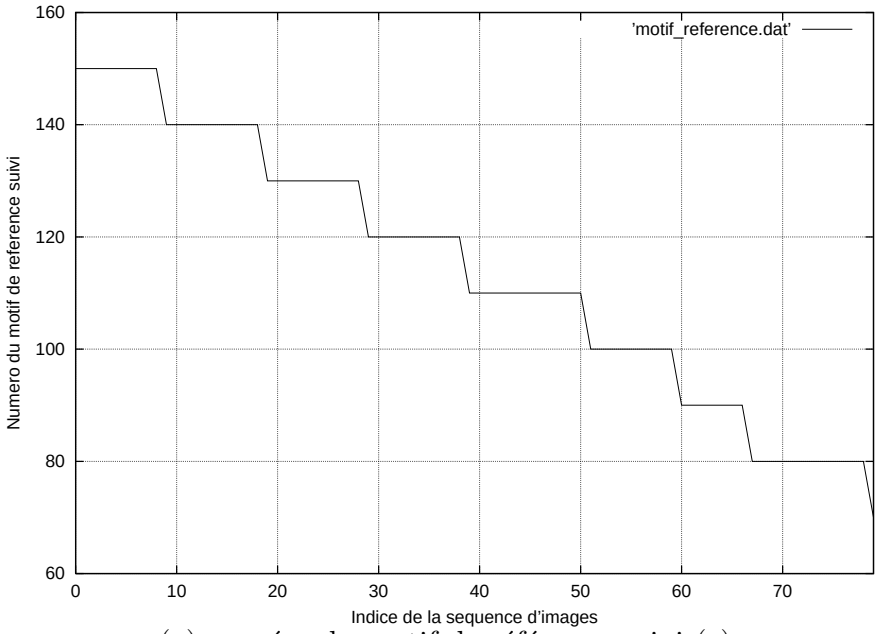
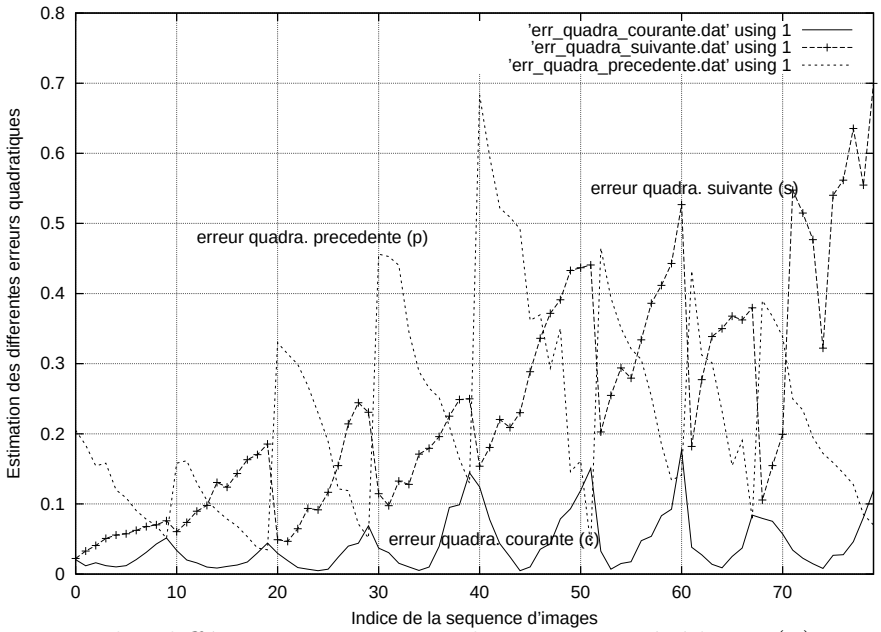


FIGURE 4.26 – résultats de la commande en position de la table à déplacement micrométrique.

Ces différents résultats présentés sur la figure 4.27 peuvent être comparés avec ceux obtenus lors de la simulation (figures 4.22 et 4.23).



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.27 – résultats de l'algorithme de suivi pour la commande en position de la table lors de l'expérimentation.

Nous remarquons que les changements de motif de référence s'opèrent toujours régulièrement et que les valeurs des différentes erreurs quadratiques entre la pratique et la simulation sont

du même ordre de grandeur. En particulier, l'erreur quadratique courante (c) correspondant au motif de référence suivi reste inférieure à 0.18. Ceci s'explique par le fait que nous avons un éclairage artificiel constant de la scène et que la position de l'objet devant la caméra est proche de celle définie lors de l'acquisition de la base d'apprentissage. De plus, les mouvements élaborés par la loi de commande (variation en site) sont parfaitement modélisés par la base d'apprentissage et détectés par l'algorithme de suivi. Le choix de conserver le même pas de commande pour l'acquisition de la collection d'images et pour la phase d'asservissement permet de garantir une fiabilité dans le contrôle de la table.

Ces résultats plutôt prometteurs nous ont décidé à développer cette commande en position sur le robot portique du laboratoire. Pour cette nouvelle application, ce n'est plus l'objet qui se déplace devant la caméra mais la caméra qui évolue autour de l'objet.

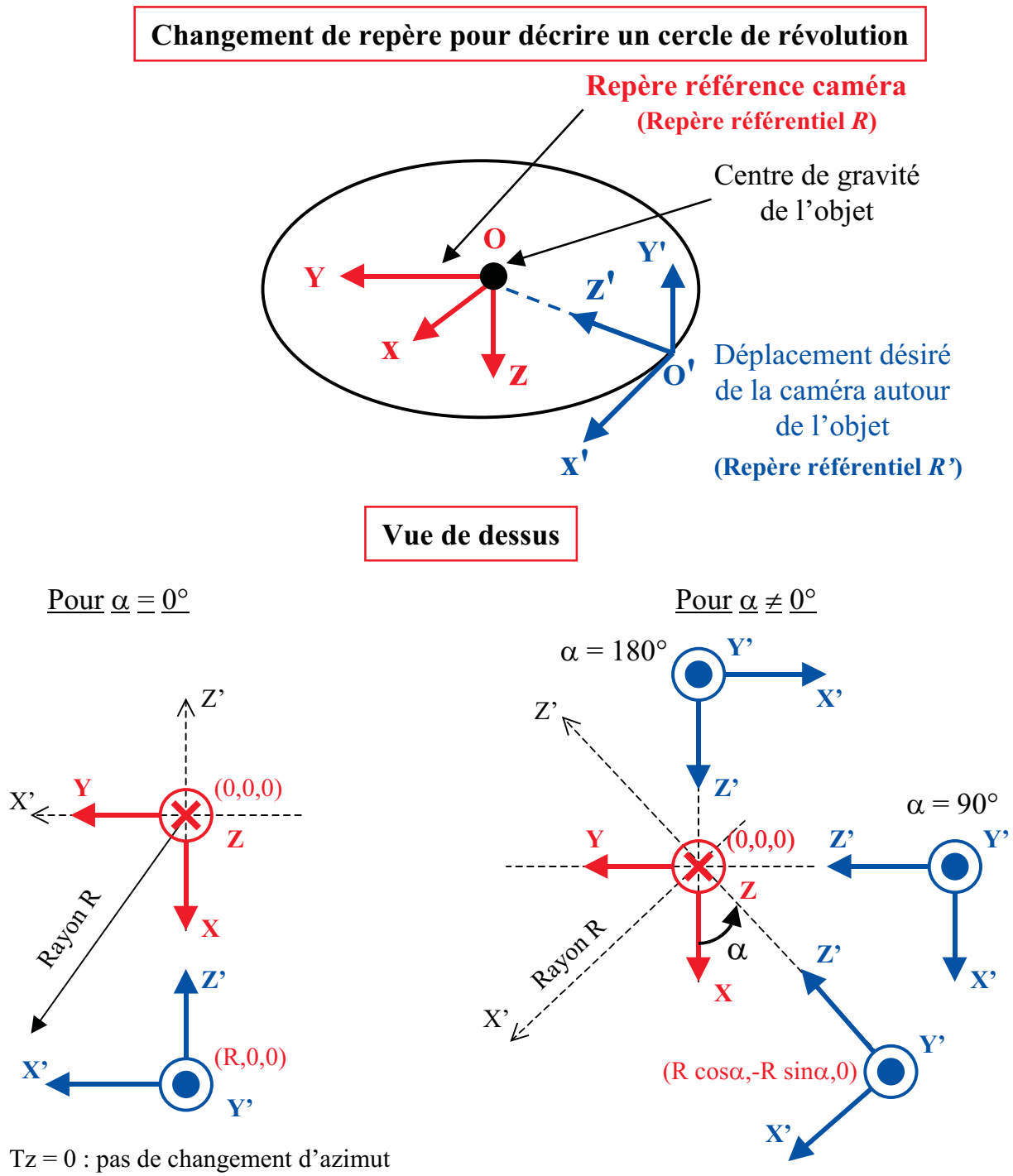
4.3 Commande en position "plus ou moins" d'un bras robotique autour d'un objet connu

4.3.1 Présentation générale de l'application

Dans cette section, nous présentons une application robotique qui consiste à faire naviguer un bras manipulateur équipé d'une caméra sur son effecteur autour d'un objet connu (ici une figurine).

L'algorithme de suivi 3D associé à la loi de commande en position "plus ou moins" nous permet de commander ce bras robotique en reproduisant la trajectoire définie lors de l'acquisition de la base d'apprentissage (figure 4.29). Cette trajectoire circulaire peut être décrite à l'aide d'une *matrice homogène* M permettant d'exprimer les coordonnées d'un point de la trajectoire du bras robotique définies dans le référentiel $R' = (O', X', Y', Z')$ dans le repère référence caméra $R = (O, X, Y, Z)$ (figure 4.28).

Cette matrice homogène M (voir annexe C sur les *notions de robotique*) se décompose en deux matrices : une matrice de rotation R de dimensions (3×3) et une matrice de translation T de dimensions (3×1) qui peuvent s'écrire de la manière suivante :



Rotation :

$$\begin{cases} X' = Y \\ Y' = -Z \\ Z' = -X \end{cases}$$

Translation :

$$\begin{cases} Tx = R \\ Ty = 0 \\ Tz = 0 \end{cases}$$

Rotation :

$$\begin{cases} X' = X\sin\alpha + Y\cos\alpha \\ Y' = -Z \\ Z' = -X\cos\alpha + Y\sin\alpha \end{cases}$$

Translation :

$$\begin{cases} Tx = R\cos\alpha \\ Ty = -R\sin\alpha \\ Tz = 0 \end{cases}$$

FIGURE 4.28 – définition de la trajectoire du bras robotique autour de l'objet 3D.

$$\mathbf{R} = \begin{pmatrix} \sin\alpha & 0 & -\cos\alpha \\ \cos\alpha & 0 & \sin\alpha \\ 0 & -1 & 0 \end{pmatrix} \quad \mathbf{T} = \begin{pmatrix} R\cos\alpha \\ -R\sin\alpha \\ 0 \end{pmatrix} \tag{4.6}$$

Ici $Tz = 0$ car nous supposons aucune variation en azimut. Finalement, nous écrivons la matrice M sous la forme :

$$\mathbf{M} = \begin{pmatrix} R_{3\times3} & T_{3\times1} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \sin\alpha & 0 & -\cos\alpha & R\cos\alpha \\ \cos\alpha & 0 & \sin\alpha & -R\sin\alpha \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

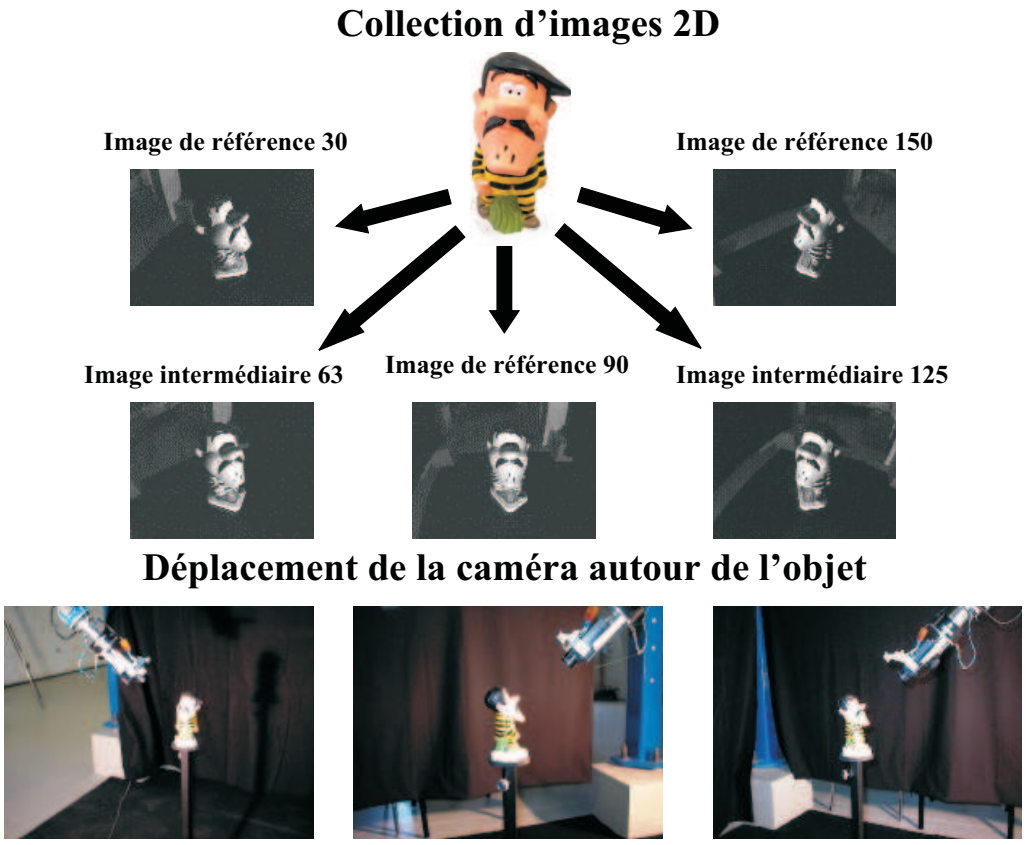


FIGURE 4.29 – acquisition de la collection d'images 2D à l'aide du bras robotique.

La base d'apprentissage modélisant l'objet 3D (la figurine) comprend au total 121 images (soit un total de 13 images de référence dont l'indice est un multiple de 10) (figure 4.29). Chaque motif de référence sera défini à partir d'une ellipse différente sélectionnée par l'opérateur. Durant cette phase d'acquisition où l'objet doit rester centré dans l'image, le bras robotique décrit une trajectoire circulaire pour laquelle l'angle de site varie de 30 à 150 degrés par pas de 1

degré (d'après les conventions prises pour le changement de repère illustré figure 4.28).

Durant la construction de la base d'apprentissage, nous ne pouvons pas utiliser l'éclairage embarqué sur l'effecteur car la distance objet/caméra est trop faible (de l'ordre d'une vingtaine de centimètres). Les réflexions de la surface de l'objet entraînaient des saturations trop importantes de la caméra. Pour éclairer notre objet, nous utilisons une source de lumière artificielle située derrière le bras manipulateur, à hauteur de figurine et à l'extérieur de la zone de déplacement du robot. Cette source de lumière nous rend moins sensible aux variations d'illumination naturelle de la scène (figure 4.30).

FIGURE 4.30 – éclairage artificiel de la scène.

Avant de développer plus précisément notre approche qui est une adaptation de la commande en position de la table à déplacement micrométrique, nous allons rappeler quelques généralités sur la robotique utilisant la vision artificielle et plus particulièrement la commande d'un bras manipulateur par vision. Des compléments d'information sont donnés dans l'annexe C.

4.3.2 Commande d'un bras manipulateur par vision

Comme les tâches à réaliser par les robots devenaient chaque jour plus complexes, les chercheurs ont voulu rendre ceux-ci "versatiles", pour qu'ils soient encore plus autonomes. Cela implique que le robot soit capable de *percevoir* son environnement afin de l'analyser. L'utilisation de la caméra vidéo s'est vite révélée comme un des moyens de perception artificielle les

plus appropriés.

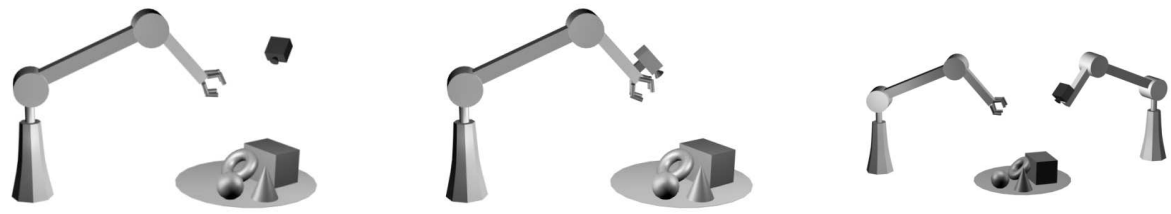


FIGURE 4.31 – trois types de configuration de la position de la caméra.

Il existe trois dispositions possibles de la caméra par rapport au robot (figure 4.31) :

1. *Caméra fixe observant la scène de l'extérieur* : le comportement du robot n'influe pas sur la position de la caméra qui garde toujours le même point de vue.
2. *Caméra fixée sur le manipulateur* : elle est en général fixée sur l'effecteur du robot (dernière articulation). Cette configuration porte dans la littérature robotique l'appélation de *"eye-in-hand"* (œil dans la main). Cette fois, les mouvements du robot agissent directement sur le point de vue de la caméra. C'est cette disposition qui est utilisée dans le cadre de notre étude.
3. *Caméra fixée sur un autre manipulateur* : on retrouve la notion d'observation extérieure pour avoir une vision globale de la scène. L'avantage supplémentaire est de pouvoir bouger la caméra indépendamment du robot principal pour obtenir un point de vue différent ou une vision locale d'un détail de la scène [106][108].

Ces configurations peuvent être combinées pour des commandes spécifiques en parallèle, mais celles-ci restent encore très problématiques pour la mise en œuvre simultanée de plusieurs caméras.

Il existe actuellement deux grandes approches du problème de la commande d'un bras manipulateur par un système de vision embarqué. Elles se différencient par la nature de la consigne imposée au système. La première de ces techniques consiste à asservir le robot de manière à obtenir dans l'image une vue particulière de l'objet cible lorsque la consigne est atteinte. Dans la seconde méthode, l'ensemble robot/caméra doit se positionner dans une attitude spatiale fixée par rapport à la cible poursuivie. On trouvera une étude complète de ces différentes techniques dans [59].

La première approche citée, aussi appelée *asservissement dans l'image* a fait l'objet des développements les plus récents dans le domaine de l'asservissement par vision (voir par exemple [18], [50] et plus récemment [58], [10]). La commande du robot est calculée de manière à positionner l'ensemble caméra/effecteur de façon à obtenir un motif prédéfini dans l'image. Ce type de méthode présente le grand avantage de s'affranchir des problèmes de calibration fine de la caméra et de limiter les opérations de traitement de l'image. Celles-ci se réduisent à l'extraction des primitives visuelles 2D et au calcul du motif courant. Les temps de calcul sont ainsi restreints et permettent de commander le bras manipulateur à la cadence vidéo. L'utilisation de notre algorithme de suivi 3D pour commander le bras robotique à partir du motif visuel observé dans l'image peut s'intégrer dans cette approche.

Dans le second type d'approche, le système robotique s'asservit sur une position et une orientation données de la caméra ou de l'effecteur par rapport à un objet cible. La commande est donc élaborée de manière à amener le bras dans une situation spatiale précise. La réalisation de ce type de consigne nécessite de calculer la localisation courante de l'objet par rapport à la caméra à chaque mise à jour de la commande. De ce fait, il est nécessaire d'avoir effectué un étalonnage précis de la caméra et de connaître le modèle géométrique de la cible. Bien que l'*asservissement en situation* ait été chronologiquement la première solution envisagée, les temps de calcul des algorithmes de localisation sont demeurés longtemps un obstacle à son emploi. Toutefois, le développement de nouvelles méthodes de calcul et l'apparition de machines plus performantes rendent maintenant possible la réalisation de ces traitements en quelques millisecondes.

4.3.3 Utilisation de l'algorithme de suivi 3D pour la commande en position "plus ou moins"

L'installation robotique du laboratoire est constituée par un robot portique AFMA commandé à partir d'un rack VME et d'une station de travail Silicon Graphics O_2. Ce robot, à structure cartésienne, possède six degrés de liberté qui se décomposent en trois axes de translation orthogonaux et trois rotations (voir annexe C pour une présentation plus détaillée). Pour les applications de vision, le robot est commandé à partir des informations collectées par la caméra vidéo embarquée sur l'effecteur.

L'algorithme prenant en charge le suivi 3D de l'objet dans l'image et l'élaboration de la

commande en position est exécuté sur la station de travail qui a la particularité d'avoir une carte d'acquisition vidéo intégrée permettant ainsi d'acquérir et de stocker les images en provenance de la caméra du robot. Celle-ci communique avec le rack VME via le réseau ethernet local. Le rack gère les dialogues entre la station et l'armoire de commande du robot par l'intermédiaire d'une application temps réel.

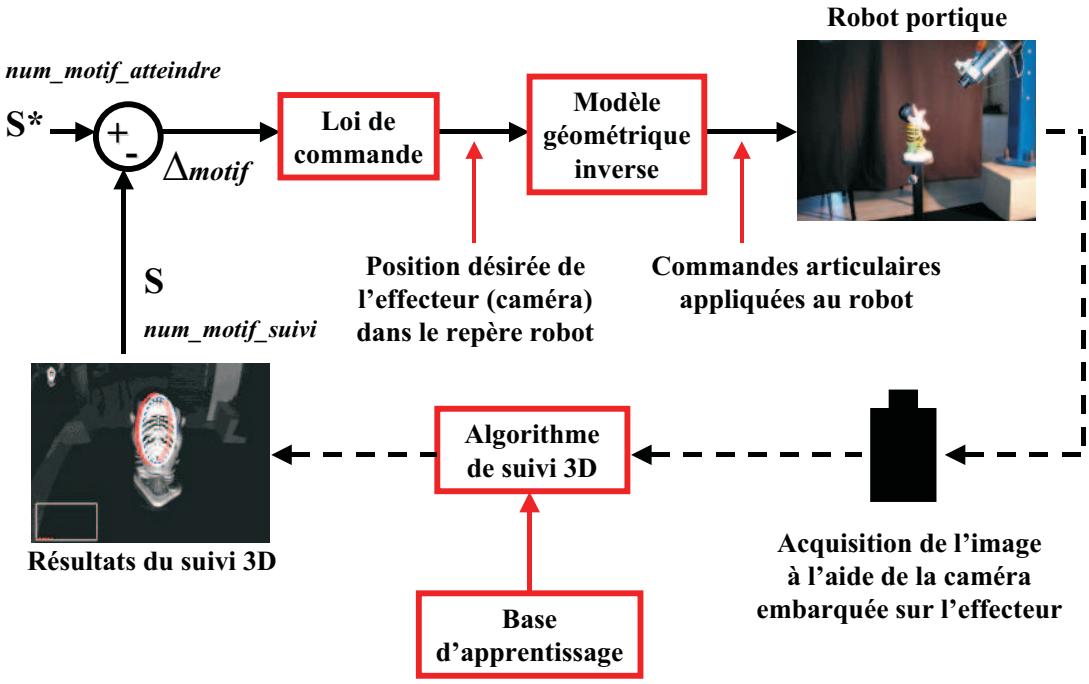


FIGURE 4.32 – commande en position "plus ou moins" du robot portique AFMA.

Notre application consiste à choisir aléatoirement une position angulaire en site du bras manipulateur sur la trajectoire définie lors de l'acquisition de la base d'apprentissage et à exécuter l'algorithme de détection et de reconnaissance automatique d'un motif dans l'image courante pour retrouver dans la collection d'images le motif de référence donnant la plus faible erreur quadratique de la différence de niveaux de gris entre son vecteur de forme et le motif échantillonné dans la zone d'intérêt après correction de sa position. Le motif courant d'indice *num_motif_suivi* étant maintenant reconnu, nous déterminons une consigne S^* définie comme le nouveau motif de référence à atteindre d'indice *num_motif_atteindre*.

La phase d'initialisation terminée, nous pouvons asservir la position angulaire en site de l'effecteur du robot autour de l'objet (figure 4.32). Pour cela, l'algorithme de suivi 3D est utilisé pour déterminer le motif de référence suivi dans l'image courante après chaque déplacement du bras robotique sur la trajectoire mémorisée et ceci jusqu'à ce que le motif de référence suivi correspond au motif de référence à atteindre. Pour chaque nouvelle acquisition d'image, la *loi*

de commande en position élabore la nouvelle valeur de l'angle de site $\alpha_{nouveau}$ en comparant la consigne S^* (le motif de référence à atteindre) avec la mesure courante S (le motif de référence actuellement suivi) issue de notre approche de suivi 3D. Le robot se déplace alors de 1 degré en site par rapport à sa position angulaire courante $\alpha_{courant}$ sur la portion de cercle pour obtenir la nouvelle position en site $\alpha_{nouveau}$. Ce déplacement correspond en fait à la variation de position en site du robot entre deux images consécutives de la base d'apprentissage modélisant la figurine. Lorsque la différence $\Delta motif$ entre les deux indices des motifs de référence devient nulle, la consigne est atteinte et le déplacement du robot est arrêté.

En posant $\Delta motif$ = *num_motif_atteindre* - *num_motif_suivi*, cette loi de commande en position "plus ou moins" peut être décrite de la manière suivante :

1. si $\Delta motif > 0$ alors $\alpha_{nouveau} = \alpha_{courant} + 1$ degré (le motif courant se situe avant le motif à atteindre dans la collection d'images de référence),
2. si $\Delta motif < 0$ alors $\alpha_{nouveau} = \alpha_{courant}$ - 1 degré (le motif courant se situe après le motif à atteindre dans la collection d'images de référence),
3. si $\Delta motif = 0$ alors $\alpha_{nouveau} = \alpha_{courant}$ (le motif courant correspondant au motif à atteindre, on arrête le déplacement du robot).

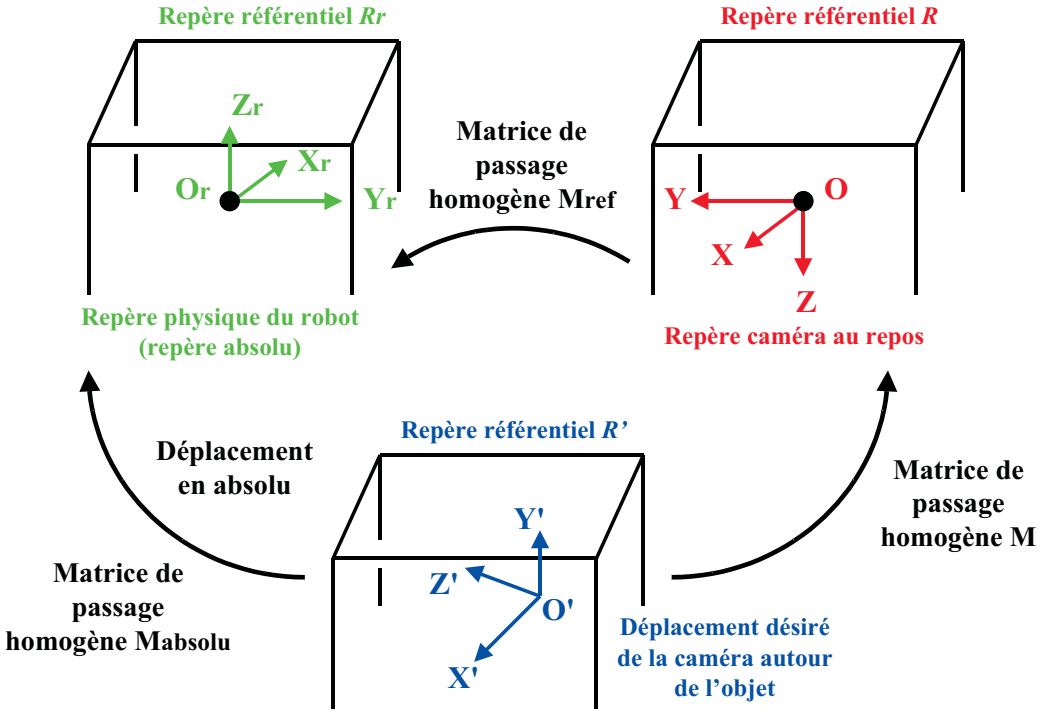


FIGURE 4.33 – estimation des coordonnées courantes de l'effecteur (caméra) dans le repère physique du robot (repère absolu).

A partir de la nouvelle valeur de l'angle de site $\alpha_{nouveau}$, nous exprimons le positionnement désiré de la caméra embarquée sur l'effecteur dans le repère robot R_r à l'aide de la matrice homogène M_{absolu} (figure 4.33). Cette matrice est obtenue à partir du produit de quatre matrices homogènes :

$$M_{absolu} = M_{ref} * M * M_{inclin} * M_{eff/cam} \tag{4.8}$$

– M_{ref} est la matrice de passage du repère caméra au repos R au repère physique du robot R_r (repère absolu) :

$$M_{ref} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.9}$$

– M est la matrice de passage du repère courant R' au repère caméra au repos R (ici $Tz = Azimut$) :

$$M = \begin{pmatrix} \sin\alpha_{nouveau} & 0 & -\cos\alpha_{nouveau} & R\cos\alpha_{nouveau} \\ \cos\alpha_{nouveau} & 0 & \sin\alpha_{nouveau} & -R\sin\alpha_{nouveau} \\ 0 & -1 & 0 & Azimut \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.10}$$

– M_{inclin} est la matrice homogène permettant d'incliner la caméra d'un angle β par rapport à l'horizontale tout en gardant son centre optique aligné avec le centre de gravité de l'objet (figue 4.34) :

$$M_{inclin} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.11}$$

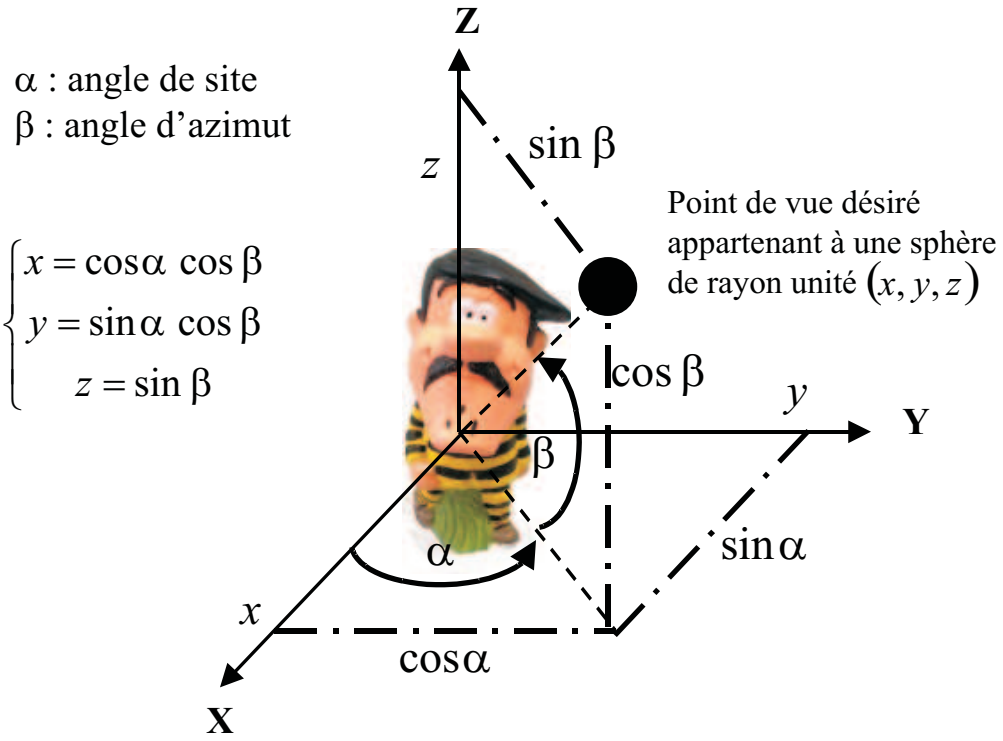


FIGURE 4.34 – coordonnées 3D d'un point sur une sphère de rayon unité.

– $M_{eff/cam}$ est la matrice de passage caméra / effecteur (lors du déplacement du robot, l'asservissement se fait par rapport au centre optique de la caméra située à l'extrémité de l'effecteur avec une distance $d_{bras/oeil} = 190$ mm) :

$$M_{eff/cam} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_{bras/oeil} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.12}$$

La position désirée de la caméra étant maintenant exprimée dans le repère absolu R_r à l'aide de la matrice homogène M_{absolu}, nous utilisons le *modèle géométrique inverse du robot* pour retrouver les commandes articulaires à appliquer a chacun des six axes du robot.

Comme l'objet est proche du capteur vidéo, nous sommes obligés d'utiliser une caméra à focale courte entraînant des distorsions importantes de l'objet dans l'image. De ce fait, nous ne pouvons pas supposer que la profondeur de l'objet par rapport à la distance objet/caméra reste faible. Pour toutes ces raisons, nous n'utilisons pas dans cette application de matrices d'interaction B pour estimer les variations du motif dans l'image.

4.3.4 Simulation de l'algorithme de suivi 3D pour la navigation en position

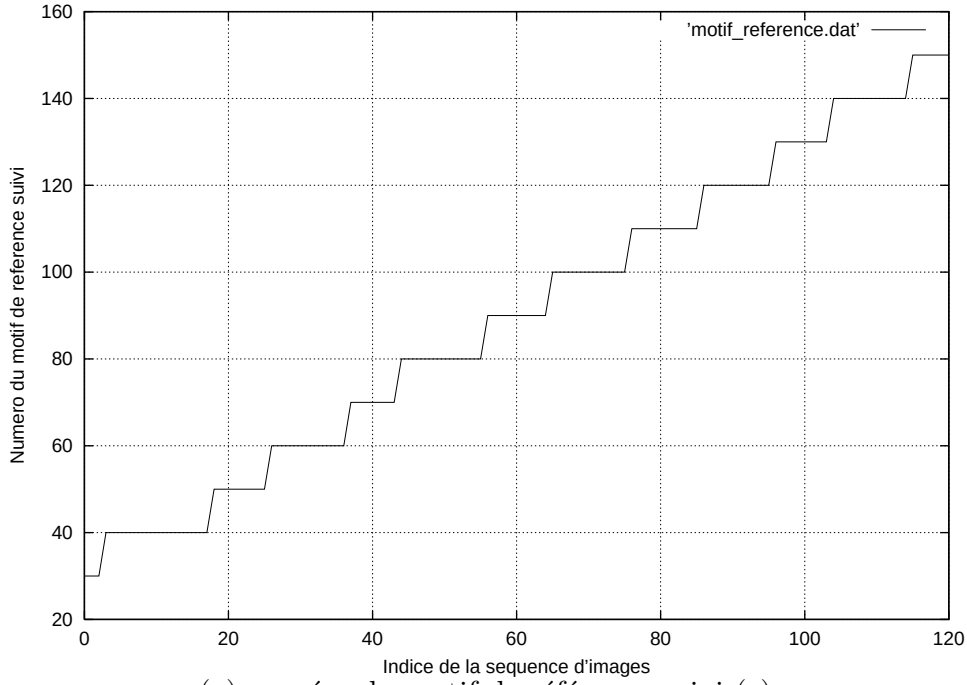
Avant d'expérimenter l'algorithme en situation réelle, nous avons d'abord simulé son comportement sur l'ensemble des images de la base d'apprentissage où le passage entre deux images consécutives correspond à une variation en site de 1 degré de la caméra embarquée par rapport à l'objet.

Pour chacune de ces images, nous enregistrons les différentes erreurs quadratiques de la différence de niveaux de gris entre le motif de référence testé et le motif échantillonné dans l'ellipse corrigée ainsi que l'indice du motif de référence donnant l'erreur quadratique la plus faible et qui sera le nouveau motif à suivre dans la prochaine image.

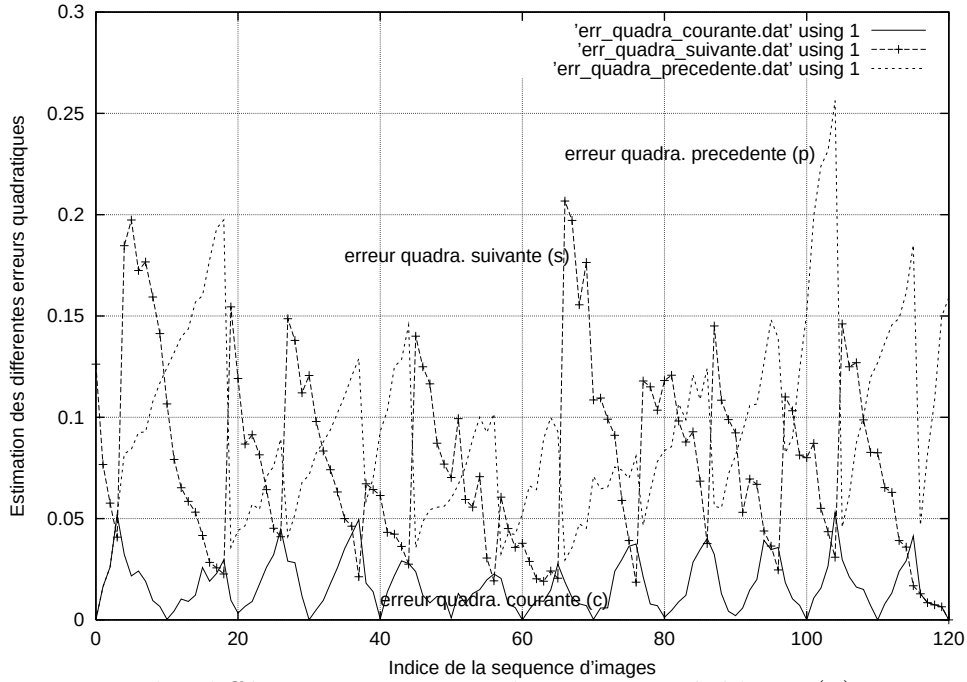
Les figures 4.35 et 4.36 montrent les résultats obtenus par l'algorithme de suivi 3D sur l'ensemble de la collection d'images. L'initialisation du suivi est réalisée soit sur la première image de la base d'apprentissage (vue de référence 30) soit sur la dernière (vue de référence 150).

Pour ces deux essais, comme pour les deux applications décrites précédemment, l'enchaînement des motifs de référence à suivre s'effectue correctement par paliers successifs. Ce changement de motif de référence se produit lorsqu'une des deux erreurs quadratiques précédente (p) ou suivante (s) devient inférieure à l'erreur quadratique (c) associée au motif de référence actuellement suivi dans l'image courante. Nous remarquons également que la valeur de l'erreur quadratique courante (c) tend vers une valeur proche de zéro pour chacune des vues de référence modélisant une des apparences de l'objet 3D (la figurine) à un instant du suivi. Cela s'explique, tout simplement, car c'est à partir de ces vues que la phase d'apprentissage hors ligne est réalisée pour assurer le suivi 2D de chacun des motifs de référence. Par conséquent, au cours du traitement de la séquence d'images en simulation, et en particulier, pour chacune des images de référence, l'ellipse corrigée englobant le motif courant suivi doit parfaitement se recaler sur le motif de référence défini lors de la phase d'apprentissage (sélection manuelle par l'opérateur du motif de référence à l'intérieur d'une zone d'intérêt : l'ellipse). Pour ces vues particulières, nous n'avons pas de variation d'apparence du motif suivi car nous n'avons pas de variation angulaire relative en site. La valeur de l'erreur quadratique de la différence de niveaux de gris entre le motif courant échantillonné dans l'ellipse corrigée et le vecteur de forme du motif de référence suivi est alors proche de zéro. Dans les vues intermédiaires, l'erreur

quadratique courante (c) augmente quand on s'éloigne de la vue de référence associée au motif actuellement suivi puisque les variations d'apparence sont plus importantes jusqu'au moment où nous changeons de motif de référence à suivre.

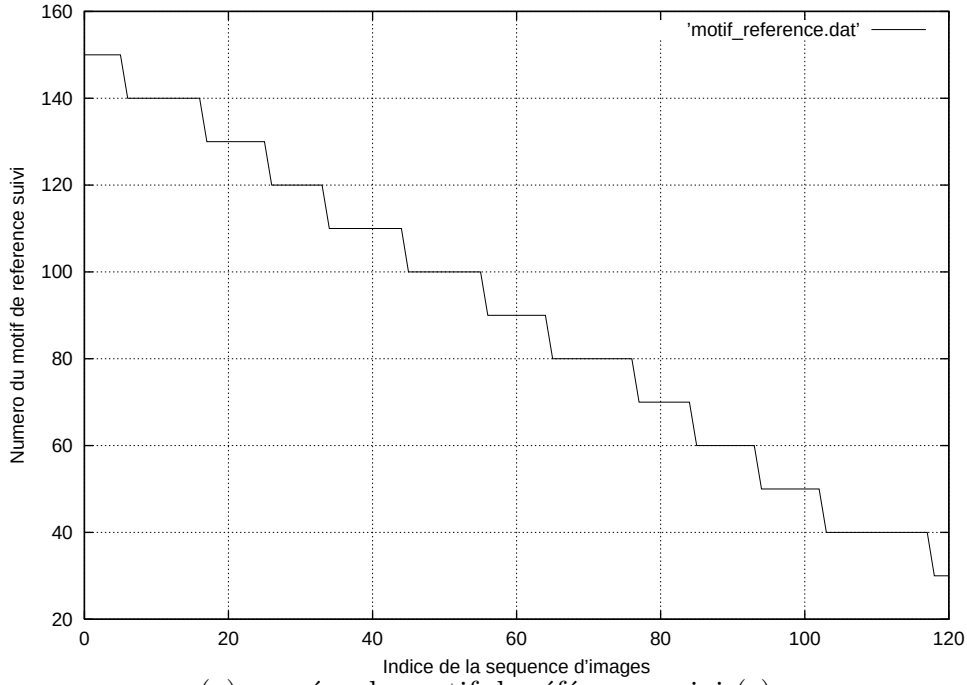


(a) numéro du motif de référence suivi (c).

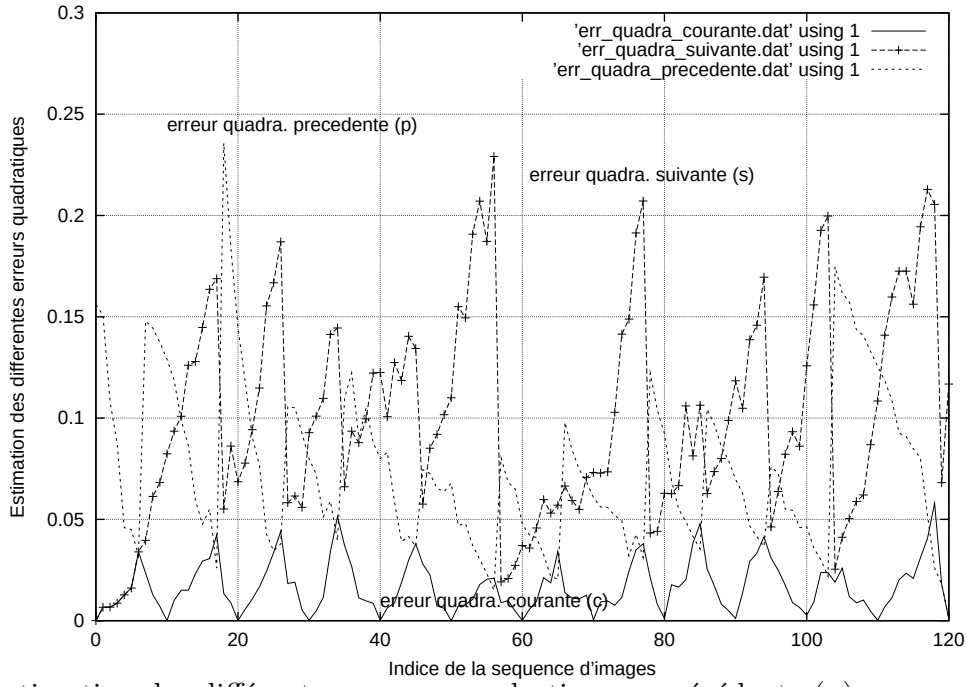


(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.35 – résultats de l'algorithme de suivi sur l'ensemble des 121 vues de la base d'apprentissage (de la première image vers la dernière image).



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.36 – résultats de l'algorithme de suivi sur l'ensemble des 121 vues de la base d'apprentissage (de la dernière image vers la première image).

Si nous comparons ces deux essais avec les résultats de simulation obtenus pour les deux expérimentations précédentes, plusieurs remarques peuvent être formulées :

– à la différence du suivi 3D de visages, les valeurs des trois erreurs quadratiques sont

très faibles. Ici, les erreurs quadratiques précédente (p) et suivante (s) restent inférieures à 0.27 alors que dans la première application, elles pouvaient atteindre la valeur 2.70. Même l'erreur quadratique courante (c) reste très inférieure (0.07 actuellement contre 0.26 pour le suivi de visages). Ceci s'explique par la différence de nature des objets suivis (un visage et une figurine) avec des formes plus ou moins grossières et une apparence pouvant produire des différences de niveaux de gris plus ou moins importantes (textures homogène pour le visage et plus nuancée pour la figurine). De plus, le choix d'une ellipse par image de référence dans le cas de la figurine permet de sélectionner plus précisement le motif de référence qu'une seule ellipse pour l'ensemble des vues de référence modélisant le visage.

– ayant utilisé le même objet 3D (la figurine) pour les deux commandes en position, nous nous apercevons que les erreurs quadratiques issues de la simulation sont très différentes :

  1. erreurs quadratiques précédente (p) et suivante (s) inférieures à 0.70 pour la table et à 0.27 pour le robot,

  2. erreur quadratique courante (c) inférieure à 0.18 pour la table et à 0.07 pour le robot.

  Cela signifie que pour un même objet mais avec un éclairage de la scène et des motifs de référence modélisant son apparence 3D différents, nous pouvons obtenir des résultats de suivi avec des variations d'amplitude très différentes. C'est pourquoi, il est important de porter une grande attention aux phases de modélisation 3D de l'objet et d'apprentissage hors ligne pour garantir la fiabilité du suivi avec des conditions données d'illumination.

Ces différents résultats nous ont permis de vérifier que nous étions capables d'assurer le suivi sans ambiguïté sur l'ensemble de la séquence d'images. Nous pourrons, par conséquent, initialiser correctement notre application quel que soit le motif détecté dans l'image courante et donner par la suite des informations fiables à la loi de commande pour contrôler le déplacement du robot autour de l'objet en condition réelle d'expérimentation.

4.3.5 Expérimentation de l'algorithme de navigation autour d'un objet

Avant de présenter l'application illustrée par la figure 4.37, nous rappelons que l'objet est positionné de façon que l'axe optique de la caméra passe au mieux par son centre de gravité afin que ce dernier apparaisse centré dans l'image.

Le principe de l'expérimentation consiste à tirer de manière aléatoire une position en site

du bras robotique sur la trajectoire circulaire définie lors de l'acquisition de la base d'apprentissage avant l'initialisation du processus. Nous obtenons une variation angulaire $\alpha = 140$ degrés.

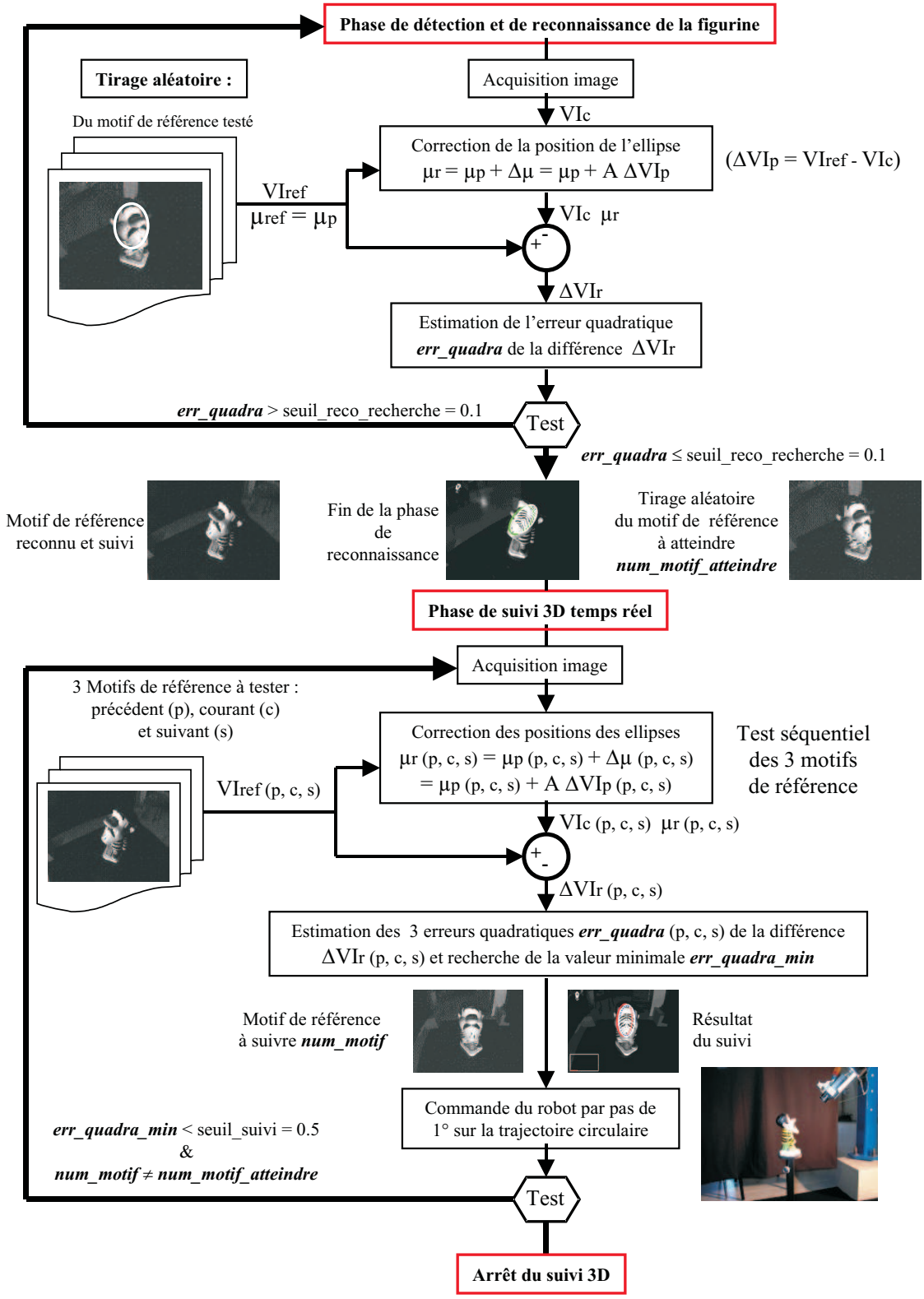


FIGURE 4.37 – principe de l'algorithme de navigation du bras robotique autour d'un objet connu.

L'algorithme de détection a reconnu le motif de référence d'indice 140 comme étant le motif de référence donnant l'erreur quadratique la plus faible dans l'image courante. Nous fixons ensuite le motif d'indice 70 comme étant le motif de référence à atteindre. L'algorithme de suivi 3D et la loi de commande en position contrôlent alors le déplacement en site du bras robotique par pas de 1 degré jusqu'à ce que le motif suivi d'indice *num_motif_suivi* dans l'image courante soit le motif de référence d'indice $num\_motif\_atteindre = 70$. Notons que ce déplacement entre deux acquisitions d'image successives correspond à la variation de position angulaire en site de l'effecteur autour de l'objet utilisée lors de l'acquisition de la base d'apprentissage.

Cet algorithme, développé sur une station de travail Silicon Graphics O_2, a un temps d'exécution inférieur à 20 ms. Comme pour la commande en position de la table à déplacement micrométrique, il fournit en temps réel les informations associées à l'erreur quadratique courante, le motif de référence suivi et la position de l'ellipse pour chacun des trois motifs de référence testés (figure 4.38).

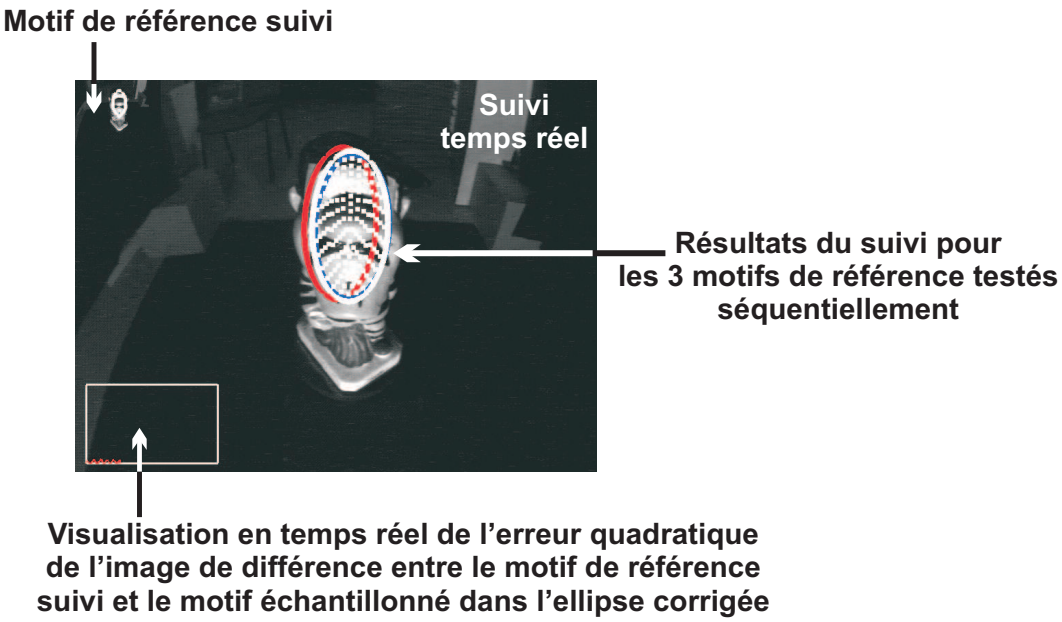


FIGURE 4.38 – visualisation des résultats lors de la commande en position du robot.

La figure 4.39 illustre les résultats de notre algorithme obtenus à des instants différents de la navigation. Au cours de l'asservissement visuel, nous sauvegardons, après chaque nouvelle acquisition d'image, les valeurs des différentes erreurs quadratiques associées aux trois motifs de référence testés séquentiellement et l'indice du motif de référence à suivre dans la prochaine image.

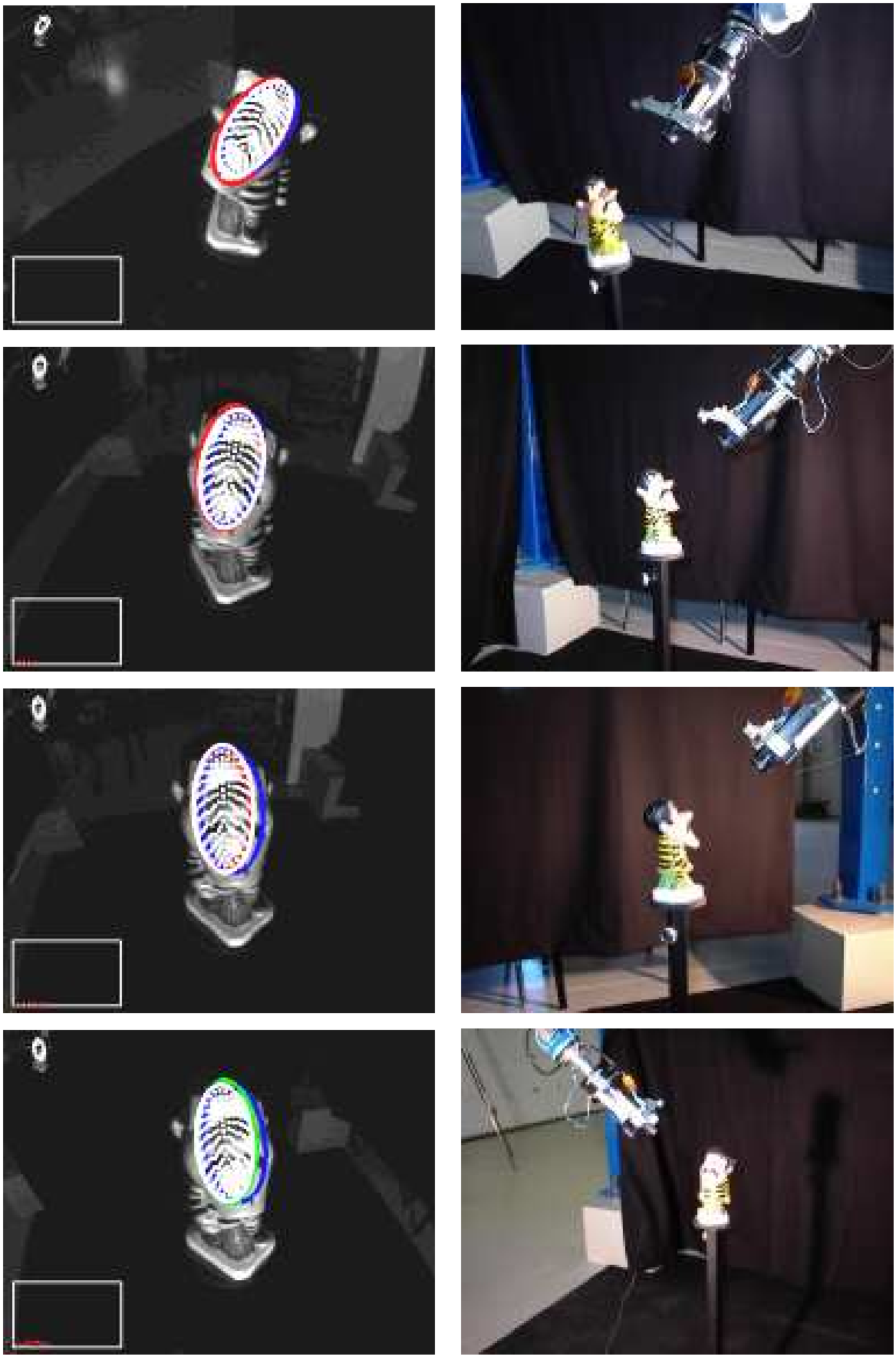
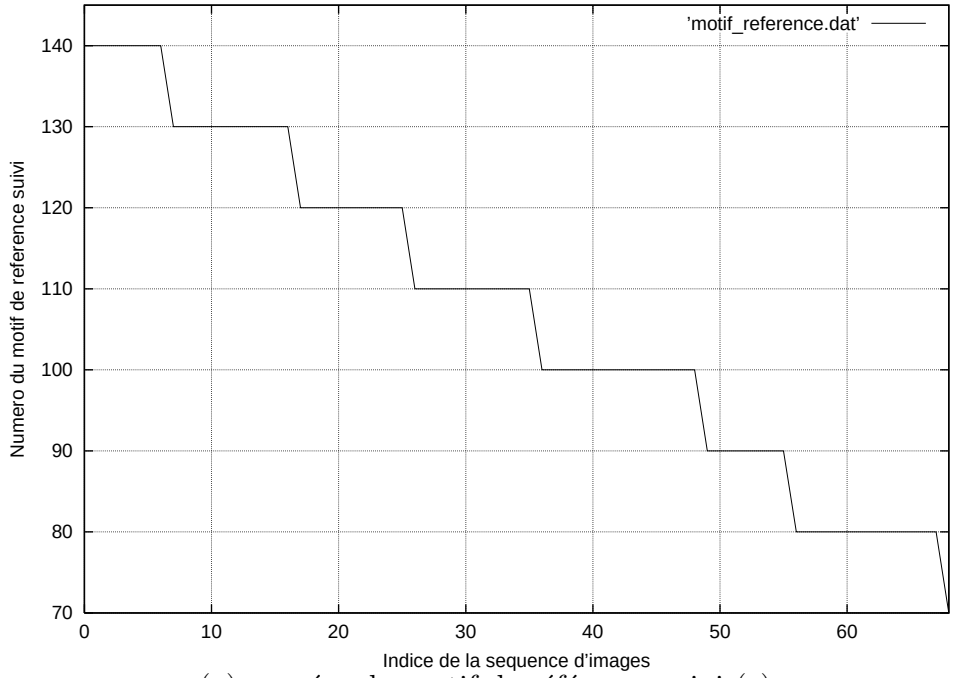
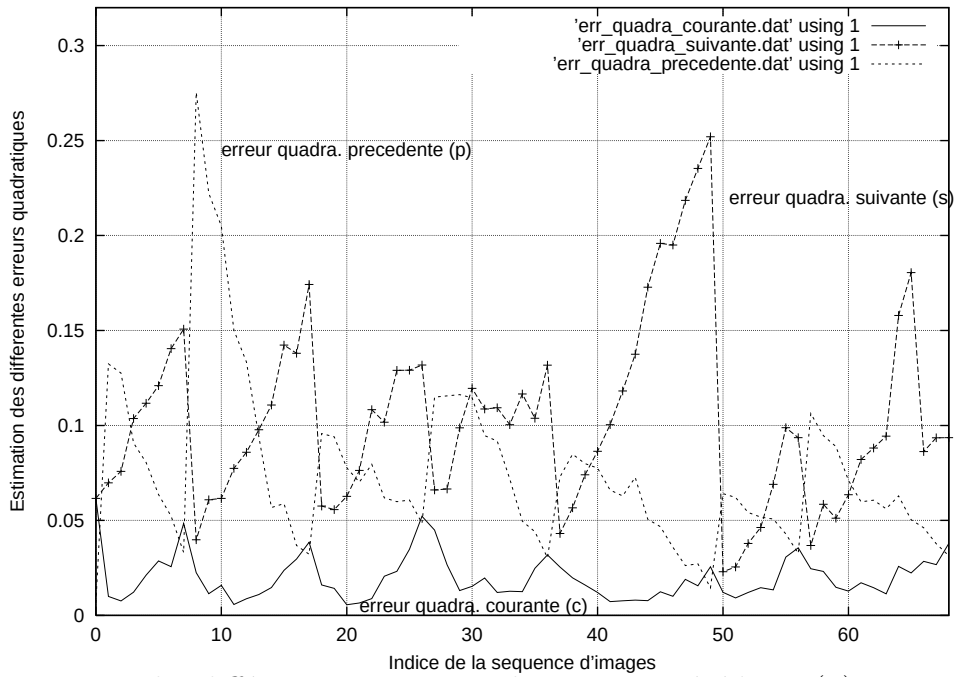


FIGURE 4.39 – exemple de résultats de la commande en position du robot.

Ces différents résultats présentés sur la figure 4.40 peuvent être comparés avec ceux obtenus lors de la simulation (figures 4.35 et 4.36).



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.40 – résultats de l'algorithme de navigation en condition réelle.

Nous remarquons que le changement de motif de référence s'opère toujours par paliers et

que les valeurs des différentes erreurs quadratiques en pratique et en simulation sont du même ordre de grandeur (erreur courante $<$ à 0.07, erreurs précédente et suivante $<$ à 0.30). Ceci s'explique, par le fait, que nous contrôlons parfaitement l'éclairage de la scène (illumination artificielle dominante) et que l'objet est positionné exactement de la même manière que pendant l'acquisition de la collection d'images 2D utilisée pour sa modélisation 3D. De plus, les mouvements élaborés en site par la loi de commande sont parfaitement modélisés par la base d'apprentissage et détectés correctement par l'algorithme de suivi permettant ainsi de garantir une fiabilité dans le contrôle du bras robotique. Cette expérimentation montre donc que l'algorithme de suivi 3D peut être utilisé dans différentes applications de la vision artificielle en fournissant toujours d'excellents résultats dans la mesure où les conditions d'éclairement varient faiblement entre le moment d'acquérir la base d'apprentissage et la phase de suivi en ligne.

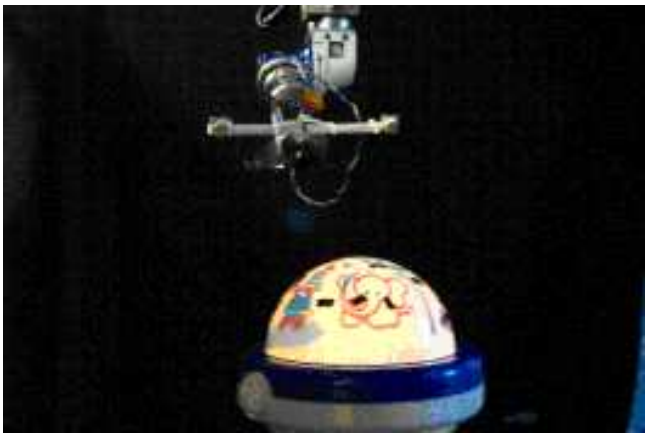
L'inconvénient majeur de cette commande en position "plus ou moins" est que l'objet doit rester centré dans l'image au cours de l'asservissement visuel. En effet, la mesure courante S que l'on asservit par rapport à la consigne S^* correspond à l'indice du motif de référence suivi (*num_motif_suivi*) dans l'image courante et non les paramètres corrigés de l'ellipse englobant le motif suivi. Pour remédier au problème où l'objet peut être en mouvement en même temps que l'effecteur du robot, nous avons développé une commande en vitesse. Dans ce cas précis, la consigne S^* n'est plus le motif de référence à atteindre (*num_motif_atteindre*) mais les paramètres de l'ellipse que l'on souhaite obtenir dans l'image (en position, échelle et orientation) pour le motif de référence actuellement suivi. Pour cette nouvelle approche, l'algorithme de suivi 3D assure la gestion du changement de motif de référence et fournit maintenant, comme information à la loi de commande en vitesse, les paramètres de l'ellipse corrigée englobant le motif suivi dans l'image courante. Cette loi de commande compare alors la forme courante de l'ellipse avec celle définie comme consigne pour la positionner, comme il convient, dans l'image en déplaçant l'effecteur du robot. Le développement de cette commande en vitesse fait l'objet de la prochaine section de ce chapitre.

4.4 Commande en vitesse d'un bras robotique

Dans cette dernière application, nous souhaitons utiliser notre algorithme de suivi 3D associé à une commande en vitesse pour asservir la position du bras robotique afin d'obtenir, dans l'image courante, la forme de l'ellipse (en position, échelle et orientation) définie comme consigne. Des compléments d'information sur les asservissements visuels sont donnés dans l'an-

nexe C.

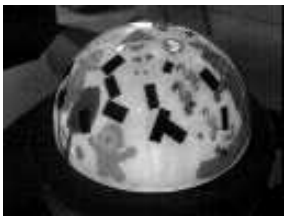
Pour présenter nos résultats, la base d'apprentissage modélisant l'objet 3D (un globe lumineux) comprend au total 141 images (soit un total de 15 images de référence dont l'indice est un multiple de 10) (figure 4.41). Chaque motif de référence sera défini à partir d'une ellipse différente sélectionnée par l'opérateur. Durant cette phase d'acquisition où l'objet doit rester centré dans l'image, le bras robotique décrit une trajectoire circulaire pour laquelle l'angle de site α varie de 20 à 160 degrés par pas de 1 degré (d'après les conventions prises pour le changement de repère illustré figure 4.28).



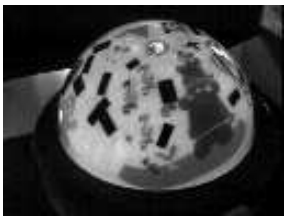
globe lumineux 3D à modéliser

vue de référence 20

vue de référence 60



vue de référence 120



vue de référence 160

FIGURE 4.41 – modélisation du globe lumineux pour une variation en site de 140 degrés.

Durant la construction de la base d'apprentissage, nous ne pouvons pas utiliser l'éclairage embarqué sur l'effecteur car la distance objet/caméra est trop faible (de l'ordre d'une vingtaine de centimètres). Au lieu de se servir d'une source de lumière artificielle située derrière le bras manipulateur comme pour la commande en position, nous avons choisi ce globe lumineux qui simplifie la mise en oeuvre de l'expérimentation et qui, nous l'espérons, nous rendra moins sensible aux variations d'illumination naturelle de la scène.

Dans les paragraphes suivants, nous présentons la commande en vitesse que nous avons développée avant de tester en condition réelle ses performances qui reposent principalement sur la fiabilité des résultats obtenus à partir de notre algorithme de suivi 3D. Durant l'asservissement dans l'image, l'objet peut maintenant se déplacer en même temps que l'effecteur du robot.

4.4.1 Solution retenue pour la commande en vitesse

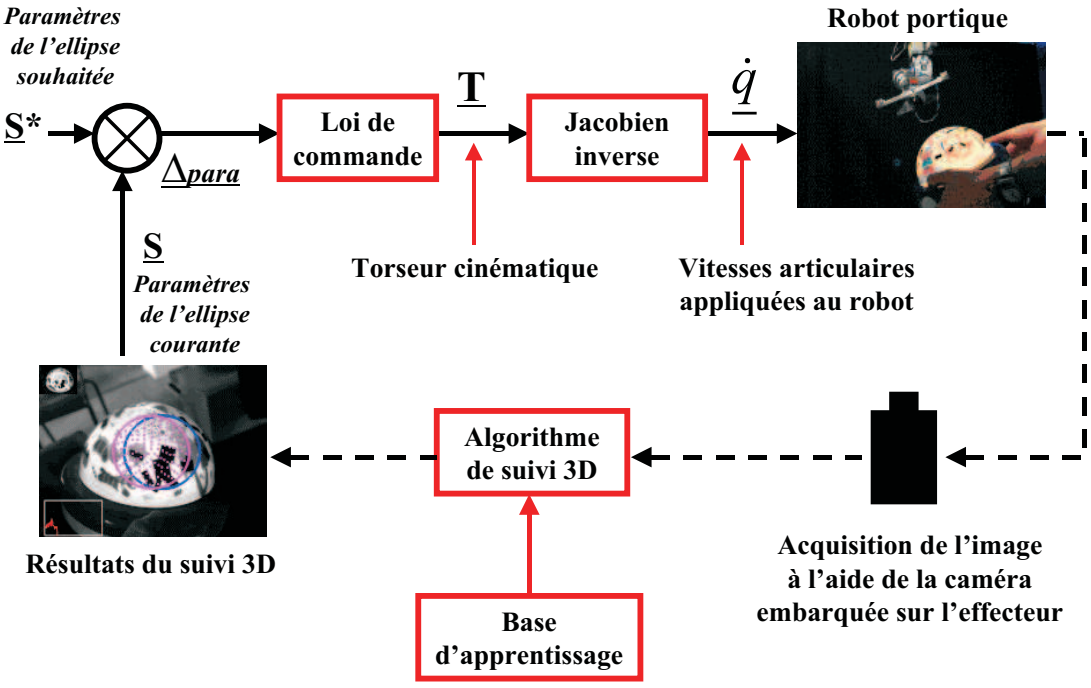


FIGURE 4.42 – commande en vitesse du robot portique AFMA.

Pour initialiser l'application, nous positionnons le bras manipulateur sur la trajectoire définie lors de l'acquisition de la base d'apprentissage pour une variation en site α égale à 90 degrés. Après avoir placé l'objet devant la caméra (de préférence au centre de l'image), nous exécutons l'algorithme de détection et de reconnaissance automatique d'un motif dans l'image pour retrouver, dans la collection d'images, le motif de référence donnant la plus faible erreur quadratique de la différence de niveaux de gris entre son vecteur de forme et le motif échantillonné dans la zone d'intérêt après correction de sa position. Le motif courant étant maintenant reconnu et l'ellipse recalée sur le motif suivi, nous déterminons une consigne $\underline{S}^*$ définie à partir des paramètres de l'ellipse que l'on souhaite obtenir dans l'image.

A partir de la taille originale d'une image courante (768*576 pixels), la forme de cette ellipse illustrée figure 4.43 peut être décrite par le vecteur de paramètres suivant :

– *position* centrée dans l'image : $X_c^* = 384$ et $Y_c^* = 288$,

– *orientation* : θ^* = -90 degrés (ou -1.57 radians),

– *échelle* : $R_1^* = 120$ et $R_2^* = k * R_1^*$ par définition.

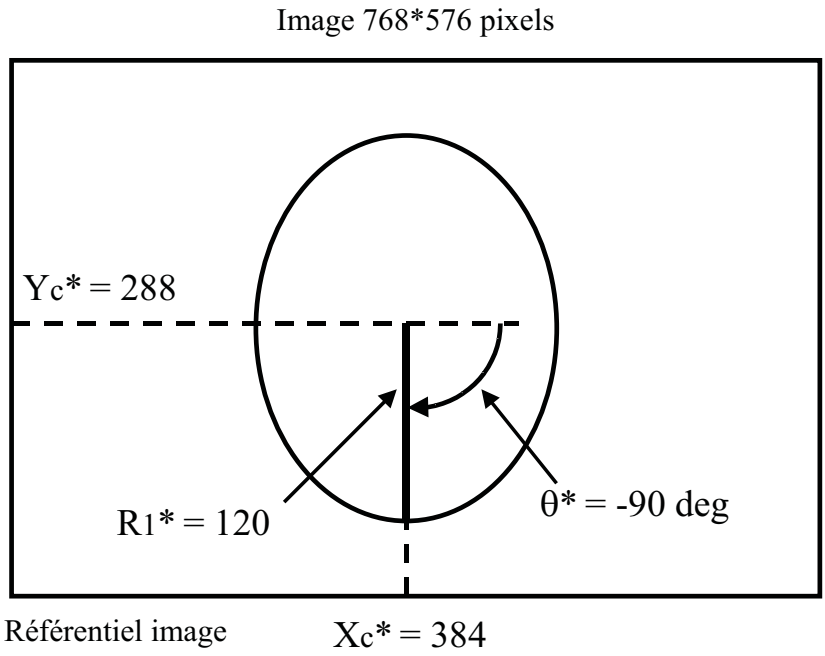


FIGURE 4.43 – définition des paramètres de l'ellipse utilisés comme consigne.

La phase d'initialisation terminée, nous pouvons maintenant asservir en vitesse la position de l'effecteur par rapport au motif suivi dans l'image courante. Le principe de cet asservissement visuel est décrit figure 4.42. Ici, l'algorithme de suivi 3D est utilisé pour gérer les changements de motif de référence et déterminer les paramètres courants de l'ellipse corrigée après chaque déplacement du bras robotique. Pour chaque nouvelle acquisition d'image, la *loi de commande en vitesse* calcule le *torseur cinématique* $\underline{T}$ en comparant la consigne $\underline{S}^*$ (le vecteur des paramètres de l'ellipse à atteindre) avec la mesure courante $\underline{S}$ (les paramètres courants de l'ellipse recalée sur le motif suivi) issue de notre approche de suivi 3D. Le torseur cinématique étant estimé, nous utilisons le *Jacobien inverse du robot* pour calculer les vitesses articulaires $\underline{\dot{q}}$ à appliquer pour commander le déplacement de la caméra embarquée sur l'effecteur du robot (sachant que $\underline{q}$ est le vecteur des variables articulaires). Les mouvements du robot sont engendrés par l'application de couples $\underline{\Gamma}$ qui sont transmis aux différents axes du robot par l'intermédiaire d'actionneurs. Le comportement dynamique d'un robot rigide est décrit par l'équation d'état suivante :

$$\underline{\Gamma} = M(\underline{q}) \cdot \underline{\ddot{q}} + N(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}) \tag{4.13}$$

où :

- $\underline{\Gamma}$ est le vecteur des couples extérieurs appliqués au robot,
- $M(\underline{q})$ représente la matrice d'énergie cinétique,
- $N(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}})$ rassemble les contributions des forces de gravité, centrifuge, de Coriolis et de frottement.

Pour illustrer cette loi de commande, prenons l'exemple d'une commande en vitesse simplifiée. D'après le formalisme de la commande référencée capteur (voir annexe C), une *tâche robotique* peut se mettre sous la forme d'une *régulation à zéro d'une fonction de tâche* $\underline{E}$ qui exprime *l'erreur entre la configuration souhaitée et la configuration courante* :

$$\underline{E} = C \cdot (\underline{S} - \underline{S}^*) \tag{4.14}$$

où C est la *matrice de combinaison* permettant de prendre en compte un nombre d'informations visuelles supérieur au nombre de degrés de liberté contraint par la liaison souhaitée entre le robot et la caméra.

Pour une cible fixe, la dérivée temporelle des primitives visuelles s'écrit :

$$\dot{\underline{S}} = L_{\underline{S}} \cdot \underline{T} \tag{4.15}$$

où $L_{\underline{S}}$ est la *matrice d'interaction*, appelée aussi *Jacobien de tâche* ou *Jacobien d'image*. Cette matrice traduit les variations des informations visuelles en fonction des différents déplacements de la caméra. Elle caractérise complètement les interactions entre le capteur et son environnement. Le *torseur cinématique* $\underline{T}$ représente, quant à lui, les vitesses de translation et de rotation du repère caméra dans le repère absolu du robot.

Si on impose une décroissance exponentielle de la fonction de tâche, alors :

$$\dot{\underline{E}} = -\lambda \cdot \underline{E} = -\lambda \cdot C \cdot (\underline{S} - \underline{S}^*) \tag{4.16}$$

et on obtient la commande en vitesse suivante :

$$\underline{T} = -\lambda \cdot C \cdot (\underline{S} - \underline{S}^*) \tag{4.17}$$

où λ est un gain positif fixé par l'utilisateur.

Pour assurer la convergence de la loi de commande, il suffit de prendre C tel que $L_{\underline{S}} \cdot C > 0$. Un choix optimal pour C est de la prendre égale à l'inverse de $L_{\underline{S}}$ (notée $L_{\underline{S}}^{-1}$) si la matrice $L_{\underline{S}}$

est carrée, ou à la pseudo-inverse de $L_{\underline{S}}$ (notée $L_{\underline{S}}^{+}$) si $L_{\underline{S}}$ n'est pas une matrice carrée. Pour une tâche donnée, le problème consiste à choisir $\underline{S}$ pertinemment afin de réaliser la régulation de $\underline{E}$, puis à construire C. Pour cela, il nous faut déterminer la matrice d'interaction (relative au choix de $\underline{S}$).

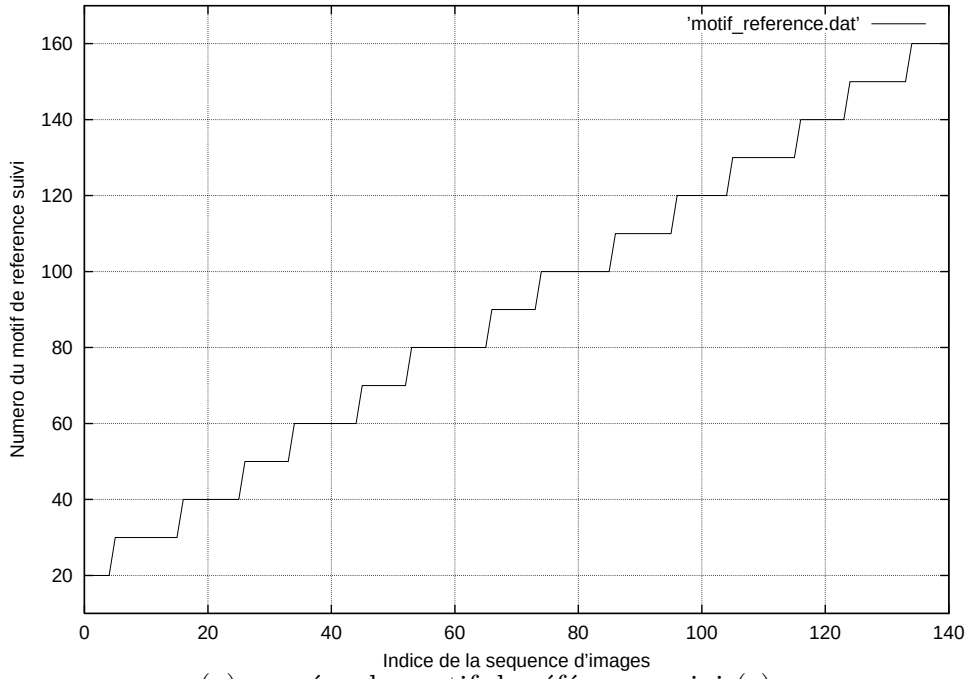
4.4.2 Simulation de l'algorithme de suivi 3D pour la commande en vitesse

Avant d'expérimenter l'algorithme en situation réelle, nous devons vérifier que nous sommes capables d'assurer le suivi sur l'ensemble de la collection d'images qui modélise l'objet 3D. En effet, les performances de l'algorithme de commande en vitesse sont fortement liées aux résultats de notre approche de suivi 3D (les paramètres courants de l'ellipse corrigée englobant le motif suivi). Nous avons donc simulé son comportement sur l'ensemble des images de la base d'apprentissage où le passage entre deux images consécutives correspond à une variation en site de 1 degré de la caméra embarquée par rapport à l'objet (le globe lumineux).

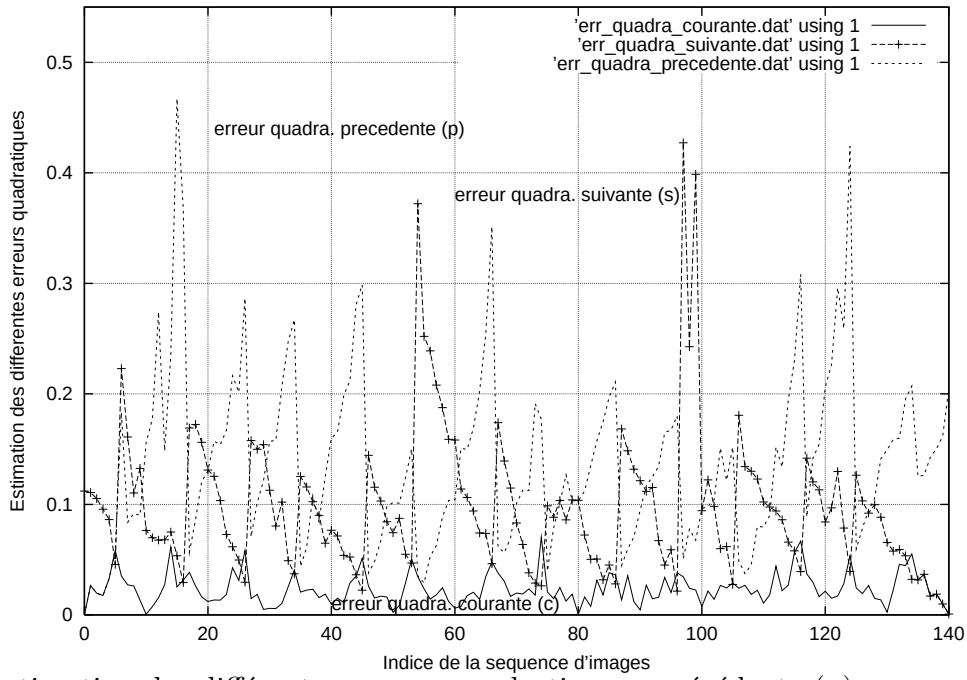
Pour chacune de ces images, nous enregistrons l'erreur quadratique de la différence de niveaux de gris associée à chacun des motifs de référence testés après correction des paramètres de l'ellipse ainsi que le numéro du nouveau motif de référence à suivre dans la prochaine image.

Les figures 4.44 et 4.45 montrent les résultats obtenus par l'algorithme de suivi 3D sur l'ensemble de la collection d'images. L'initialisation du suivi est réalisée soit sur la première image de la base d'apprentissage (vue de référence 20) soit sur la dernière (vue de référence 160). Bien que l'objet à modéliser soit différent ainsi que les conditions d'éclairage de la scène, le suivi 3D est correctement assuré. Le changement de motif de référence s'effectue toujours régulièrement autour des vues intermédiaires situées à "mi-chemin" des différentes vues de référence et ceci quand l'une des deux erreurs quadratiques précédente (p) ou suivante (s) devient inférieure à l'erreur courante (c). Les valeurs de ces erreurs restent relativement faibles et du même ordre de grandeur que pour la figurine malgré des propriétés intrinsèques (forme, apparence et réflectance) différentes entre les deux objets (erreurs précédente et suivante $<$ à 0.55 et erreur courante $<$ à 0.10). Cette régularité dans les résultats observés jusqu'à présent se justifie par la méthode d'acquisition de la collection d'images modélisant l'objet 3D pour une variation angulaire donnée (ici en site). L'écart relatif de 10 degrés en site entre deux images de référence consécutives de la base semble bien approprié pour assurer un changement fiable de

motif de référence même si nous n'avons pas développé en parallèle, une méthode permettant d'optimiser le nombre de vues de référence représentant au mieux la modélisation 3D d'un objet à partir d'une collection d'images.

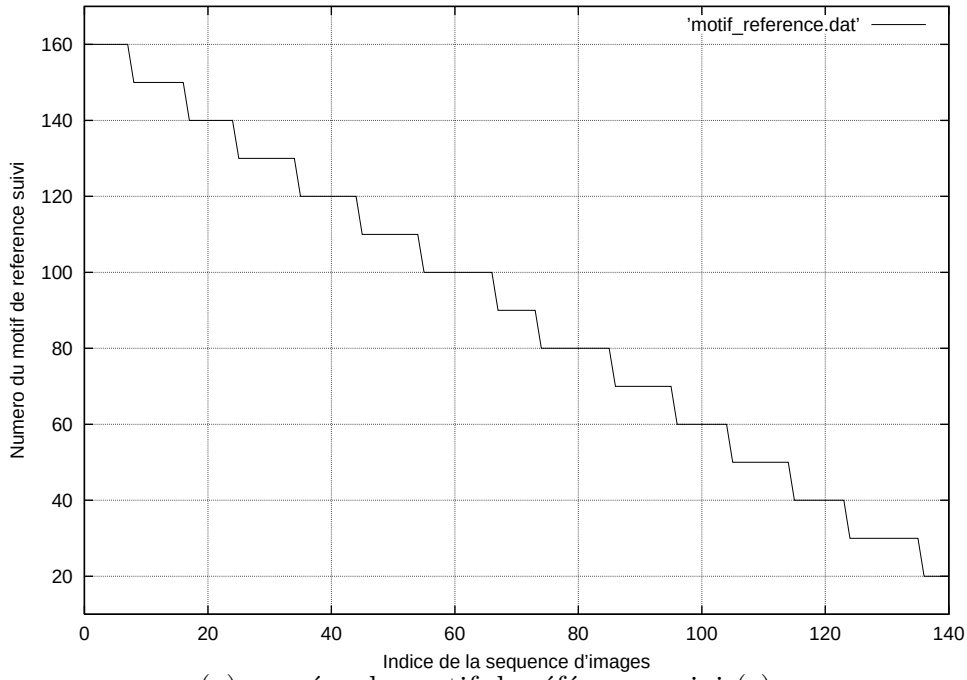


(a) numéro du motif de référence suivi (c).

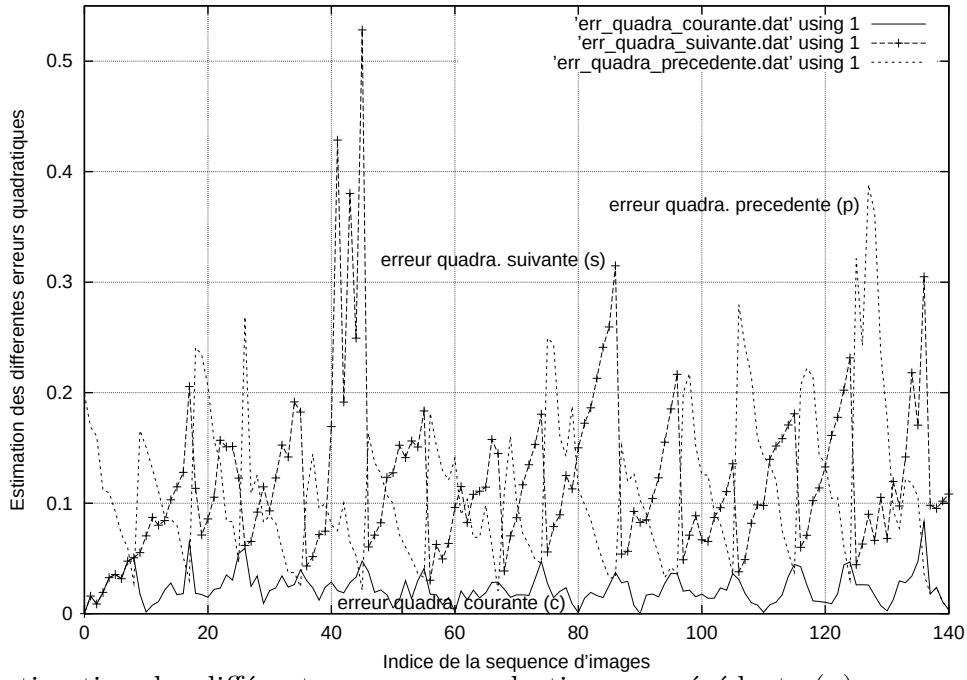


(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.44 – résultats de l'algorithme de suivi sur l'ensemble des 141 vues de la base d'apprentissage (de la première image vers la dernière image).



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.45 – résultats de l'algorithme de suivi sur l'ensemble des 141 vues de la base d'apprentissage (de la dernière image vers la première image).

Le suivi 3D étant validé sur l'ensemble de la base d'apprentissage, nous pouvons maintenant tester les performances de l'asservissement en vitesse en condition réelle, à partir des différents résultats issus de notre approche 3D (en particulier, les paramètres courants de l'ellipse cor-

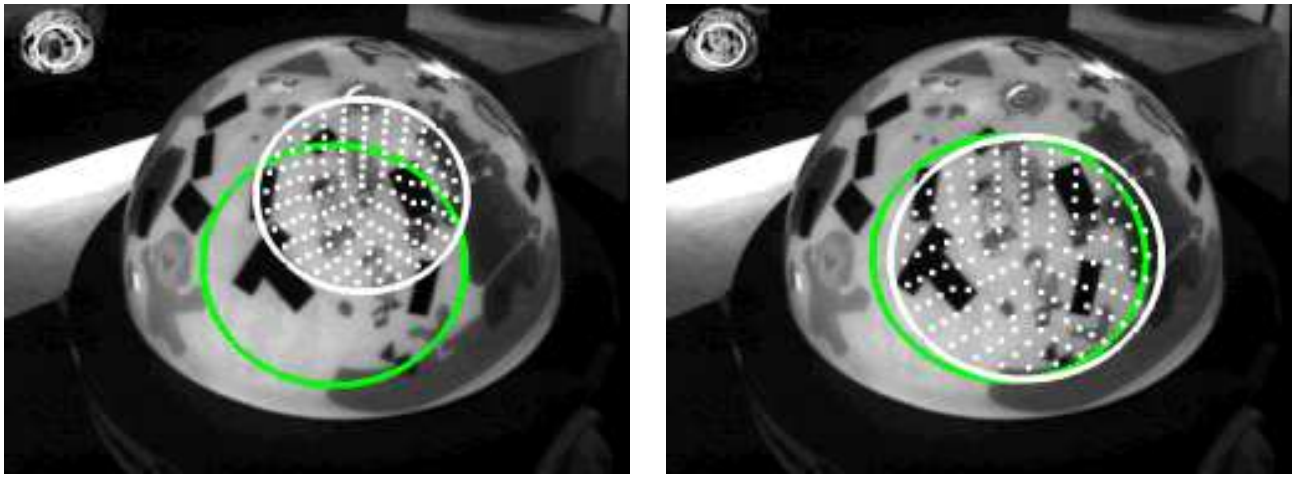
rigée). D'après les résultats de ces deux essais, nous pourrons initialiser correctement notre application quel que soit le motif détecté dans l'image courante.

4.4.3 Expérimentation de l'algorithme de commande en vitesse du bras robotique

Les différents résultats présentés maintenant ont été obtenus pour une séquence d'acquisition de 400 images. Ils montreront sous forme de graphiques et d'illustrations l'évolution de l'asservissement visuel au cours du temps. Pour chacune des images de la séquence, nous avons sauvegardé les informations suivantes :

1. *les résultats issus de l'algorithme de suivi 3D* :
   - la valeur de l'erreur quadratique associée à chacun des trois motifs de référence testés séquentiellement,
   - l'indice du motif de référence donnant l'erreur quadratique la plus faible et correspondant au nouveau motif à suivre dans la prochaine image,
   - les paramètres courants de l'ellipse corrigée englobant le motif suivi.
2. *les résultats issus de la loi de commande en vitesse* :
   - les valeurs des différentes composantes du torseur cinématique $\underline{T}$.

Initialisation de l'application



(a) recherche aléatoire d'un motif dans l'image.

(b) détection et reconnaissance du motif dans l'image.

FIGURE 4.46 – phase de détection automatique d'un motif dans l'image.

L'objet étant posé devant la caméra, l'algorithme de détection automatique d'un motif

dans l'image est exécuté pour initialiser l'application (figure 4.46). Pour cela, nous tirons aléatoirement un motif de référence dans la base d'apprentissage et une position de l'ellipse dans l'image courante. Nous cherchons alors à corriger les paramètres de cette ellipse pour essayer de se recaler sur le motif détecté et ceci, tant que l'erreur quadratique entre le motif de référence testé et le motif courant échantillonné dans l'ellipse corrigée reste supérieure à un seuil de valeur 0.20. En moyenne, vingt tirages aléatoires sont nécessaires pour un temps d'initialisation de l'ordre de la seconde.

Le motif courant étant maintenant reconnu et l'ellipse recalée sur le motif suivi, nous déterminons une consigne $\underline{S}^*$ définie à partir des paramètres de l'ellipse que l'on souhaite obtenir dans l'image (figure 4.43).

Illustrations de l'asservissement visuel

La phase d'initialisation terminée, nous pouvons maintenant asservir en vitesse la position de l'effecteur par rapport au motif suivi dans l'image courante. Pour chaque nouvelle acquisition d'image, la *loi de commande en vitesse* calcule le *torseur cinématique* $\underline{T}$ en comparant la consigne $\underline{S}^*$ (le vecteur des paramètres de l'ellipse à atteindre) avec la mesure courante $\underline{S}$ (les paramètres courants de l'ellipse recalée sur le motif suivi) issue de notre approche de suivi 3D. Le torseur cinématique étant estimé, le *Jacobien inverse du robot* permet de calculer les vitesses articulaires $\underline{\dot{q}}$ à appliquer aux différents axes du robot pour commander les déplacements de la caméra embarquée.

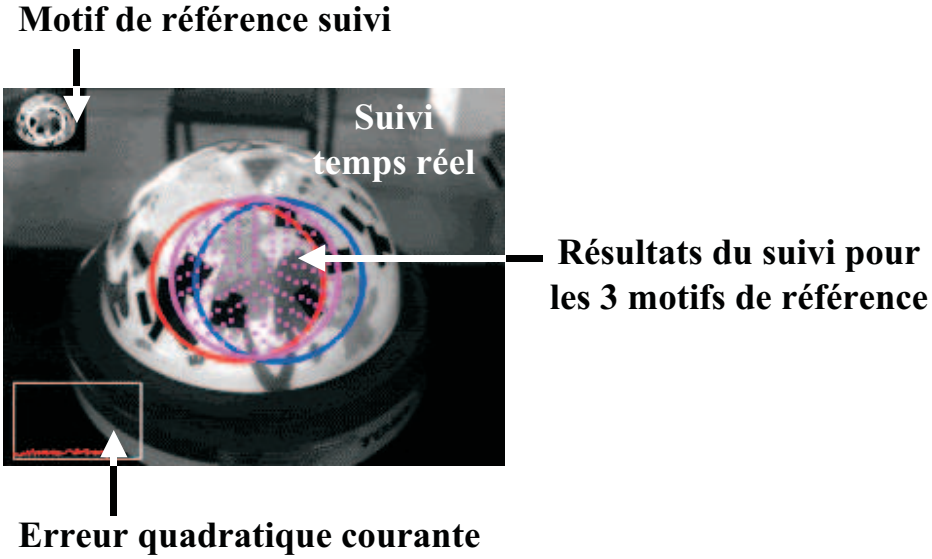


FIGURE 4.47 – visualisation des résultats lors de la commande en vitesse du robot.

Cet algorithme, développé sur une station de travail Silicon Graphics O_2, a un temps d'exécution inférieur à 20 ms. Il fournit en temps réel les informations associées à l'erreur quadratique courante, le motif de référence suivi et la position de l'ellipse pour chacun des trois motifs de référence testés (figure 4.47).

Les figures 4.48 et 4.49 illustrent les résultats de notre algorithme obtenus à des instants différents de l'asservissement (l'objet pouvant se déplacer en même temps que l'effecteur du robot). Les images retenues ont été sélectionnées de manière à montrer la robustesse de notre approche.

Nous remarquons que la caméra suit précisément les déplacements de l'objet et se positionne de façon que le motif suivi reste centré dans l'image (à l'intérieur de l'ellipse définie comme consigne si l'asservissement est correct). L'algorithme de suivi 3D gère avec efficacité les changements de motif de référence et fournit à la loi de commande en vitesse des informations fiables sur les paramètres de l'ellipse englobant le motif suivi dans l'image courante.

Même si la valeur de l'erreur quadratique courante semble plus importante pour certains déplacements de l'objet devant la caméra (d'après le graphique en surimpression dans le coin inférieur gauche de l'image courante), le suivi est encore assuré. Ces variations d'amplitude de l'erreur quadratique courante s'expliquent par des variations d'apparence du motif de référence suivi et ceci pour deux raisons principales :

1. les variations d'illumination dues à l'éclairage naturel du hall robotique,
2. les mouvements non modélisés lors de la phase d'acquisition de la base d'apprentissage (en particulier, des variations de position en azimut de l'objet par rapport à la caméra).

Bien entendu, si les variations d'apparence sont trop importantes (erreur quadratique courante supérieure à un seuil de perte de suivi égal à 0.50), nous perdrons le suivi du motif actuel dans l'image car il ne coïncidera plus avec l'un des motifs de référence de la base d'apprentissage. En cas de perte de suivi, le déplacement du robot est alors stoppé.

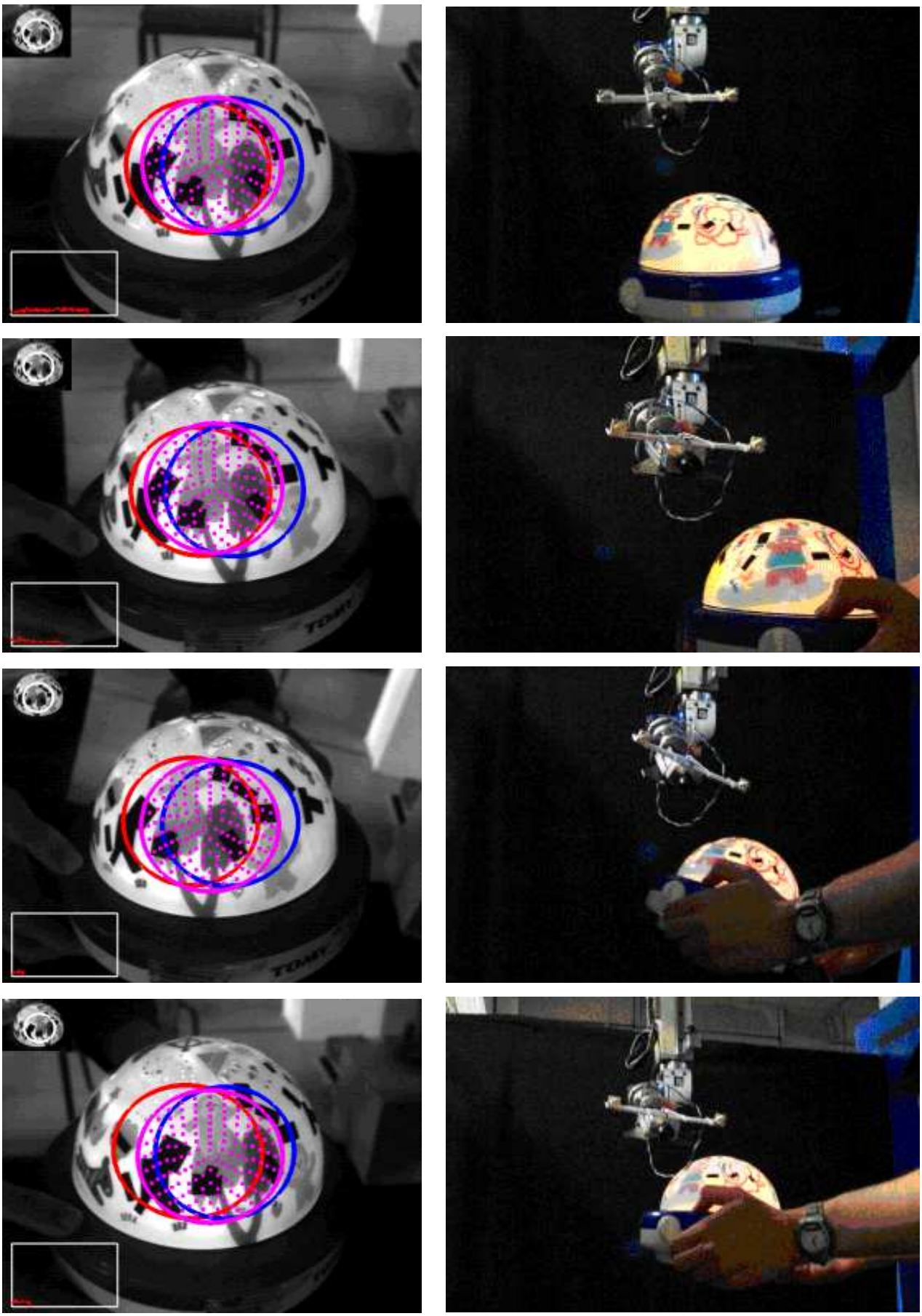


FIGURE 4.48 – résultats de la commande en vitesse du robot portique : séquence 1.

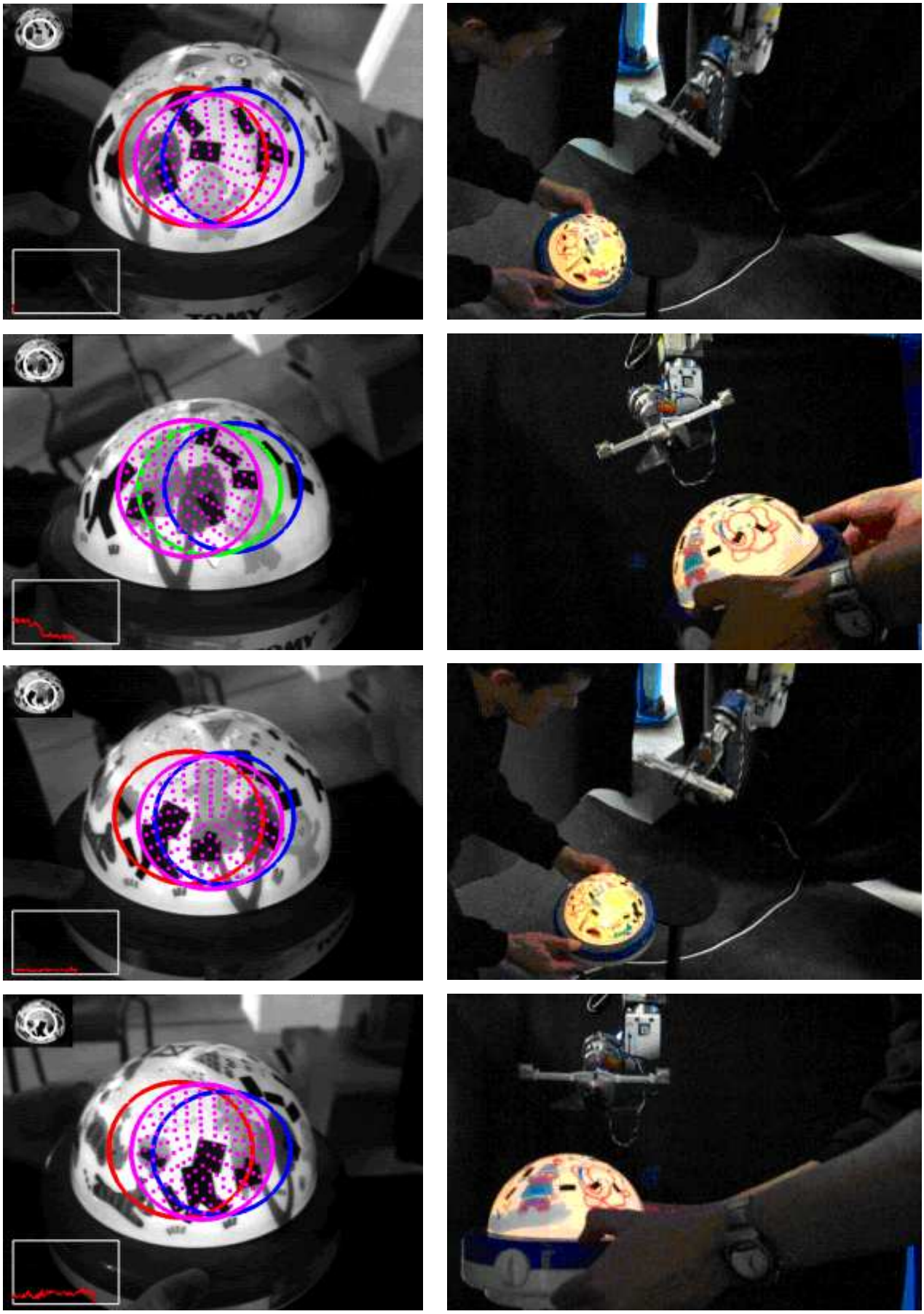
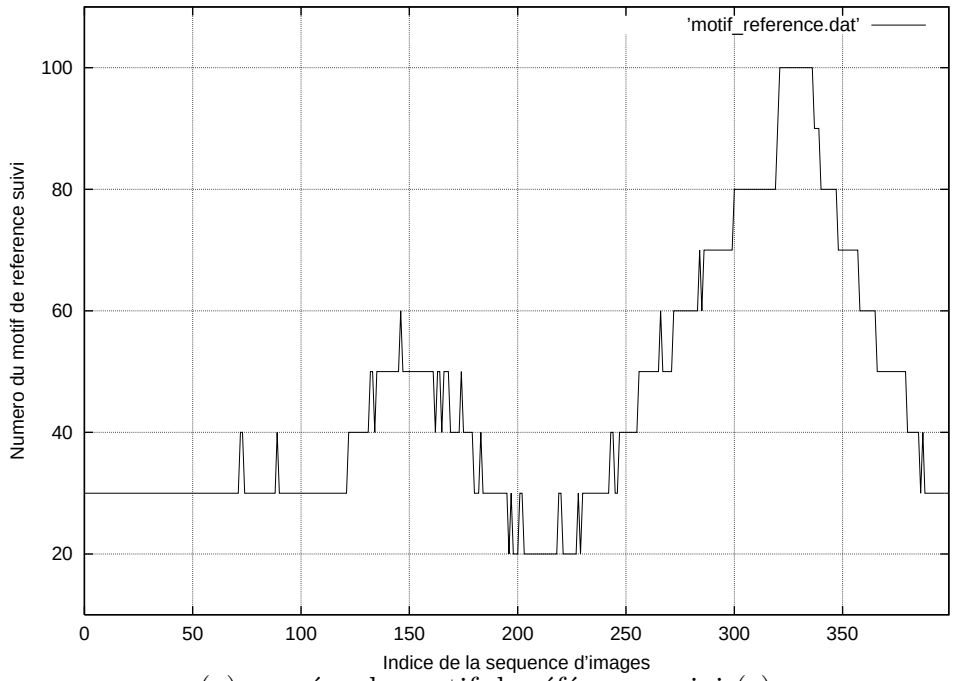
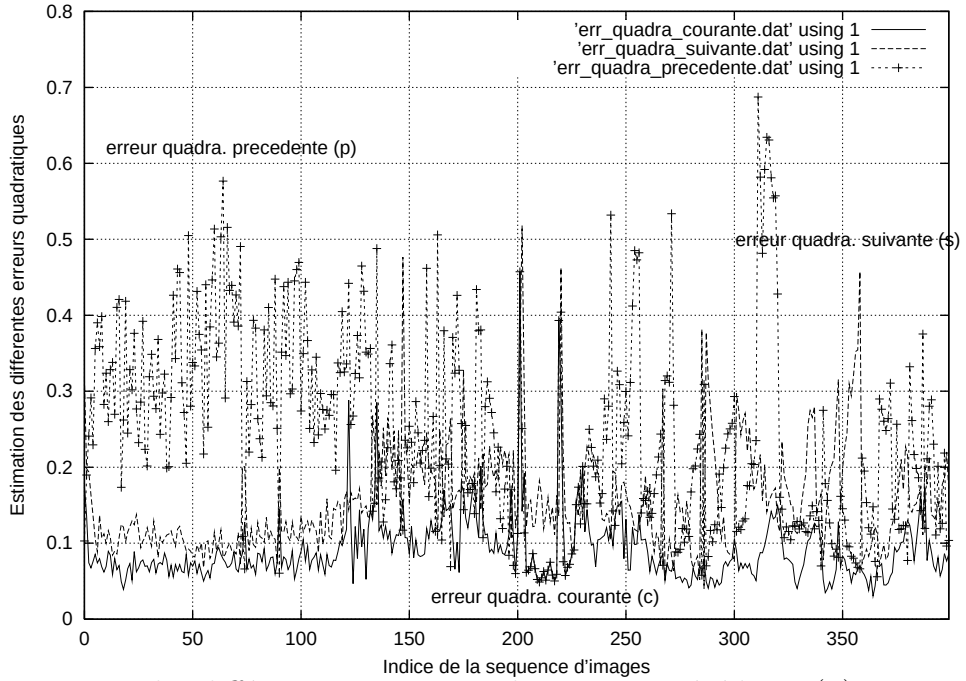


FIGURE 4.49 – résultats de la commande en vitesse du robot portique : séquence 2.

Erreurs quadratiques et motif de référence suivi dans l'image courante



(a) numéro du motif de référence suivi (c).



(b) estimation des différentes erreurs quadratiques : précédente (p), courante (c) et suivante (s).

FIGURE 4.50 – résultats du changement de motif de référence donnés par l'algorithme de suivi 3D en condition réelle.

La figure 4.50 montre, pour chacune des images de la séquence, les valeurs des différentes erreurs quadratiques associées aux trois motifs de référence testés séquentiellement et l'indice

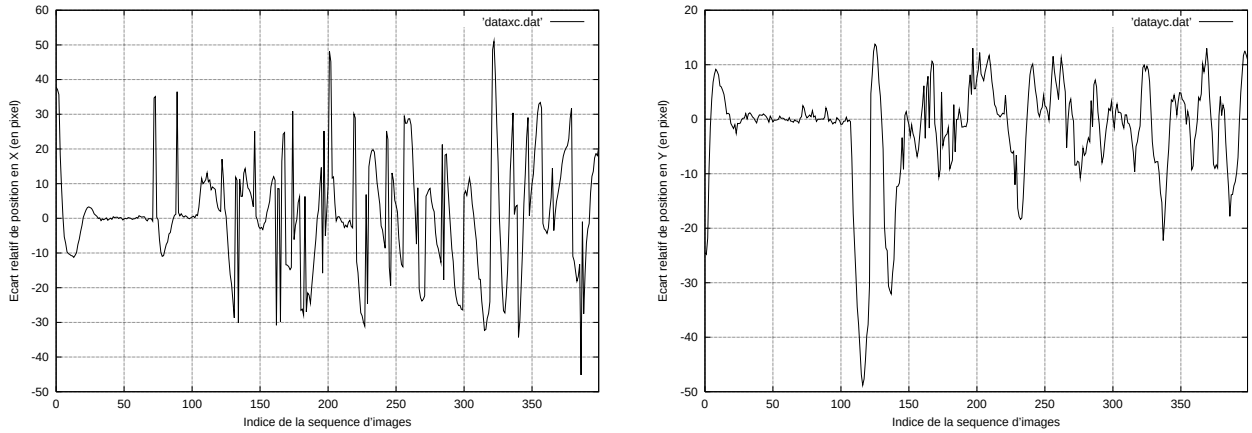
du motif de référence à suivre dans la prochaine image. Le changement de motif de référence s'opère toujours quand l'une des deux erreurs quadratiques précédente (p) ou suivante (s) devient inférieure à l'erreur quadratique courante (c). Les erreurs quadratiques obtenues en simulation (figures 4.44 et 4.45) puis en condition réelle sont du même ordre de grandeur (erreurs précédente et suivante $<$ à 0.70 et erreur courante $<$ à 0.20).

L'augmentation sensible de l'erreur quadratique courante (c) est due principalement aux variations d'éclairement de la scène et de la position caméra/objet comme nous l'avions supposé précédemment.

Paramètres courants de l'ellipse corrigée dans l'image

Pour valider l'asservissement en vitesse de la caméra en fonction des paramètres de l'ellipse de consigne $\underline{S}^*$, nous donnons les valeurs courantes des différents paramètres de l'ellipse corrigée $\underline{S}$ englobant le motif suivi dans chacune des images de la séquence. Cette mesure $\underline{S} = (X_c, Y_c, R_1, \theta)^t$ issue de l'algorithme de suivi 3D pourra être comparée avec la consigne $\underline{S}^*$ qui a été définie, à partir de la taille originale d'une image (768*576 pixels), de la manière suivante :

- *position* centrée dans l'image : $X_c^* = 384$ et $Y_c^* = 288$,
- *orientation* : θ^* = -90 degrés (ou -1.57 radians),
- *échelle* : $R_1^* = 120$ et $R_2^* = k * R_1^*$ par définition.



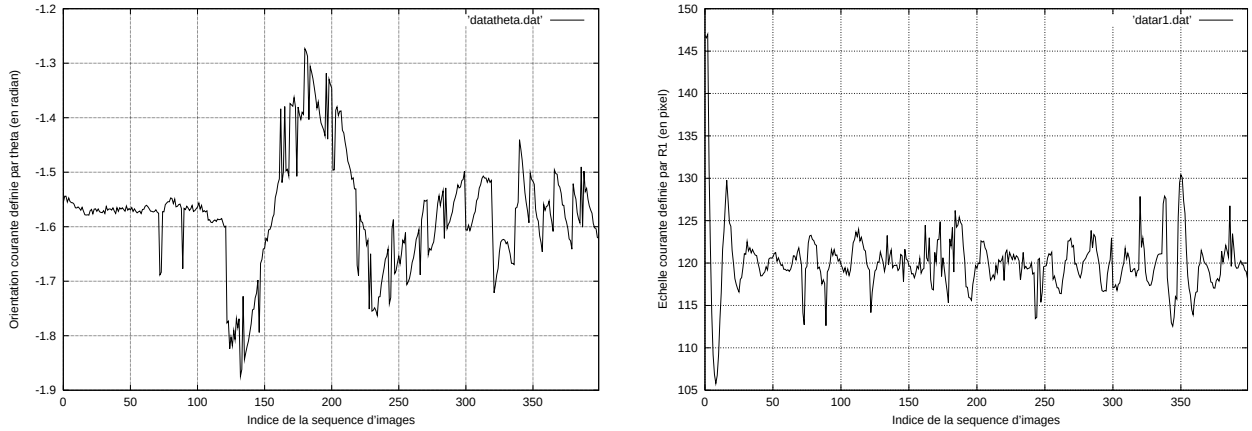
(a) écart relatif en X (en pixel). (b) écart relatif en Y (en pixel).

FIGURE 4.51 – écarts relatifs de position entre les deux ellipses (courante et consigne).

La figure 4.51 décrit l'évolution de la position de l'ellipse courante (X_c, Y_c) par rapport à la position de l'ellipse de consigne (X_c^*, Y_c^*). Pour cela, les écarts relatifs de position ΔX et ΔY entre les deux ellipses ont été calculés de la manière suivante :

– *écart relatif en* X (en pixel) : $\Delta X = X_c$ - $X_c^* = X_c$ - 384,

– *écart relatif en* Y (en pixel) : $\Delta Y = Y_c$ - $Y_c^* = Y_c$ - 288.



(a) orientation courante (en radian). (b) échelle courante (en pixel).

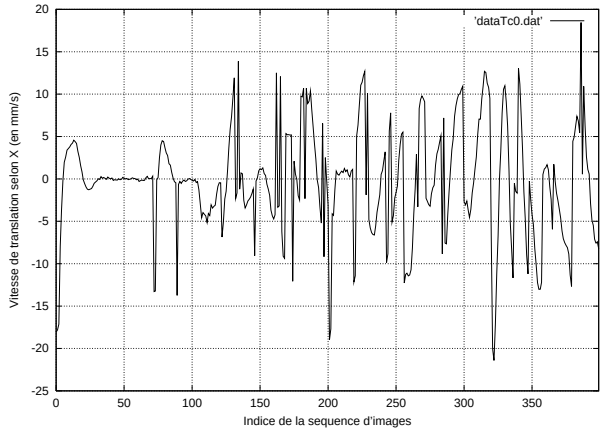
FIGURE 4.52 – orientation et échelle de l'ellipse courante.

La figure 4.52, quant à elle, donne les valeurs courantes des paramètres associés à l'orientation et à l'échelle du motif dans l'image (à l'intérieur de l'ellipse corrigée définie par les paramètres θ et R_1).

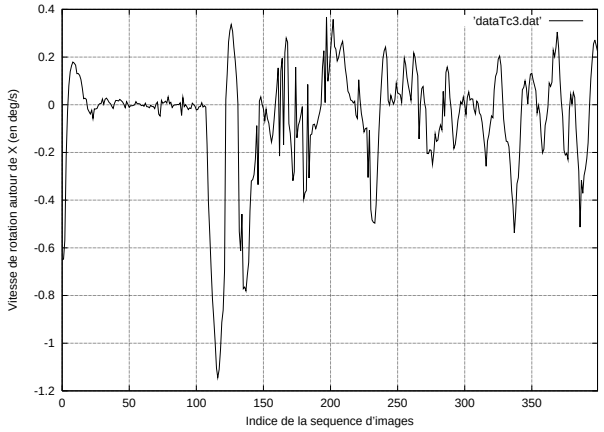
Ces deux figures montrent à la fois les performances (en terme de poursuite et régulation autour de la consigne) et les limites (en terme de temps de réponse) de l'asservissement en vitesse. Nous remarquons que les valeurs des différents paramètres de l'ellipse courante $\underline{S}$ englobant le motif suivi dans chacune des images de la séquence varient autour des valeurs des paramètres de l'ellipse $\underline{S}^*$ définie comme consigne. Les fortes variations des paramètres de l'ellipse courante autour des valeurs de consigne s'expliquent du fait que l'objet peut se déplacer plus rapidement que la caméra embarquée dont les mouvements sont limités en amplitude (en fonction des vitesses articulaires maximales autorisées). Dans ce cas, l'algorithme assure toujours le suivi du motif qui n'est plus centré dans l'image et il faut plusieurs itérations à la loi de commande en vitesse pour recentrer le motif dans l'image en fonction de la consigne.

Torseur cinématique

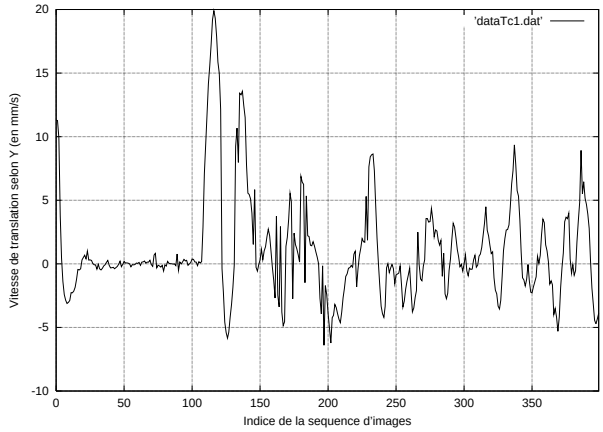
En complément d'information (figure 4.53), nous allons donner les différentes valeurs des composantes du torseur cinématique $\underline{T}$ calculées par la loi de commande en vitesse et ceci pour chacune des images de la séquence.



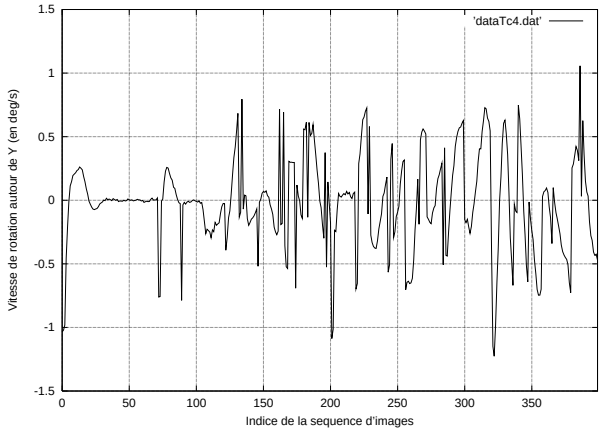
(a) vitesse de translation V_X (en mm/s).



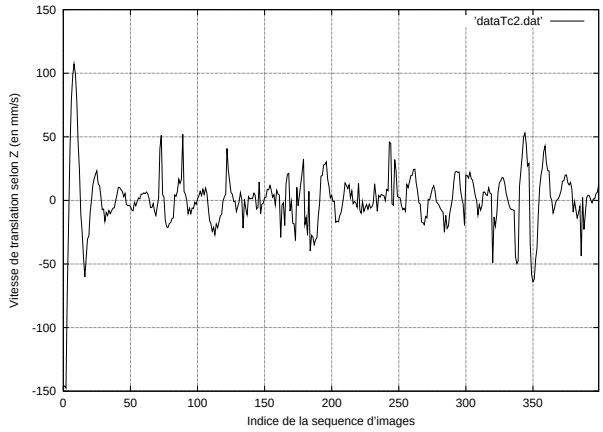
(b) vitesse de rotation Ω_X (en deg/s).



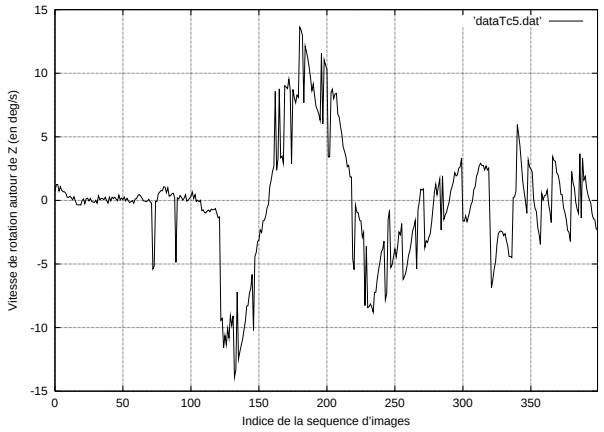
(a) vitesse de translation V_Y (en mm/s).



(b) vitesse de rotation Ω_Y (en deg/s).



(a) vitesse de translation V_Z (en mm/s).



(b) vitesse de rotation Ω_Z (en deg/s).

FIGURE 4.53 – composantes du torseur cinématique.

Nous avons vu que par définition, le torseur cinématique correspond à la vitesse de la caméra dans un repère absolu (le repère physique du robot) :

$$\underline{T} = \begin{pmatrix} \vec{V} \\ \vec{\Omega} \end{pmatrix} \tag{4.18}$$

- $\vec{V} = (V_X, V_Y, V_Z)^T$ représente la vitesse de translation du centre du repère caméra dans un repère absolu.
- $\vec{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)^T$ est le vecteur vitesse de rotation du repère caméra dans un repère absolu.

4.5 Conclusion

Dans ce chapitre, nous avons présenté quatre applications dédiées au suivi 3D. La première décrit le suivi 3D d'un visage avec détection automatique d'un motif dans l'image courante. Les variations d'aspect sont dues principalement à des rotations en site du visage de l'individu (du profil gauche au profil droit) mais l'algorithme de suivi reste robuste pour de faibles variations d'azimut et des changements d'expression.

Nous nous sommes ensuite intéressés au développement d'une commande en position "plus ou moins" pour contrôler la variation angulaire en site de la table à déplacement micrométrique. L'information retenue pour élaborer cette commande est donnée par l'algorithme de suivi 3D et correspond à l'indice du motif de référence suivi dans l'image courante. Le déplacement relatif en site de la table par rapport à sa position angulaire courante est calculé en comparant l'indice du motif de référence actuellement suivi avec l'indice du motif de référence que l'on souhaite atteindre et ceci à chaque nouvelle acquisition d'image.

La troisième application est la navigation d'un bras robotique autour d'un objet connu dont la trajectoire est imposée lors de la phase d'acquisition de la base d'apprentissage modélisant l'objet. Il s'agit ici d'utiliser le résultat du suivi 3D (l'indice du motif de référence suivi) dans l'image courante et la commande en position développée précédemment pour contrôler le déplacement du bras robotique et l'amener à une position où le motif courant suivi dans l'image correspondra au motif de référence à atteindre.

Nous avons finalement conclu ces différentes expériences par le développement d'une commande en vitesse. Cette dernière permet d'asservir le déplacement du bras robotique pour obtenir, dans l'image courante, une ellipse dont la forme (en position, échelle et orientation) est définie comme consigne. A la différence de la commande en position, l'objet peut maintenant se déplacer devant la caméra embarquée sur l'effecteur du robot.

Toutes ces expérimentations ont été menées pour tester la fiabilité et la robustesse de notre algorithme de suivi 3D qui peut être utilisé dans des applications de vision artificielle très variées (de la visioconférence à l'asservissement visuel d'un robot).

Conclusion

Les travaux présentés dans ce mémoire ont porté sur le suivi automatique d'objets 3D dans une séquence d'images. C'est une solution de suivi, temps réel, basée sur l'apparence et qui gère les changements d'aspect du motif. Pour cela, l'objet 3D est représenté par une collection d'images 2D appelées vues de référence. Cette technique de suivi 3D comprend deux étapes.

Une phase d'apprentissage hors ligne est dédiée aux calculs de deux matrices d'interaction. La première matrice notée A lie les variations de la différence de niveaux de gris entre le motif de référence suivi et le motif échantillonné dans une zone d'intérêt (une ellipse dans notre cas) après correction de sa position à son mouvement fronto parallèle (mouvement parallèle au plan image). Sous l'hypothèse d'un tel mouvement, le motif suivi ne subit pas de modification majeure. Par contre, sa position, son orientation planaire et sa taille peuvent changer. Ces mouvements fronto parallèles sont caractérisés par quatre paramètres (X_c, Y_c, R_1, θ) correspondant aux paramètres de l'ellipse utilisée comme zone d'intérêt dans la phase de suivi. La seconde matrice d'interaction notée B, quant à elle, relie les variations d'apparence du motif suite à un changement d'orientation par rapport au capteur. Ces mouvements correspondant à des variations en site et azimut de l'objet devant la caméra sont caractérisés par deux angles (α et β). Une des améliorations dans la méthode de calcul des matrices d'interaction est de remplacer l'approximation Jacobienne proposée par Hager *et al.* par une approximation par hyperplans. Nous avons alors montré comment une approximation précalculée peut être utilisée dynamiquement. Avec ces améliorations, la vitesse de convergence est multipliée par 3 ou 4, comparée à l'algorithme de suivi de Hager. Basée sur ces mêmes travaux, une méthode de seuillage adaptatif pour le traitement des occultations a été développée.

Une étape en ligne consiste à prédire la position du motif dans l'image (en position, échelle et orientation), à multiplier la différence entre le motif observé à l'endroit prédit avec le motif de référence qui doit être suivi par la première matrice d'interaction pour corriger cette prédiction.

Une nouvelle différence entre le motif courant corrigé et le motif de référence multipliée par la deuxième matrice d'interaction nous donne les variations d'aspect du motif suivi par rapport au motif de référence le plus proche dans la collection d'images. Nous pouvons ainsi changer de motif de référence et continuer le suivi du nouveau motif dans la prochaine image. C'est une méthode généraliste car les variations de position du motif dans l'image correspondent aux variations des paramètres d'une transformation géométrique dont le choix est à l'initiative du programmeur suivant l'aspect et le volume de l'objet à suivre (une transformation affine dans notre cas). Un des avantages de cette méthode de suivi est de ne pas utiliser une phase d'exploration autour de la position prédite du motif dans l'image puisque l'écart de position du motif entre deux images reste compatible avec les variations apprises lors de la phase d'apprentissage. De plus, c'est un algorithme très peu coûteux en temps de calcul (multiplication d'une matrice par un vecteur) permettant une mise en oeuvre temps réel (inférieure à 20 ms sur une station de travail Silicon Graphics O_2).

Les premiers essais de suivi ayant donné des résultats plutôt encourageants sur des objets rigides, notre souhait fut ensuite de développer des algorithmes de suivi 3D dédiés aux domaines de la visioconférence et de la robotique manufacturière. Le premier algorithme permet donc le suivi 3D d'un visage avec détection et sélection automatique d'un motif dans l'image courante. Les variations d'aspect sont dues principalement à des rotations en site du visage de l'individu (du profil gauche au profil droit) mais l'algorithme de suivi reste robuste pour de faibles variations d'azimut et des changements d'expression. De plus, la détection de la perte de suivi du motif courant permet d'avoir une application autonome qui se réinitialise automatiquement pour rechercher un nouveau motif dans l'image. L'une des applications envisagées était la visioconférence où la transmission et la compression des données sont des problèmes importants. Pour diminuer la taille des données utilisées, il serait intéressant de transmettre soit le motif suivi à l'intérieur de la zone d'intérêt soit l'image de différence entre le motif de référence et le motif courant échantillonné après correction pour reconstruire l'expression faciale de l'individu à partir de la collection d'images de référence qui aurait été transmise à l'interlocuteur à l'initialisation de l'application (lors de la première connexion). Ces différents résultats nous ont amené à conclure que l'image de différence fournit autant d'information dans son contenu que l'image courante. De plus cet algorithme séquentiel a fait l'objet d'une parallélisation basée sur des squelettes fonctionnels à la demande des personnes du thème *Architectures des Machines de Perception* du laboratoire.

Nous nous sommes ensuite intéressés au développement d'une commande en position "plus ou moins" pour contrôler la variation angulaire en site de la table à déplacement micrométrique. L'information retenue pour élaborer cette commande est donnée par l'algorithme de suivi 3D et correspond à l'indice du motif de référence suivi dans l'image courante. Le déplacement relatif en site de la table par rapport à sa position angulaire courante est calculé en comparant l'indice du motif de référence actuellement suivi avec l'indice du motif de référence que l'on souhaite atteindre et ceci à chaque nouvelle acquisition d'image. Cette étape intermédiaire était importante pour valider l'intégration de notre algorithme de suivi 3D dans une boucle d'asservissement avant de travailler sur le robot portique du laboratoire.

Les deux derniers développements sont donc dédiés au domaine de la robotique manufacturière. Le premier concerne la navigation d'un bras robotique autour d'un objet connu dont la trajectoire est imposée lors de la phase d'acquisition de la base d'apprentissage modélisant l'objet. Il s'agit ici d'utiliser le résultat du suivi 3D (l'indice du motif de référence suivi) dans l'image courante et la commande en position développée précédemment pour contrôler le déplacement du bras robotique et l'amener à une position où le motif courant suivi dans l'image correspondra au motif de référence à atteindre. Nous avons finalement conclu ces expérimentations par le développement d'une commande en vitesse. Cette dernière permet d'asservir le déplacement du bras robotique pour obtenir, dans l'image courante, une ellipse dont la forme (en position, échelle et orientation) est définie comme consigne. A la différence de la commande en position, l'objet peut maintenant se déplacer devant la caméra embarquée sur l'effecteur du robot.

Dans ces différentes applications, nous n'utilisons plus les matrices d'interaction B pour gérer les changements de motif de référence lors de la phase de suivi, mais plutôt la comparaison des erreurs quadratiques de la différence de niveaux de gris entre le motif échantillonné dans l'ellipse corrigée et les différents motifs de référence testés (motif actuellement suivi et ses plus proches voisins dans la collection d'images de référence). Le motif de référence donnant l'erreur quadratique la plus faible sera considéré comme le nouveau motif de référence à suivre dans la prochaine image. Dans le cas du suivi de visage, cela s'explique par le fait que le visage est un objet déformable avec des variations d'expression et que tous les mouvements du visage d'un individu ne peuvent être appris à partir d'une simple séquence vidéo d'apprentissage. Dans le cas de l'asservissement visuel du bras robotique autour de l'objet, nous ne pouvons plus négliger la profondeur de l'objet par rapport à la distance caméra/objet (utilisation d'une caméra à focale courte entraînant des distorsions importantes du motif représentant l'une des

apparences de l'objet 3D dans l'image).

Notre souhait est de pouvoir améliorer les performances du suivi 3D suivant les applications envisagées. Pour augmenter la robustesse de l'algorithme de suivi, plusieurs points devront être traités :

- la sélection automatique des meilleurs points à échantillonner dans la zone d'intérêt du motif de référence lors de la phase d'apprentissage pour être encore plus robuste aux déplacements de forte amplitude du motif entre deux acquisitions d'image durant la phase de suivi,
- les variations de luminance : être capable de caractériser et de modéliser la réflectance de la surface de l'objet supposée lambertienne pour pouvoir travailler quelles que soient les conditions d'éclairement,
- la gestion du passage d'un motif de référence à un autre quand on perd l'information de perspective. Lors du suivi d'une face d'un cube positionnée parallèlement au plan image, les variations du motif ne nous permettent pas de dissocier précisément une rotation en site du cube dans un sens ou dans l'autre. Une sélection meilleure des points échantillonnés ou le changement de la transformation géométrique peuvent être la solution mais augmenter le nombre de paramètres de la transformation géométrique pour augmenter le nombre de degrés de liberté la rend beaucoup plus sensible aux bruits.
- l'utilisation d'une caméra couleur pour pouvoir tester l'algorithme et profiter de la richesse des informations,
- l'utilisation d'une caméra CMOS pour diminuer le temps d'acquisition entre deux images (actuellement 45 images par seconde) et donc traiter des mouvements de plus grande amplitude,
- la sélection des meilleures vues de référence suivant le volume de l'objet 3D à modéliser et les mouvements souhaités de l'objet devant la caméra,
- enfin, être capable d'assurer le suivi d'un motif dans le cas d'un changement de fond dans l'image en conservant la zone d'échantillonnage sur l'objet durant la phase d'apprentissage (mais cela limitera l'amplitude des mouvements du motif entre deux acquisitions d'image si nous n'utilisons pas de caméra CMOS).

Annexe A

Rappels sur la résolution des systèmes d'équations linéaires

Dans cette annexe, nous allons présenter de façon succincte les différentes méthodes de résolution des systèmes d'équations linéaires dont la *décomposition en valeurs singulières* **SVD** utilisée dans notre algorithme pour l'estimation des *matrices d'interaction* lors de la phase d'apprentissage hors ligne.

A.1 Formulation matricielle d'un système d'équations linéaires

Un système de n équations linéaires à n inconnues est de la forme :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \end{cases}$$

où les a_{ij} sont les coefficients du système, les x_i les inconnues et les b_i les termes constants. Un tel système peut s'écrire sous la forme matricielle suivante :

$$\mathbf{Ax} = \mathbf{b}$$

$\mathbf{A}$ est une *matrice carrée* de dimensions $(n \times n)$, $\mathbf{x}$ et $\mathbf{b}$ sont des *vecteurs colonnes* de dimension n.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

A.1.1 Cas d'une matrice régulière

Une matrice carrée $\mathbf{A}$ est dite *inversible* ou *régulière* s'il existe une matrice carrée $\mathbf{A^{-1}}$ (appelée *matrice inverse*) telle que :

$$\mathbf{A} \times \mathbf{A^{-1}} = \mathbf{A^{-1}} \times \mathbf{A} = \mathbf{I} \quad (\textit{matrice unité})$$

La matrice inverse $\mathbf{A^{-1}}$ n'existe que si le *déterminant* de la matrice $\mathbf{A}$ noté $\mathbf{det}(\mathbf{A})$ est différent de zéro.

Si la matrice $\mathbf{A}$ est régulière, nous avons en multipliant à gauche par $\mathbf{A^{-1}}$ le système suivant :

$$\mathbf{A^{-1}Ax} = \mathbf{A^{-1}b}$$

Soit :

$$\mathbf{x} = \mathbf{A^{-1}b}$$

Traitons l'exemple suivant :

Soit le système de 2 équations à 2 inconnues :

$$\begin{cases} 2x_1 + 3x_2 = 9 \\ x_1 - x_2 = 2 \end{cases}$$

Nous avons successivement :

$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 9 \\ 2 \end{pmatrix}$$

$$\mathbf{det}(\mathbf{A}) = 2 \times (-1) - 3 \times 1 = -5$$

$$\mathbf{A^{-1}} = -\frac{1}{5} \begin{pmatrix} -1 & -3 \\ -1 & 2 \end{pmatrix}$$

$$\mathbf{x} = -\frac{1}{5} \begin{pmatrix} -1 & -3 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 9 \\ 2 \end{pmatrix} = -\frac{1}{5} \begin{pmatrix} -15 \\ -5 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Nous trouvons finalement $x_1 = 3$ et $x_2 = 1$.

A.1.2 Cas d'une matrice singulière

Par définition, la matrice $\mathbf{A}$ est dite *singulière* si $\mathbf{det}(\mathbf{A}) = 0$. Dans ce cas, la matrice inverse $\mathbf{A^{-1}}$ n'existe pas.

Lorsque la matrice est singulière, deux cas sont à envisager :

1. **Système indéterminé** :

   S'il est possible d'exprimer p équations en fonction des autres, le système admet une infinité de solutions. Nous pouvons retenir le vecteur $\mathbf{x}$ qui a la plus faible norme. L'ensemble des solutions forme un sous-espace de dimension $r = n - p$ dans l'espace de dimension n. Le nombre r est le *rang* de la matrice.

   Soit l'exemple suivant :

   $$\begin{cases} x_1 + x_2 = 3 \\ 2x_1 + 2x_2 = 6 \end{cases}$$

   Le déterminant vaut : $1 \times 2 - 1 \times 2 = 0$. La matrice est bien singulière.

   La deuxième équation est égale à la première multipliée par 2. Il n'y a en fait qu'une seule équation : $x_1 + x_2 = 3$. C'est l'équation d'une droite (espace de dimension 1) dans le plan(x_1, x_2). La matrice est de rang 1.

2. **Système impossible** :

   Si les équations ne peuvent pas être exprimées les unes en fonction des autres, le système n'admet aucune solution. Nous pouvons cependant calculer un vecteur $\mathbf{x}$ tel que la norme du vecteur $\mathbf{Ax} - \mathbf{b}$ soit minimale (bien que non nulle). Ce vecteur constitue la meilleure approximation de la solution au sens des *moindres carrés* (régression linéaire).

   Soit l'exemple suivant :

   $$\begin{cases} x_1 + x_2 = 3 \\ 2x_1 + 2x_2 = 8 \end{cases}$$

   La deuxième équation divisée par 2 donnerait $x_1 + x_2 = 4$, ce qui est incompatible avec la première équation. Le système n'a pas de solution.

A.2 Résolution des systèmes d'équations linéaires

Les principales méthodes de résolution des systèmes d'équations linéaires sont les suivantes :

1. **Pour une matrice régulière** :
   - La méthode de Gauss-Jordan,
   - La décomposition **LU**,
   - La décomposition **QR**,
   - *La décomposition en valeurs singulières* **SVD**.
2. **Pour une matrice symétrique définie positive** :
   - La méthode de Cholesky.
3. **Pour une matrice singulère** :
   - *La décomposition en valeurs singulières* **SVD**.

A.2.1 Méthode de Gauss-Jordan

La méthode de Gauss-Jordan calcule simultanément la matrice inverse $\mathbf{A^{-1}}$ et le vecteur solution $\mathbf{x}$. S'il y a plusieurs systèmes à résoudre, nous pouvons réunir tous les vecteurs de termes constants $\mathbf{b_i}$ dans une matrice unique $\mathbf{B}$ (chaque vecteur constituant une colonne de la matrice). L'application de l'algorithme Gauss-Jordan fournit alors une matrice $\mathbf{X}$ dont la colonne i constitue la solution du i^{eme} système.

A.2.2 Décomposition LU

Cette méthode exprime la matrice $\mathbf{A}$ sous forme du produit d'une matrice triangulaire inférieure $\mathbf{L}$ à diagonale unité par une matrice triangulaire supérieure $\mathbf{U}$:

$$\mathbf{A} = \mathbf{LU}$$

Par exemple, pour $n = 4$, nous avons :

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

Le système devient :

$$\mathbf{LUx} = \mathbf{b}$$

Soit :

$$\mathbf{Ly} = \mathbf{b} \quad (1)$$
$$\mathbf{Ux} = \mathbf{y} \quad (2)$$

Nous résolvons le système (1) pour trouver le vecteur $\mathbf{y}$, puis le système (2) pour trouver le vecteur $\mathbf{x}$. La résolution est facilitée par la forme triangulaire des matrices.

La méthode **LU** permet aussi de calculer le déterminant de la matrice $\mathbf{A}$, qui est égal au produit des éléments diagonaux de la matrice $\mathbf{U}$, puisque $\mathbf{det}(\mathbf{A}) = \mathbf{det}(\mathbf{L}) \times \mathbf{det}(\mathbf{U})$ et $\mathbf{det}(\mathbf{L}) = 1$ (rappelons que le déterminant d'une matrice triangulaire est égal au produit de ses éléments diagonaux).

A.2.3 Décomposition QR

Cette méthode exprime la matrice $\mathbf{A}$ sous forme d'un produit d'une matrice orthogonale $\mathbf{Q}$ par une matrice triangulaire supérieure $\mathbf{R}$:

$$\mathbf{A} = \mathbf{QR}$$

Le système devient :

$$\mathbf{QRx} = \mathbf{b}$$

Soit, en multipliant à gauche par la matrice transposée $\mathbf{Q^T}$:

$$\mathbf{Q^T QRx} = \mathbf{Q^T b}$$

Soit :

$$\mathbf{Rx} = \mathbf{Q^T b}$$

Comme la transposée de la matrice orthogonale $\mathbf{Q}$ notée $\mathbf{Q^T}$ est égale à son inverse $\mathbf{Q^{-1}}$, le produit $\mathbf{Q^T Q} = \mathbf{I}$ (matrice unité).

Nous résolvons ce dernier système en tenant compte de la forme triangulaire de la matrice $\mathbf{R}$.

Remarque : la décomposition **QR** peut s'appliquer à une matrice rectangulaire de dimensions $(n \times m)$ avec $n > m$. C'est le cas le plus fréquent dans les situations expérimentales, où

nous possédons plus d'équations que d'inconnues. Nous parlons alors de système surdéterminé. Dans ce cas, la matrice **Q** est une matrice colonne-orthogonale de dimensions $(n \times m)$ et la matrice **R** de dimensions $(m \times m)$. Pour un système $\mathbf{Ax} = \mathbf{b}$, la solution retournée est celle des *moindres carrés*.

A.2.4 Décomposition en valeurs singulières SVD

La décomposition en valeurs singulières (*singular value decomposition* **SVD**) est une technique utilisée dans le calcul et l'analyse des matrices car elle présente l'avantage d'être beaucoup plus robuste aux erreurs numériques. Cette méthode est extrêmement efficace pour résoudre aussi bien les systèmes linéaires inversibles, surdéterminés ou mal conditionnés de taille moyenne.

La décomposition en valeurs singulières exprime la matrice **A** sous la forme :

$$\mathbf{A} = \mathbf{USV^T}$$

U et **V** sont des matrices orthogonales. **S** est une matrice diagonale. Ses termes diagonaux s_{ii}, tous positifs ou nuls, sont les *valeurs singulières* de **A**. Le rang de **A** est égal au nombre de valeurs singulières non nulles.

Par exemple, pour $n = 4$, nous avons :

$$\mathbf{S} = \begin{pmatrix} s_{11} & 0 & 0 & 0 \\ 0 & s_{22} & 0 & 0 \\ 0 & 0 & s_{33} & 0 \\ 0 & 0 & 0 & s_{44} \end{pmatrix} \quad ou \quad \mathbf{diag(s_{ii})}$$

De plus, nous savons par définition que $[\mathbf{diag(s_{ii})}]^{-\mathbf{1}} = \mathbf{diag}(\frac{\mathbf{1}}{\mathbf{s_{ii}}})$ et que l'inverse d'une matrice orthogonale est égale à sa transposée $\mathbf{V^{-1}} = \mathbf{V^T}$.

1. **Si A est régulière**, tous les $\mathbf{s_{ii}}$ sont > 0. La matrice inverse est donnée par :

$$\mathbf{A^{-1}} = (\mathbf{USV^T})^{-1} = (\mathbf{V^T})^{-1}\mathbf{S^{-1}U^{-1}} = \mathbf{V} \times \mathbf{diag}(\frac{\mathbf{1}}{\mathbf{s_{ii}}}) \times \mathbf{U^T}$$

   Nous en déduisons la solution du système par $\mathbf{x} = \mathbf{A^{-1}b}$.

2. **Si A est singulière**, les expressions précédentes restent valables à condition de remplacer, pour chaque valeur singulière nulle, le terme $\frac{\mathbf{1}}{\mathbf{s_{ii}}}$ par zéro.

Nous montrons que la solution ainsi calculée correspond :

- dans le cas d'un système indéterminé, au vecteur de plus faible norme,
- dans le cas d'un système impossible, à la solution des *moindres carrés*.

Remarque : tout comme la décomposition **QR**, la décomposition en valeurs singulières peut s'appliquer à une matrice rectangulaire de dimensions $(n \times m)$ avec $n > m$. C'est le cas le plus fréquent dans les situations expérimentales, où nous possédons plus d'équations que d'inconnues. Dans ce cas, la matrice $\mathbf{U}$ est une matrice colonne-orthogonale de dimensions $(n \times m)$, les matrices $\mathbf{S}$ et $\mathbf{V}$ de dimensions $(m \times m)$. Pour un système $\mathbf{Ax} = \mathbf{b}$, la solution retournée est celle des *moindres carrés*.

A.2.5 Méthode de Cholesky

Une matrice symétrique $\mathbf{A}$ est dite *définie positive* si, pour tout vecteur $\mathbf{x}$, le produit $\mathbf{x^T Ax}$ est positif.

Pour une telle matrice, nous montrons qu'il est possible de trouver une matrice triangulaire inférieure $\mathbf{L}$ telle que :

$$\mathbf{A} = \mathbf{LL^T}$$

Cette décomposition permet de :

- Calculer la matrice inverse $\mathbf{A^{-1}}$,
- Calculer $\mathbf{det}(\mathbf{A})$, égal au carré du produit des éléments diagonaux de $\mathbf{L}$,
- Résoudre le système $\mathbf{Ax} = \mathbf{b}$ selon un principe analogue à celui de la méthode $\mathbf{LU}$.

Annexe B

Parallélisation d'un algorithme de suivi 3D à l'aide de squelettes fonctionnels

Dans cette annexe, nous allons présenter les travaux menés sur la parallélisation de l'algorithme de suivi 3D d'un visage proposé dans le dernier chapitre de ce mémoire dont la phase de suivi en ligne est illustrée ici par la figure B.1.

Cette parallélisation basée sur des *squelettes fonctionnels* a été développée dans un environnement de programmation parallèle SKiPPER-II [31]. Les *squelettes fonctionnels* sont des schémas de parallélisation qui se présentent sous la forme de *constructeurs génériques paramétrables* [22] [34] [54] [100]. Ils permettent de simplifier la tâche délicate de parallélisation d'une application en déchargeant le plus possible le programmeur des tâches de programmation bas niveau (placement, ordonnancement, gestion des communications, etc.). Cette étude demandée par les personnes du thème *Architecture des Machines de Perception* du laboratoire pour la thèse de Monsieur Rémi Coudarcher [27] a fait l'objet d'un rapport technique interne [28] et deux publications [29] [30].

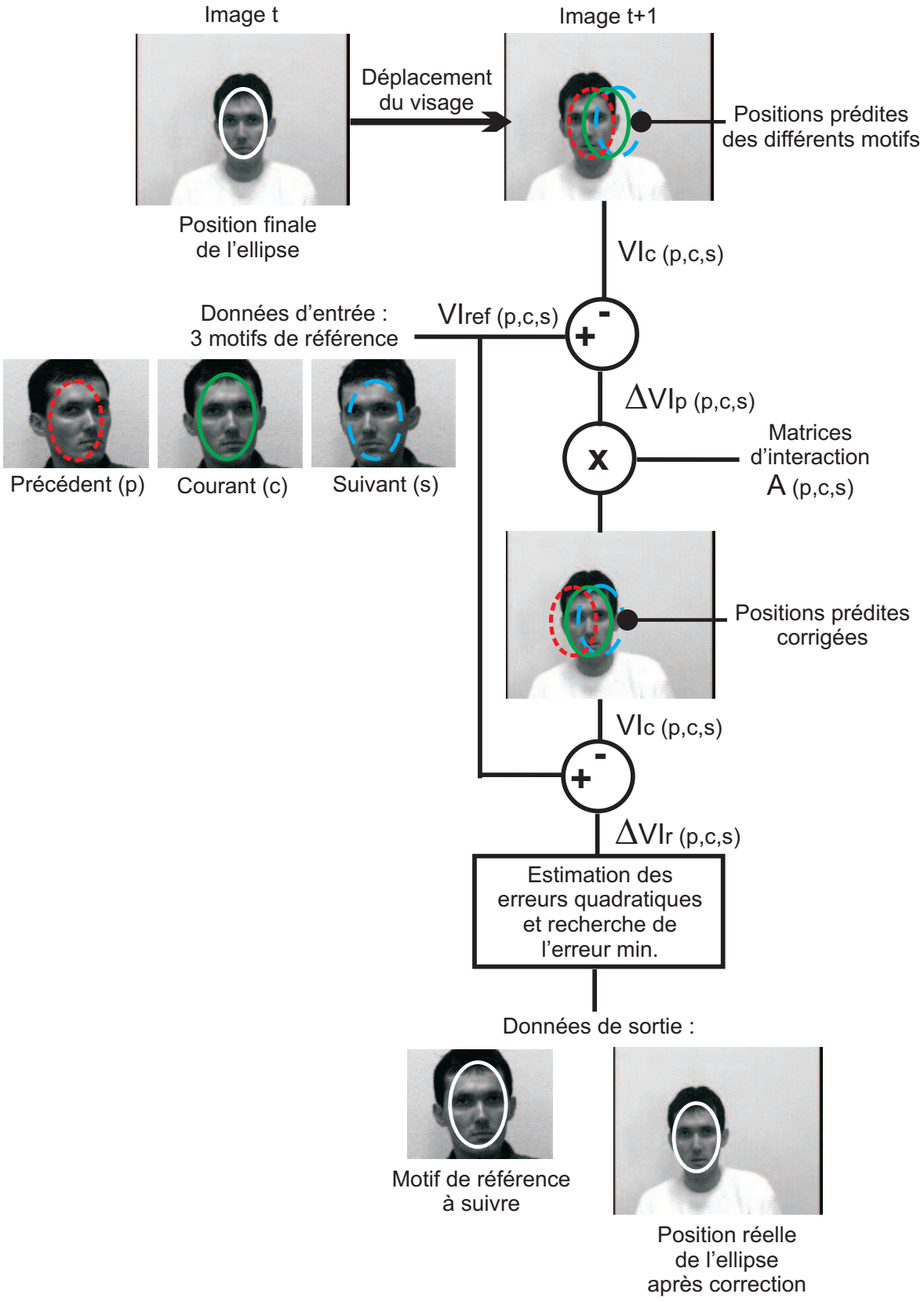


FIGURE B.1 – phase de suivi 3D en ligne à paralléliser.

B.1 Objectif de la parallélisation

Dans cette approche, il ne s'agissait pas de montrer des performances en temps d'exécution puisque notre algorithme s'exécute déjà en temps réel vidéo (< 15ms) mais plutôt de démontrer la possibilité de transformer un algorithme séquentiel (figure B.2) en un algorithme parallèle (figure B.3) avec une procédure d'implantation automatique basée sur des *squelettes fonctionnels*.

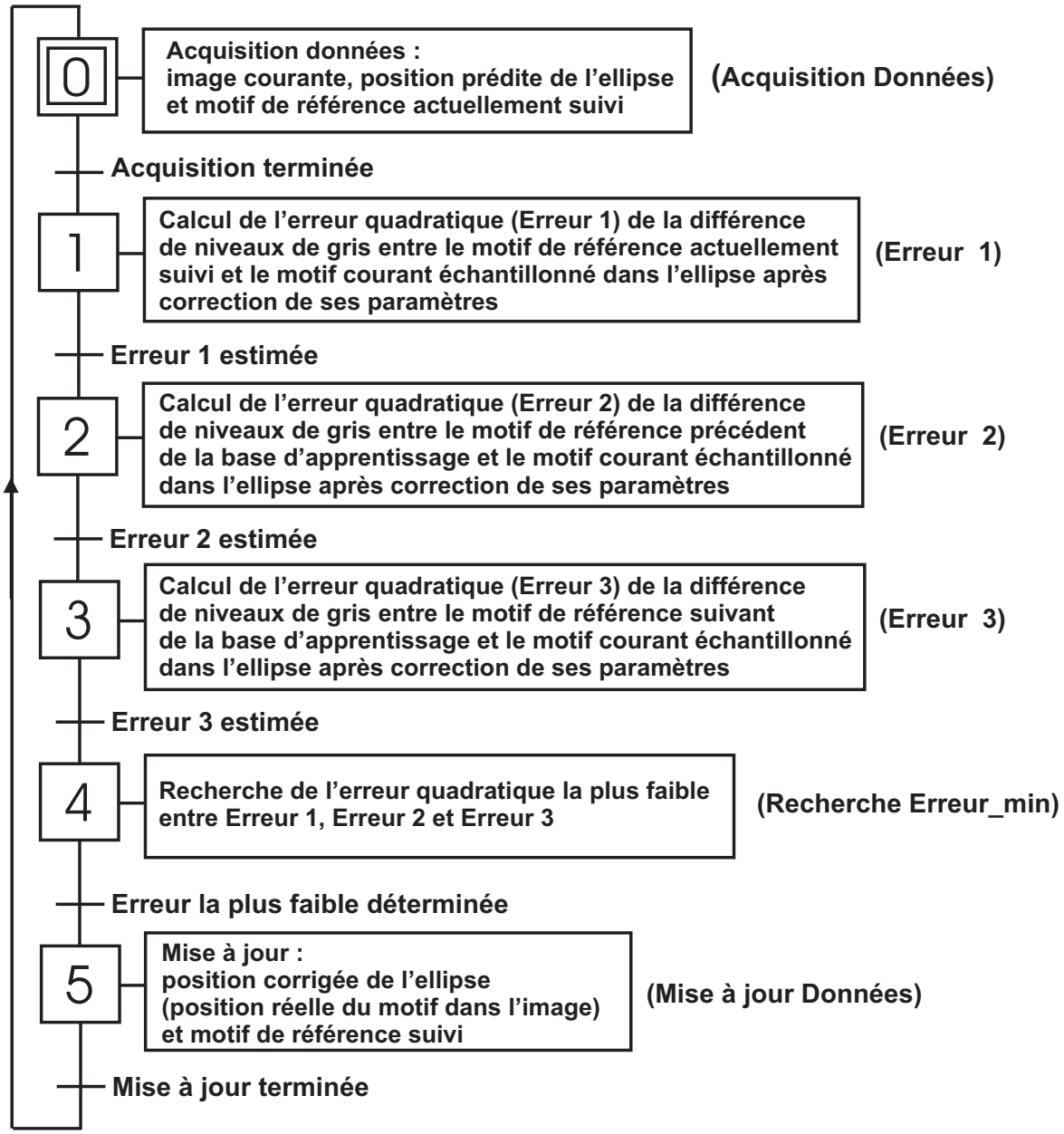


FIGURE B.2 – algorithme séquentiel de suivi 3D à paralléliser.

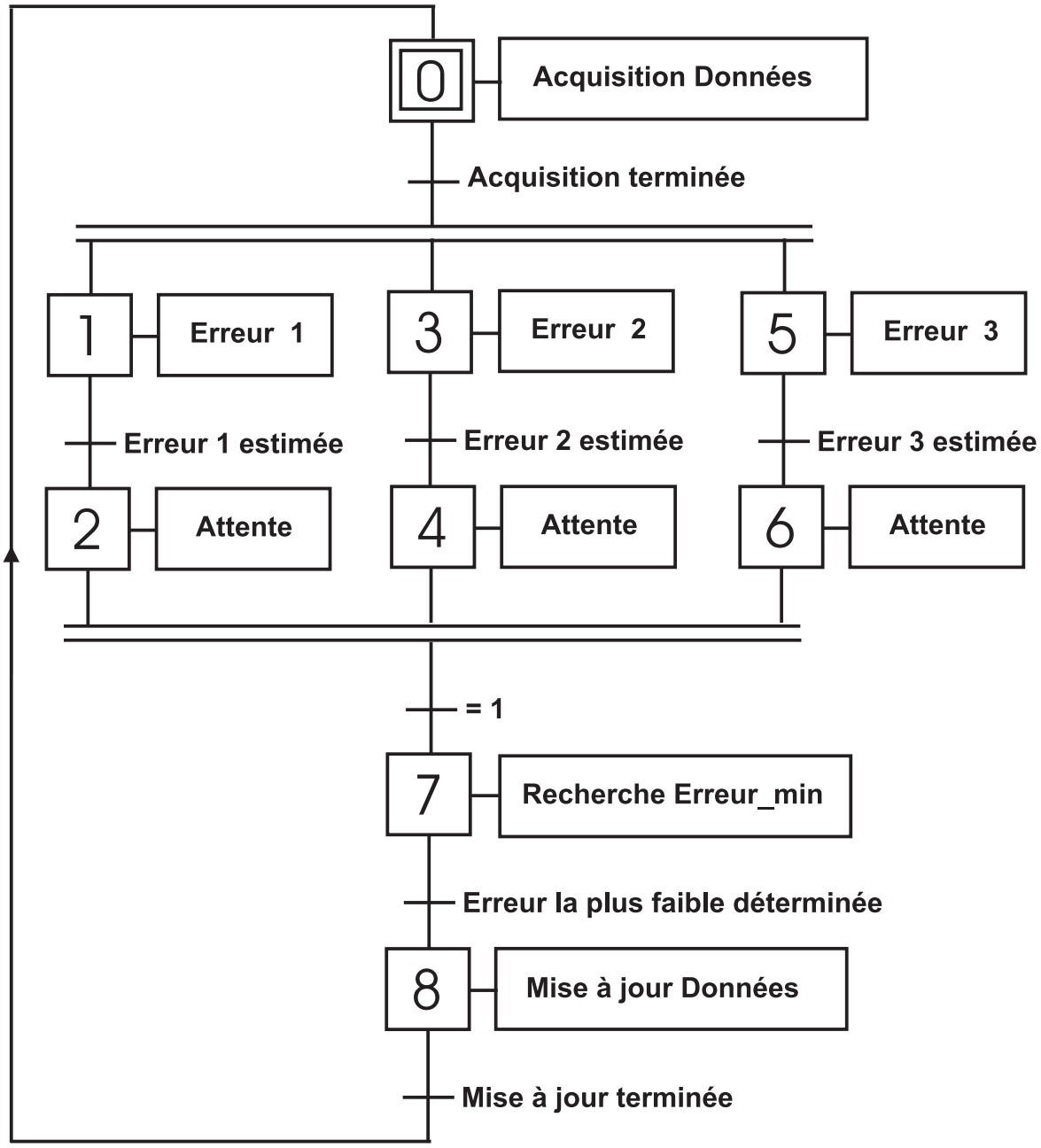


FIGURE B.3 – algorithme de suivi 3D parallélisé.

En effet, notre méthode de suivi 3D est très peu coûteuse en temps de calcul du fait du faible volume de données à traiter (multiplication d'une matrice par un vecteur). Cette phase de suivi en ligne étant l'une des premières applications de vision parallélisée dans l'environnement SKiPPER-II, le découpage de l'algorithme séquentiel à l'aide de *squelettes fonctionnels* nous paraissait le plus approprié et le simple pour débuter.

Le seul intérêt que nous pouvons avancer pour ma thèse est une ouverture dans la méthode de programmation de la phase d'apprentissage hors ligne des *matrices d'interaction*. A la différence de la phase de suivi en ligne, c'est une procédure très coûteuse en temps d'exécution. Par

exemple, dans le cas du suivi 3D en site sur 360 degrés de la canette de soda, il ne faut pas moins de cinq heures d'apprentissage, sur station *Silicon Graphics* O_2, pour calculer l'ensemble des 108 *matrices d'interaction* A_1, A_2 et B (36 *vues de référence* et 3 *matrices d'interaction* par vue pour un motif échantillonné sur 373 points). Rappelons ici que dans notre méthode d'apprentissage, l'estimation des *matrices d'interaction* est obtenue par une minimisation au *sens des moindres carrés* en utilisant un algorithme basé sur une *décomposition en valeurs singulières* d'où l'importance du volume des données à traiter et de la taille des matrices à multiplier. Par conséquent, la diminution des temps de calcul tout en gardant le même volume d'information ne peut se faire qu'en utilisant des méthodes de parallélisation dédiées plus particulièrement, dans notre cas, aux calculs matriciels. La parallélisation d'algorithmes par *squelettes fonctionnels* en est une.

Dans les sections suivantes, nous allons dans un premier temps développer, de manière générale, les différents points à aborder pour traiter le problème de la parallélisation avant de présenter la solution retenue pour notre application et les résultats obtenus en temps d'exécution.

B.2 Généralités sur la parallélisation d'algorithmes par squelettes fonctionnels

Dans le cas d'une *architecture monoprocesseur*, l'algorithme est traduit en un seul programme composé d'une séquence d'instructions s'exécutant séquentiellement sur le processeur. A l'opposé, un algorithme parallèle (devant être implanté par exemple sur une *plate-forme multiprocesseurs*) est un ensemble de séquences d'instructions, chacune étant allouée à un processeur différent, au sein desquelles on introduit des instructions de communication. Celles-ci ont pour but de gérer les dépendances de données entre les instructions s'exécutant sur des processeurs différents.

B.2.1 Définition d'un squelette de parallélisation

D'un point de vue très général, une application parallèle est décomposable en deux entités distinctes mais intimement liées. On trouve d'une part les fonctions séquentielles de calcul liées aux différentes étapes algorithmiques de l'application et d'autre part, aux divers mécanismes de coordination regroupant et liant ces fonctions de calcul. Une application parallèle est donc

un ensemble constitué de **fonctions séquentielles de calcul** et d'un **harnais de communication**. C'est généralement dans cette deuxième partie de code que l'on prend en compte les caractéristiques de la machine (allocation des ressources, synchronisation, communication, etc.).

A partir de là, on peut être tenté de franchir un niveau d'abstraction et d'encapsuler ce couple (**fonctions séquentielles** et **harnais de communication**) sous la forme d'un *constructeur générique réutilisable et paramétrable* par les fonctions de calcul spécifiques à une application donnée. Formalisée par Cole [22] et Skillicorn [101], cette approche débouche sur la notion de **squelettes de parallélisation**. Un squelette est donc une *spécification incomplète* d'une forme de parallélisme commune à un grand nombre d'applications, que le programmeur va *spécialiser* avec ses fonctions de calcul séquentiel. Pour ce programmeur, l'implantation du squelette sur une plate-forme est complètement cachée : les squelettes encapsulent tous les aspects (placement, communication, synchronisation) relatifs à l'expression d'une forme de parallélisme. De ce fait, le programmeur d'applications voit son travail de parallélisation fortement s'amoindrir puisqu'il n'a plus à traiter les aspects bas niveau de l'implantation.

B.2.2 Programmation d'une application parallèle

La programmation d'une application parallèle par *squelettes fonctionnels* se fait alors en deux étapes : une **phase de spécification** et une **phase d'implantation**.

Phase de spécification

Cette phase, indépendante de la machine cible, se réduit au *choix* de un ou plusieurs squelettes (choisis dans une base prédéfinie) suivant l'application à paralléliser.

Cette parallélisation peut s'effectuer soit au *niveau des opérations*, soit au *niveau des données*. Dans le premier cas, il s'agit de partitionner l'algorithme en un ensemble d'actions (ou tâches) s'exécutant de manière concurrente (*parallélisme de tâches*). Dans le second cas, la parallélisation revient à décomposer les données à traiter en différents blocs sur lesquels une même action va être exécutée en parallèle (*parallélisme de données*). Dans les deux cas, ce partitionnement est le résultat d'une analyse approfondie des mécanismes de calcul mis en jeu dans l'application. Cela équivaut à étudier les dépendances de données et à déterminer parmi l'ensemble des actions celles qui sont potentiellement exécutables en parallèle (celles qui n'ont pas de dépendance mutuelle).

Dans cette base prédéfinie de squelettes, il a été choisi le **squelette SCM (Split - Compute - Merge)** pour paralléliser l'algorithme de suivi 3D.
Le *squelette SCM* encapsule les stratégies à *parallélisme de données* dans lesquelles la donnée d'entrée est divisée (*Split*) en un nombre fixe de partitions. Chacun des sous-domaines ainsi généré est alors traité (*Compute*) indépendamment par un processeur. Le résultat final est obtenu par combinaison (*Merge*) (selon une stratégie plus ou moins complexe) des solutions intermédiaires. La figure B.4 décrit un exemple d'implantation d'un tel squelette sur une architecture à n processeurs (avec $n = 4$).

Du coté du *processeur "maître"* (P0), on assiste à un enchaînement séquentiel des opérations suivantes :

1. génération de n partitions de données (X0 à X3) par la fonction spécifique *Split*,
2. distribution de $n - 1$ paquets (X1, X2 et X3) aux autres *processeurs du réseau* (P1, P2 et P3),
3. application de la fonction *Compute* sur le paquet non distribué (X0),
4. collecte des $n - 1$ résultats intermédiaires (Y1, Y2 et Y3),
5. fusion des résultats (Y0 à Y3) par la fonction spécifique *Merge*.

Les autres *processeurs "utilisateurs"* (P1, P2 et P3) exécutent, quant à eux, la séquence suivante :

1. réception d'un paquet de données (X1, X2 ou X3),
2. traitement local du paquet par la fonction *Compute*,
3. envoi du résultat obtenu (Y1, Y2 ou Y3) vers le *processeur "maître"*.

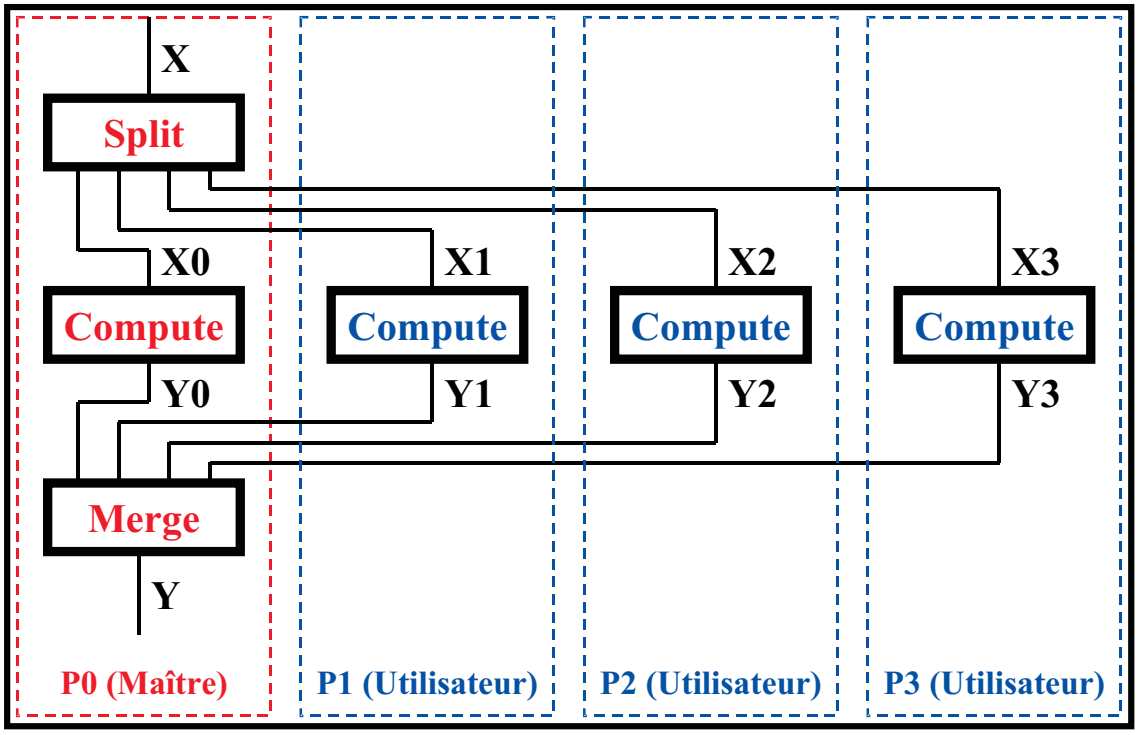


FIGURE B.4 – exemple d'implantation du *squelette SCM* sur n processeurs ($n = 4$).

Ce type de squelette permet donc de diviser des données initiales, d'effectuer un ensemble de calculs identiques sur des données distribuées et de fusionner finalement les résultats intermédiaires. La caractéristique principale de son comportement parallèle est sa nature totalement statique. Du fait de ses propriétés statiques, ce squelette n'est utilisable que pour des algorithmes caractérisés par une uniformité temporelle des calculs sur chaque partition. L'efficacité d'un tel squelette dépend en effet implicitement de l'équilibre de charge des calculs entre les processeurs.

Phase d'implantation

Dans cette phase, prise entièrement en charge par SKiPPER-II, le parallélisme de chaque squelette est *explicité* en utilisant une représentation intermédiaire des programmes adaptée à l'architecture cible afin de créer au final le code exécutable. L'une des caractéristiques de cet outil d'aide à la parallélisation est de pouvoir imbriquer plusieurs squelettes dans la même application. Cette propriété est utilisée pour la parallélisation de l'algorithme de suivi 3D que nous allons maintenant développer dans la prochaine section.

B.3 Parallélisation de l'algorithme de suivi 3D avec imbrication de squelettes

B.3.1 Principe de la parallélisation

A partir de notre algorithme séquentiel de suivi 3D (figure B.2), nous proposons la parallélisation suivante :

1. Les calculs effectués pour les différents motifs de référence (précédent (p), courant (c) et suivant (s)) peuvent être réalisés de manière complètement indépendante les uns des autres, et donc en parallèle. De plus, ces traitements représentent la même charge de travail pour chaque processeur s'ils sont exécutés indépendamment les uns des autres. Nous considérons donc que le *premier niveau de parallélisation* coïncide avec un premier *squelette à parallélisme de données* (squelette noté A sur la figure B.5). Ce squelette est utilisé pour réaliser la comparaison entre les différents résultats de suivi calculés en parallèle et obtenus à partir des trois motifs de référence (c'est à dire comparer les différentes erreurs quadratiques de différence de niveaux de gris pour chaque motif recalé).
2. La multiplication d'une *matrice d'interaction* par un *vecteur de différence de niveaux de gris* permettant de recaler le motif dans l'image courante pour chacun des motifs de référence testés peut aussi être parallélisée. Ce *second niveau de parallélisation* correspond également à un *squelette à parallélisme de données* mais qui est imbriqué dans le précédent (il est représenté par le squelette B sur la figure B.5).

Il est important de noter que ces deux niveaux de parallélisation ne peuvent pas être fusionnés en un seul et même niveau. Cela tient au fait que les trois opérations correspondant chacune à la multiplication d'une *matrice d'interaction* par un *vecteur de différence de niveaux de gris* entre le motif courant échantillonné dans l'ellipse prédite et le motif de référence testé ne peuvent pas être fusionnées en une seule (trois blocs de données différentes).

Quel que soit le niveau de parallélisation étudié, les traitements sont de nature régulière, c'est à dire que leur complexité calculatoire ne dépend pas du contenu des données, mais seulement de leur taille, taille qui est connue lors de la compilation. Cela autorise l'emploi de *squelettes SCM* comme schémas de parallélisation. Par conséquent, la structure de l'application est une *imbrication sur deux niveaux de squelettes SCM*, celui qui est imbriqué jouant le rôle de la fonction de calcul du squelette englobant (fonction *Compute* du *squelette SCM englobant*).

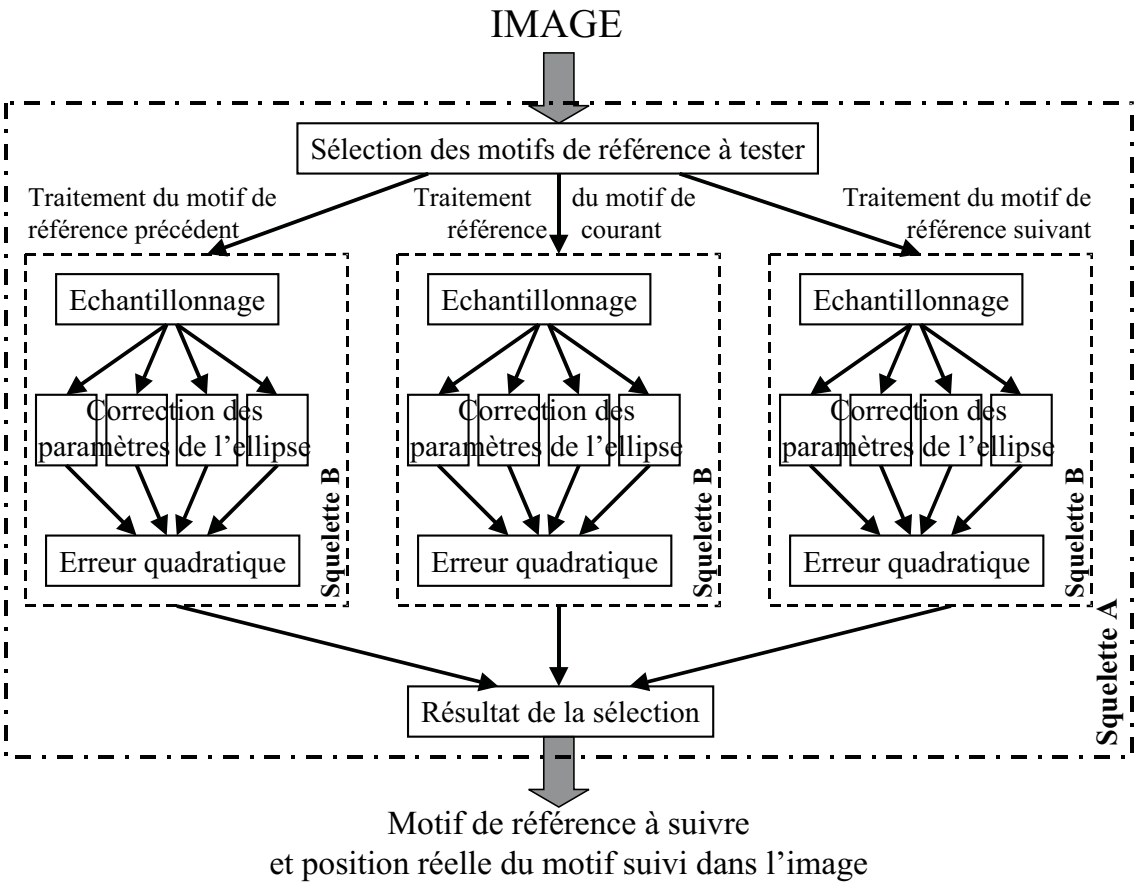


FIGURE B.5 – structure générale de la version parallèle de l'algorithme de suivi.

En résumé, le squelette englobant (squelette A sur la figure B.5) permet de gérer en parallèle le suivi des trois motifs de référence sélectionnés. Il comprend une fois déployé :

1. une fonction de répartition *Split* en entrée,
2. une fonction de calcul *Compute* remplacée par un squelette imbriqué (squelette B sur la figure B.5), dupliquée trois fois pour traiter indépendamment le suivi des différents motifs de référence (calcul des corrections des paramètres de l'ellipse),
3. une fonction de comparaison *Merge* en sortie.

Le squelette imbriqué (squelette B), quant à lui, se déploie de la manière suivante :

1. une fonction de répartition *Split* en entrée,
2. une fonction de calcul *Compute* dupliquée quatre fois pour traiter indépendamment les différents paramètres de l'ellipse à corriger (X_c, Y_c, R_1 et θ),
3. une fonction de fusion *Merge* des résultats intermédiaires en sortie.

Dans cette approche, nous supposons également que l'image à traiter est disponible dans la mémoire locale de chaque unité de calcul (ou processeur). Le cheminement des différentes informations à travers l'*imbrication des squelettes déployés* s'effectue de la manière suivante :

1. la fonction de répartition *Split* du squelette englobant (squelette A) sélectionne dans un premier temps les trois motifs de référence à tester. Elle répartit et transmet ensuite un numéro de motif de référence avec les paramètres associés de l'ellipse prédite à chaque fonction *Split* des trois *squelettes SCM imbriqués* (squelettes B).
2. la fonction *Split* d'un squelette imbriqué (squelette B) échantillonne tout d'abord, dans l'image courante, le motif à l'intérieur d'une zone délimitée par la position prédite de l'ellipse qui lui a été transmise. Elle calcule ensuite les différences de niveaux de gris entre le motif prédit et le motif de référence sélectionné qui seront utilisées pour calculer les corrections sur les paramètres de l'ellipse. Elle transmet enfin ce vecteur de différence, accompagné de la *matrice d'interaction* associée et le numéro de la ligne de la matrice servant au calcul à chacune des quatre fonctions *Compute* du squelette imbriqué.
3. toutes ces fonctions *Compute* sont chargées d'estimer les corrections à apporter aux paramètres de l'ellipse. En fait, chaque fonction de calcul se charge d'établir la correction d'un seul de ces paramètres. Une correction est obtenue en multipliant une ligne de la matrice par le vecteur de différence de niveaux de gris (d'où l'intérêt de transmettre le numéro de ligne de la matrice pour identifier sur quel paramètre il faudra appliquer la correction calculée).
4. les résultats obtenus sont envoyés à la fonction *Merge* du squelette imbriqué correspondant (squelette B) qui va procéder aux calculs de la nouvelle différence de niveaux de gris après la mise à jour des paramètres de l'ellipse et de l'erreur quadratique associée.
5. la fonction *Merge* du squelette englobant (squelette A) peut alors recueillir les trois résultats (un par squelette imbriqué). Ils contiennent les paramètres corrigés de l'ellipse, l'erreur quadratique et le numéro du motif de référence qui est associé au calcul. La fonction peut alors finalement sélectionner le motif de référence parmi les trois initiaux qui donne la plus petite erreur quadratique et donc par la suite la position correcte du motif suivi dans l'image et le numéro du nouveau motif de référence à suivre.

B.3.2 Implantation

Une fois la structure parallèle de l'application identifiée et fixée, SKiPPER-II peut être utilisé pour obtenir l'implantation parallèle finale de l'application sur une machine cible. Connaissant

les squelettes à utiliser et la manière dont ils s'organisent, il ne reste plus au programmeur de l'application qu'à fournir les fonctions séquentielles de calcul pour les intégrer aux squelettes. Comme nous l'avons vu, aucune modification ne doit être apportée aux fonctions de calcul, mais leur interface doit faire l'objet d'une adaptation pour se mettre en conformité avec le format imposé par le noyau K/II.

Nous avons représenté sur la figure B.8 la structure graphique de l'application sous sa forme intermédiaire TF/II (a) et la façon dont le noyau de SKiPPER-II l'encode dans sa représentation interne (b). Dans cette représentation graphique, les notations Si et Mi correspondent aux fonctions *Split* et *Merge* du squelette i ($i = 1$ pour le squelette A et $i = 2$ pour le squelette B), F2 pour la fonction *Compute* du squelette B. La structure de données utilisée pour la représentation interne est un simple tableau écrit en C rapidement exploitable par le noyau. Sur cette figure nous avons respecté les notations suivantes :

- à chaque ligne de la structure est associée un squelette,
- la première colonne est l'identificateur du squelette,
- la seconde colonne indique si les résultats produits par le squelette doivent être communiqués au squelette qui le suit dans le graphe ou au squelette englobant en cas d'imbrication,
- la troisième colonne spécifie le statut du squelette en cas d'imbrication, ce qui permet de savoir si une fonction de calcul pure doit être exécutée par ce squelette ou non,
- la dernière colonne spécifie l'identificateur du squelette à imbriquer dans le cas d'une imbrication.

Puisque SKiPPER-II n'impose aucune restriction sur le prototype des fonctions utilisateur pour garantir plus de souplesse dans l'écriture des applications, seules quelques lignes de *stub-code* viennent s'intercaler entre le noyau et les fonctions utilisateur pour faire la correspondance. La figure B.9 donne l'aspect des signatures des fonctions utilisateur pour cette application. On peut ici constater qu'aucune contrainte ne vient gêner leur écriture et qu'aucun ajout n'a eu besoin d'être fait par rapport à la syntaxe C normalisée.

B.4 Résultats de l'expérimentation

Les mesures des performances pour cette application ont été menées sur une machine Beowulf équipée de 32 nœuds de calcul (ou processeurs) de type Intel Celeron cadensés à

533 MHz. Le Beowulf était équipé d'un réseau Ethernet à 100 Mbits/s. Les deux figures B.6 et B.7 montrent respectivement le temps d'exécution de l'algorithme en fonction du nombre de processeurs mis en jeu, et l'accélération relative obtenue. Par définition, l'accélération *Acc* (ou *speedup*) mesure le gain temporel entre la meilleure exécution séquentielle ts sur un processeur et l'exécution tp sur n processeurs :

$$Acc = \frac{ts}{tp}$$

Deux valeurs de points d'échantillonnage ont été considérées pour les tests, à savoir 170 et 373 points. On pourra cependant noter qu'utiliser plus de 170 points ne donne pas de meilleurs résultats en termes de stabilité et de performance intrinsèque pour l'algorithme de suivi lui-même. Le suivi est déjà suffisamment robuste avec seulement 170 points. Le choix d'augmenter le nombre de points d'échantillonnage dans l'ellipse n'a été décidé que dans le cadre de l'évaluation des performances de SKiPPER-II (question du rapport calculs/communications qu'influence directement ici ce nombre). Les courbes représentées sur les figures B.6 et B.7 montrent trois phases distinctes dans le fonctionnement du noyau de SKiPPER-II.

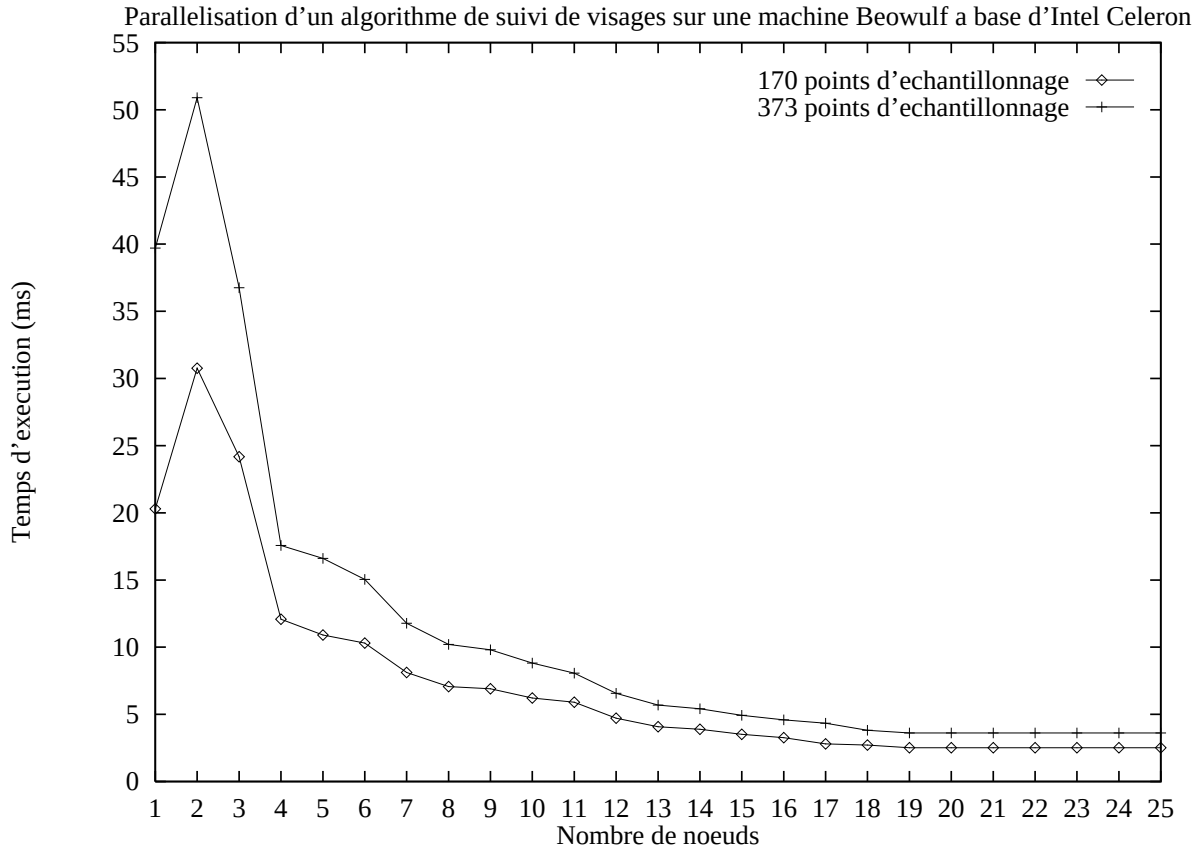


FIGURE B.6 – temps d'exécution en ms pour 170 et 373 points d'échantillonnage.

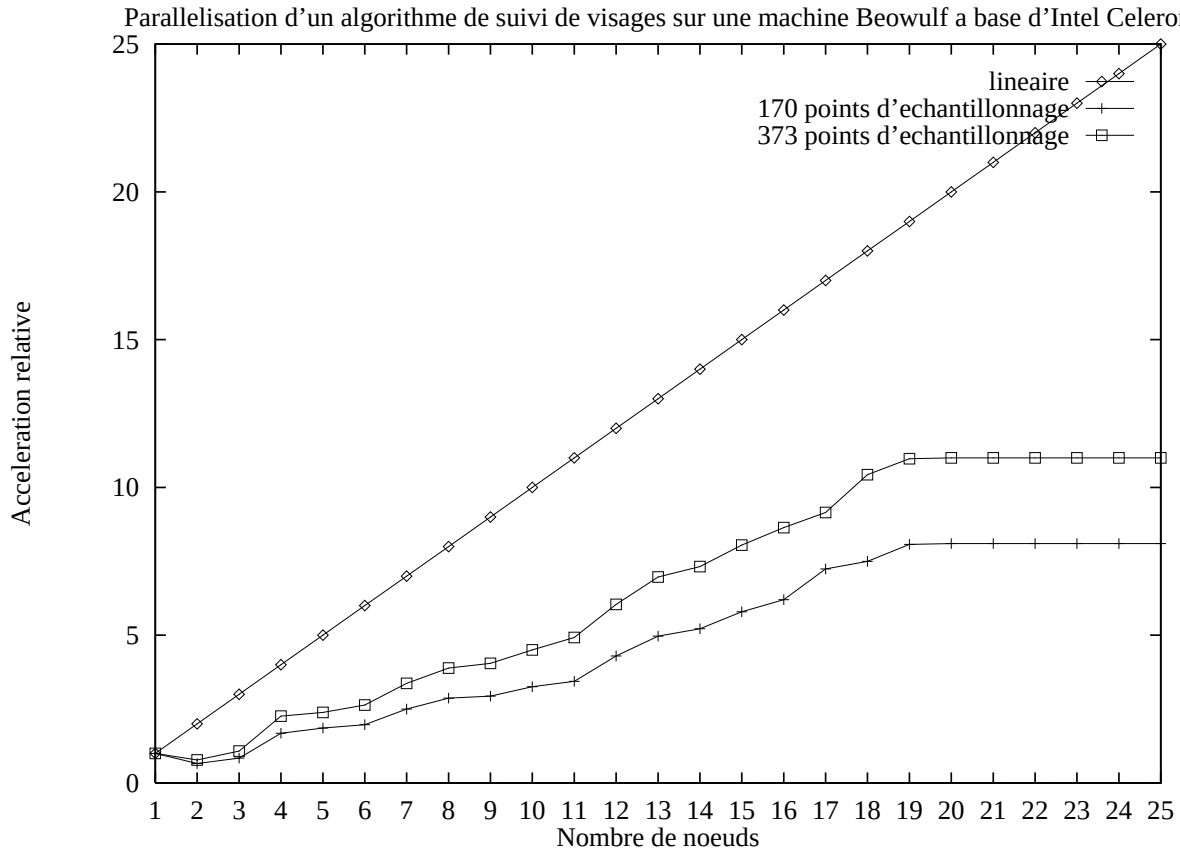


FIGURE B.7 – accélération pour 170 et 373 points d'échantillonnage.

La première de ces phases apparaît lorsque nous utilisons seulement 1 ou 2 processeurs pour 373 points d'échantillonnage et entre 1 et 3 processeurs pour 170 points. Au début de la parallélisation, on constate que le temps d'exécution est plus important lorsque plusieurs processeurs sont mis en œuvre que lorsqu'il n'y en a qu'un seul. Ce phénomène s'explique qu'avec l'utilisation de plus d'un processeur, mais néanmoins extrêmement peu, le *squelette SCM englobant* est déployé sur les unités de calcul. Par conséquent, et conformément au fonctionnement de SKiPPER-II [31], utiliser 2 nœuds de calcul au lieu d'un seul ne fournit pas plus de puissance de calcul, mais par contre accroît très sensiblement le volume des communications. Cela est dû au fait qu'un des deux nœuds de calcul est utilisé comme processus de répartition des traitements à effectuer et non pour réaliser concrétement des traitements. Lorsque le ratio calculs/communications est faible (comme dans le cas où l'on n'utilise que 170 points), le phénomène est très sensible et donc s'observe jusqu'à 3 nœuds. Dans ces cas, le temps passé à communiquer les données entre les processeurs annihile le gain en temps d'exécution obtenu par la mise en parallèle de ces processeurs.

De 4 à 19 nœuds, les performances augmentent avec le nombre de processeurs. Cette phase

correspond au déploiement du squelette imbriqué qui réalise la multiplication de la matrice par le vecteur de différence de niveaux de gris en parallèle.

La dernière phase commence, quant à elle, à partir de 19 nœuds. A ce moment là, l'augmentation du nombre de processeurs ne fournit plus de puissance de calcul supplémentaire. Cela s'explique par le fait que les *squelettes SCM* encapsulent une stratégie de parallélisme de données à gain fixe, c'est à dire que pour se développer complètement, un *squelette SCM* a besoin d'un nombre déterminé de processeurs fonction du travail qu'il a à effectuer mais qui ne peut être modifié pendant l'exécution. Le maximum d'efficacité est donc atteint pour un nombre de processeurs équivalent au nombre d'esclaves du *squelette SCM.*[1] Passé ce nombre (ici de 19 à 32), il n'y a plus de parallélisme à exploiter, et par conséquent l'efficacité décroît d'autant. Cela implique aussi qu'en dessous du nombre de processeurs critique nécessaire à un *squelette SCM*, le noyau de SKiPPER-II ne peut déployer le squelette dans son intégralité. Comme nous l'avons déjà vu, il répartit séquentiellement les calculs sur un même nœud esclave ou utilisateur.

Les résultats relativement faibles en terme d'efficacité qui sont relevés sur ces figures reflètent en fait un ratio calculs/communications lui aussi relativement faible dans la version parallèle. Il faut mentionner ici que la version séquentielle de l'application de suivi était déjà très efficace puisqu'elle permet de suivre un motif en temps réel vidéo ($<$ 15ms). La cause principale en est la faible représentation de calculs intensifs dans l'algorithme. C'est tout particulièrement vrai pour la multiplication matricielle faite ici au sein d'un *squelette SCM* pour la version parallèle : le calcul porte seulement sur la multiplication d'une matrice $(p \times N)$ par un vecteur de dimension N. Nous rappelons ici que $p = 4$ (nombre de paramètres de l'ellipse ou de la transformation géométrique modélisant le mouvement du motif dans l'image) et $N = 170$ ou 373 (nombre de points échantillonnés dans la zone d'intérêt).

B.5 Discussion et conclusion

La parallélisation d'une application de vision artificielle réelle nécessitant la mise en œuvre de l'imbrication dans une méthodologie utilisant les *squelettes algorithmiques ou fonctionnels* a rarement été proposée à notre connaissance. Le suivi de visage que nous venons de décrire

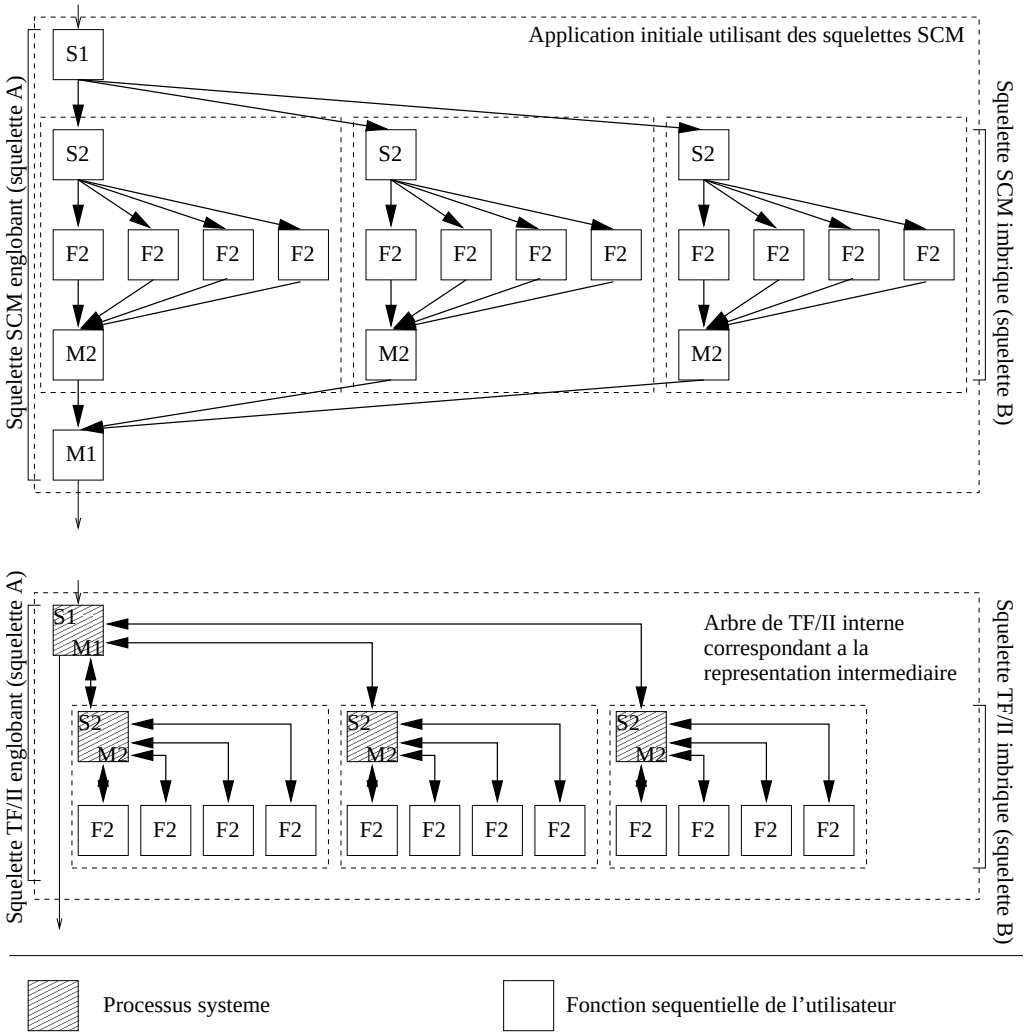
1. Dans le cas présent il faut tenir compte du fait que les *squelettes SCM* au moment de leur exécution sont sous la forme de TF/II. Par conséquent le nombre optimal de processeurs à utiliser pour le squelette englobant est de 4 (3 pour les calculs et 1 pour les répartir), et de 5 pour le squelette imbriqué (4+1).

a été parallélisé en ce sens, pour montrer l'intérêt de disposer de la possibilité d'imbriquer les squelettes des bibliothèques des environnements de programmation parallèle fondés sur ce concept, afin d'élargir leur champ d'application. Sans cette possibilité, l'algorithme de suivi de visage aurait dû être modifié pour se prêter à une parallélisation complète. En effet, les niveaux de parallélisation que nous avons isolés ne peuvent pas être en l'état fusionnés.

Au-delà, cette expérience rappelle que l'*imbrication de squelettes* n'est pas une opération transparente en terme d'efficacité. L'utiliser par nécessité ou par commodité pour paralléliser un algorithme ne doit pas faire oublier les mécanismes et les transferts supplémentaires dédiés à cette situation particulière d'exécution des squelettes qu'elle impose.

Concernant les aspects méthodologiques de cette implantation, il faut retenir que la parallélisation complète de cet algorithme (non prévue à l'origine pour cela) n'a demandé que quelques jours de travail. Ce temps a d'ailleurs été consacré pour l'essentiel à la mise en évidence des schémas de parallélisation adéquates (sélection des squelettes et organisation de leurs interconnections). Une fois cet important travail préparatoire terminé, la deuxième étape est d'effectuer le découpage de l'algorithme en fonctions autonomes qui prendront place comme fonctions de calcul dans les squelettes choisis à l'étape précédente (repartition des fonctions déjà existantes et adaptation de leurs interfaces).

Concernant les choix de parallélisation pour l'algorithme de suivi que nous venons de présenter, nous pensons à la lumière du travail qui a été mené que, pour des aspects d'efficacité pure, il serait plus judicieux de paralléliser la première phase de l'algorithme, à savoir la phase d'apprentissage hors ligne des différentes *matrices d'interaction*. La raison principale est de diminuer son temps d'exécution.



(a) représentation graphique.

```
#define SKL_NBR 2

SK2_Desc app_desc[SKL_NBR] =

{

   {SK0, END_OF_APP, MASTER, SK1 },

   {SK1, UPPER     , SLAVE , NIL }

};
```

(b) représentation intermédiaire.

FIGURE B.8 – représentation interne de l'application de suivi d'un visage.

```
S1(
  int         pattern_number,               /* Entree        */
  Ellipse     current_ellipse,              /* Entree/Sortie */
  int      ** tracker_number_to_test        /*       /Sortie */
); S2(
  Ellipse     current_ellipse,              /* Entree/Sortie */
  int         tracker_number_to_test,       /* Entree/Sortie */
  int      *  gray_level_vector_size,       /*       /Sortie */
  float    ** gray_level_difference_vector, /*       /Sortie */
  int      ** matrix_line_number,           /*       /Sortie */
  float   *** matrix                        /*       /Sortie */
); F2(
  Ellipse     current_ellipse,              /* Entree/Sortie */
  int         tracker_number_to_test,       /* Entree/Sortie */
  int         gray_level_vector_size,       /* Entree        */
  float    *  gray_level_difference_vector, /* Entree        */
  int         matrix_line_number,           /* Entree/Sortie */
  float    *  matrix,                       /* Entree        */
  float    *  correction                    /*       /Sortie */
); M2(
  Ellipse     current_ellipse,              /* Entree        */
  int         tracker_number_to_test,       /* Entree/Sortie */
  int         matrix_line_number,           /* Entree        */
  float       correction                    /* Entree        */
  float    *  quadratic_error,              /*       /Sortie */
  Ellipse  *  corrected_ellipse,            /*       /Sortie */
); M1(
  Ellipse     corrected_ellipse,            /* Entree        */
  int         tracker_number_to_test,       /* Entree        */
  float       quadratic_error,              /* Entree        */
  Ellipse  *  final_current_ellipse,        /*       /Sortie */
  int      *  final_pattern_number          /*       /Sortie */
);
```

FIGURE B.9 – prototypes des fonctions utilisateur pour l'application de suivi d'un visage.

Annexe C

Notions de robotique

Dans cette annexe, nous présentons des généralités sur la modélisation d'un robot et donnons des compléments d'information sur les différents asservissements possibles avant de décrire les spécificités de la plate-forme robotique utilisée au laboratoire.

C.1 Introduction

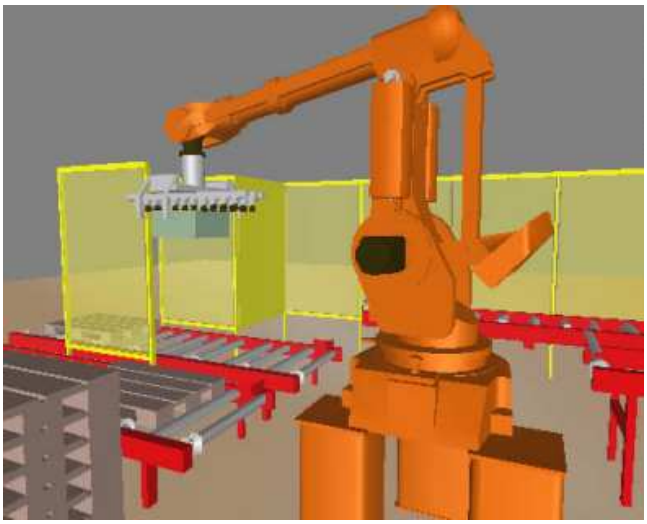
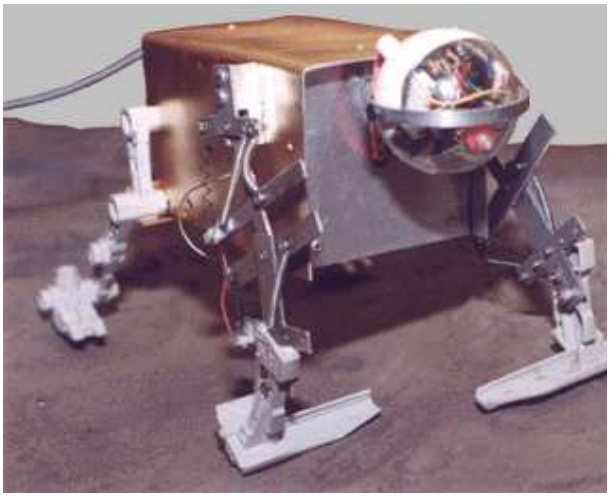


FIGURE C.1 – robot manipulateur dans un atelier de stockage.

Il y a encore quelques années, les robots industriels ne réalisaient que des tâches robotiques dont les trajectoires étaient parfaitement prédéfinies ou connues à l'avance. Ainsi, la variété des applications industrielles était très limitée car ces applications ne prenaient pas en compte les informations relatives à l'environnement extérieur à la tâche (figure C.1).



(a) prototype BiRoD (Biomorphic Robot with Distributed power) de l'Université d'Arizona.

(b) exploration du sol de Mars par le robot surnommé "*Rocky*".

FIGURE C.2 – robots dédiés à l'exploration du système solaire.

Cette limitation a poussé les chercheurs en robotique à mener des travaux afin de doter les robots de capteurs externes capables de donner des informations sur la situation du robot par rapport à son environnement. Ces développements ont alors permis de concevoir de nombreuses autres tâches robotiques comme par exemple la détection et l'évitement d'obstacles (figure C.2).

L'utilisation de ces capteurs a tout de même posé de nombreux problèmes théoriques concernant l'intégration des différentes données externes dans la commande des robots. Une partie importante de ces travaux porte le nom de *Commande Référencée Capteur* (CRC).

Très vite, le capteur de vision (la caméra) s'est imposé pour résoudre les problèmes complexes de perception de l'environnement. Ce capteur est particulièrement intéressant par la richesse des informations qu'il fournit. Sa miniaturisation et les récents développements du traitement d'images ont permis, dans un premier temps, de positionner le capteur visuel di-

rectement sur l'effecteur du robot et, dans un deuxième temps, d'intégrer les données visuelles dans la boucle de commande du robot.

Nous distinguerons par la suite deux grandes catégories de lois de commande s'appuyant sur des primitives visuelles pour établir un signal d'erreur à corriger dans une boucle d'asservissement. La première utilise des primitives directement extraites de l'image (coordonnées d'un point dans l'image par exemple) et la seconde utilise des primitives de l'espace cartésien (3D), reconstruites à partir des informations visuelles (2D). Mais tout d'abord, rappelons quelques définitions.

C.2 Rappels sur l'attitude d'un repère par rapport à un autre

C.2.1 Les degrés de liberté d'un solide

Soit un solide indéformable représenté par son repère R. La mécanique définit les capacités de positionnement et d'orientation (notées *attitude*) de ce solide selon *six degrés de liberté* (*d.d.l.*) par rapport à l'espace réel à trois dimensions de son environnement. Ces six d.d.l. (trois translations et trois rotations) lient l'attitude du solide dans l'espace environnant.

C.2.2 Les degrés de liberté d'un robot

Les degrés de liberté d'un robot sont caractérisés par le type de liaisons mécaniques reliant l'ensemble des éléments du robot. La liaison (ou articulation) entre deux éléments du robot peut être : *prismatique* (une translation), *rotoïde* (une rotation), *pivot glissant* (une translation et une rotation) ...

Par conséquent, son nombre de d.d.l. dépend du nombre et du type d'articulations qu'il comprend. Dans notre cas, nous utilisons un robot cartésien à six degrés de liberté. Il se décompose en trois axes de translation orthogonaux et trois rotations.

C.2.3 Attitude d'un repère par rapport à un autre

On définit une *attitude* (aussi notée *situation* ou *pose* en anglais) d'un repère par rapport à un autre comme étant la position relative de l'un vers l'autre. Pour le robot, un repère est lié

à chacune des articulations qu'il comprend.

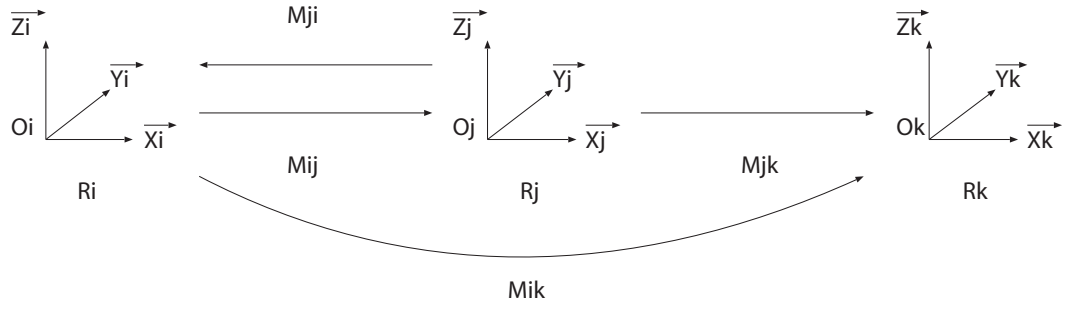


FIGURE C.3 – changements de repère.

Soient R_i et R_j deux repères orthonormés $(O_i, \vec{X}_i, \vec{Y}_i, \vec{Z}_i)$ et $(O_j, \vec{X}_j, \vec{Y}_j, \vec{Z}_j)$. On peut représenter l'attitude de R_j par rapport à R_i de deux façons :

1. soit par un *vecteur de six variables* (les six degrés de liberté) qui sont nécessaires et suffisantes à cette représentation : trois en translation T (ou position) et trois en rotation R (ou orientation). Les positions sont en général des *coordonnées cartésiennes* et les rotations des *angles d'Euler*.

$$\underline{v} = (T_X, T_Y, T_Z, R_X, R_Y, R_Z)$$

2. soit par une *matrice homogène de transformation*, notée M_{ij} (figure C.3) : c'est en général cette représentation qui est utilisée dans les applications. Elle est définie de la manière suivante :

$$M_{ij} = \begin{bmatrix} R_{ij} & T_{ij} \\ 0 & 1 \end{bmatrix}$$

où

- R_{ij} est la matrice de rotation (3×3) entre $(\vec{X}_i, \vec{Y}_i, \vec{Z}_i)$ et $(\vec{X}_j, \vec{Y}_j, \vec{Z}_j)$,
- T_{ij} est le vecteur de translation (3×1) correspondant aux coordonnées de O_j dans R_i.

Inversement, la *matrice de transformation homogène* M_{ji} représentant R_i par rapport à R_j est définie par :

$$M_{ji} = \begin{bmatrix} R_{ji} & T_{ji} \\ 0 & 1 \end{bmatrix} = M_{ij}^{-1} = \begin{bmatrix} R_{ij}^T & -R_{ij}^T.T_{ij} \\ 0 & 1 \end{bmatrix}$$

Le passage à un troisième repère R_k (figure C.3) est représenté par la relation :

$$M_{ij}.M_{jk} = M_{ik}$$

C.3 Modélisation d'un robot

Physiquement, un robot à n d.d.l. est constitué de $n+1$ éléments (ou parties) reliés entre eux par n axes (ou articulations). Ces axes permettent (dans la plupart des cas) soit une liaison en translation (*prismatique*), soit une liaison en rotation (*rotoïde*).

Le dernier élément porte le nom d'*effecteur*. Son repère est bien sûr lié à la dernière articulation.

Le but de la modélisation d'un robot est de pouvoir commander par un processus informatique l'attitude de cet effecteur afin de l'asservir sur une attitude (ou situation) souhaitée.

C.3.1 Les différents modèles composant un robot

Pour générer un mouvement désiré, il faut tenir compte des grandeurs successives suivantes (en notant que les n variables de chaque vecteur sont numérotées par convention de 0 à $n-1$) :

1. $\underline{V(t)} = [V_0(t) \dots V_{n-1}(t)]^T$: le vecteur des tensions de commande appliqué à chaque moteur. Il appartient à l'*espace des commandes* du plus bas niveau.
2. $\underline{\Gamma(t)} = [\Gamma_{0(t)} \dots \Gamma_{n-1}(t)]^T$: le vecteur des couples moteurs appliqué sur chaque arbre moteur avant les transmissions. Il appartient à l'*espace des couples moteurs*.
3. $\underline{C(t)} = [C_0(t) \dots C_{n-1}(t)]^T$: le vecteur des couples articulaires appliqué sur chaque axe articulaire. Il appartient à l'*espace des couples articulaires*.
4. $\underline{q(t)} = [q_0(t) \dots q_{n-1}(t)]^T$: le vecteur des *variables articulaires* ou *généralisées* correspondant aux valeurs des angles de rotation ou des longueurs de translation prises par chaque axe (valeurs fournies par les capteurs proprioceptifs). Il appartient à l'*espace articulaire*.
5. $\underline{F(t)} = [F_0(t) \dots F_{m-1}(t)]^T$: le vecteur des *variables opérationnelles* correspondant à l'attitude de l'effecteur par rapport au repère de référence du robot. Ce repère de référence noté R_r est aussi appelé repère fixe du robot ou encore repère cartésien. Il représente l'*espace opérationnel* ou l'*espace de travail du robot*. C'est dans ce dernier que s'effectueront les tâches robotiques souhaitées. Le vecteur des variables opérationnelles n'a pas un indice associé au nombre n d'articulations, mais au nombre m de paramètres déterminant

l'attitude de l'effecteur par rapport au repère fixe du robot. Ce nombre est défini par la méthode utilisée pour le calcul du modèle du robot.

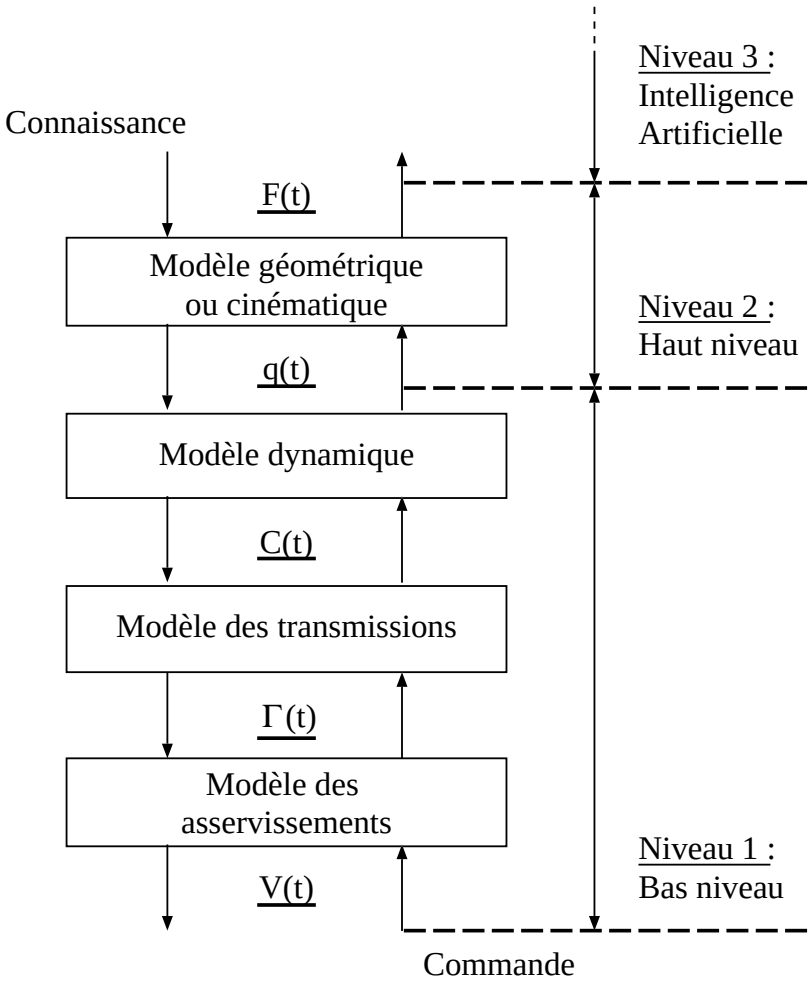


FIGURE C.4 – modèle complet d'un robot.

D'après la figure C.4 illustrant la modélisation complète d'un robot, nous constatons que chaque modèle peut être utilisé dans les deux sens :

- sens *direct* : correspondant au *modèle direct* appelé aussi *modèle de connaissance* pour le calcul de la sortie, en connaissant l'entrée.
- sens *inverse* : correspondant au *modèle inverse* appelé aussi *modèle de commande* pour le calcul de l'entrée, en connaissant la sortie.

Ensuite, intervient le choix du niveau de commande que l'on souhaite mettre en œuvre. Nous souhaitons nous affranchir au maximum des problèmes matériels (induits par des commandes

de niveau 1) en choisissant les espaces de commande de niveau 2 (le niveau 3 étant réservé à l'intelligence artificielle). Nous utiliserons donc uniquement le *modèle géométrique*. Mais ceci implique deux hypothèses. La première, dite *statique*, suppose que la commande de niveau 1 soit parfaite, c'est à dire qu'elle ne tienne pas compte des aspects dynamiques du robot. La seconde suppose que tous les éléments (segment du robot entre deux axes) soient parfaitement *rigides* et que les articulations soient *mécaniquement parfaites* (pas de frottement, pas d'élasticité...).

C.3.2 Le modèle géométrique

Le robot cartésien à six d.d.l. est schématisé par la figure C.5. Sur ce dessin, nous avons matérialisé les articulations *prismatiques* à l'aide de parallélépipèdes et les articulations *rotoïdes* à l'aide de cylindres. En attachant un repère R_i à chaque articulation i du robot (pour i variant de 0 jusqu'à 5), nous pouvons alors définir le *modèle géométrique* qui établit les fonctions de correspondance entre l'attitude de l'effecteur (*les variables opérationnelles* $\underline{F}$) dans le repère de référence du robot (grâce aux matrices de transformation homogènes entre les repères) et la position de tous les axes (*les variables articulaires* $\underline{q}$). Si nous nous intéressons aux relations entre les vitesses $\underline{\dot{F}}$ et $\underline{\dot{q}}$, nous parlons dans ce cas de *modèle cinématique* ou *Jacobien du robot*.

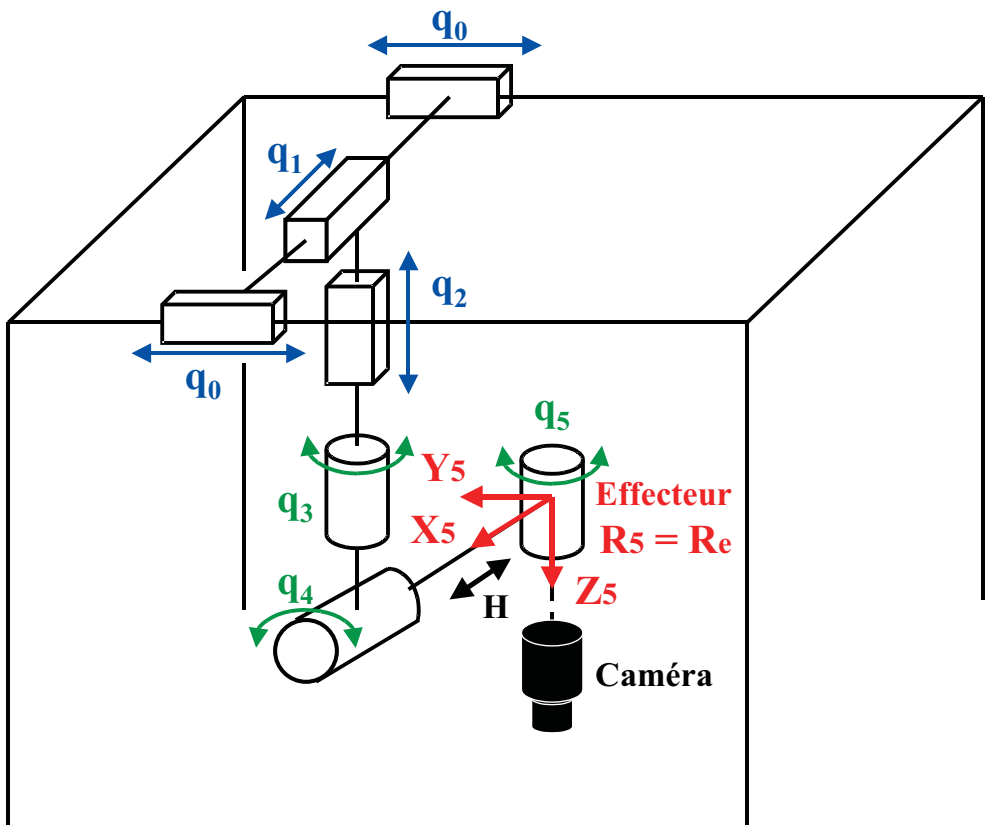


FIGURE C.5 – schéma du robot cartésien à six d.d.l..

Dans cette représentation (figures C.5 et C.6), le repère de référence R_r est considéré comme *fixe* (par rapport à la cage du robot) et n'est attaché à aucune articulation. Les autres repères (R_0 à R_5) sont *mobiles*. Le centre du volume formé par la cage du robot a été choisi comme origine du repère R_r. Son orientation et sa position sont définies par rapport à une position initiale (butées mécaniques de chaque axe qui délimitent l'espace de travail du robot).

Il est donc possible de déplacer l'effecteur du robot soit en *coordonnées absolues* (par rapport à la position de référence zéro du robot pour $q_i = 0$) soit en *coordonnées relatives* (par rapport à la position courante du robot).

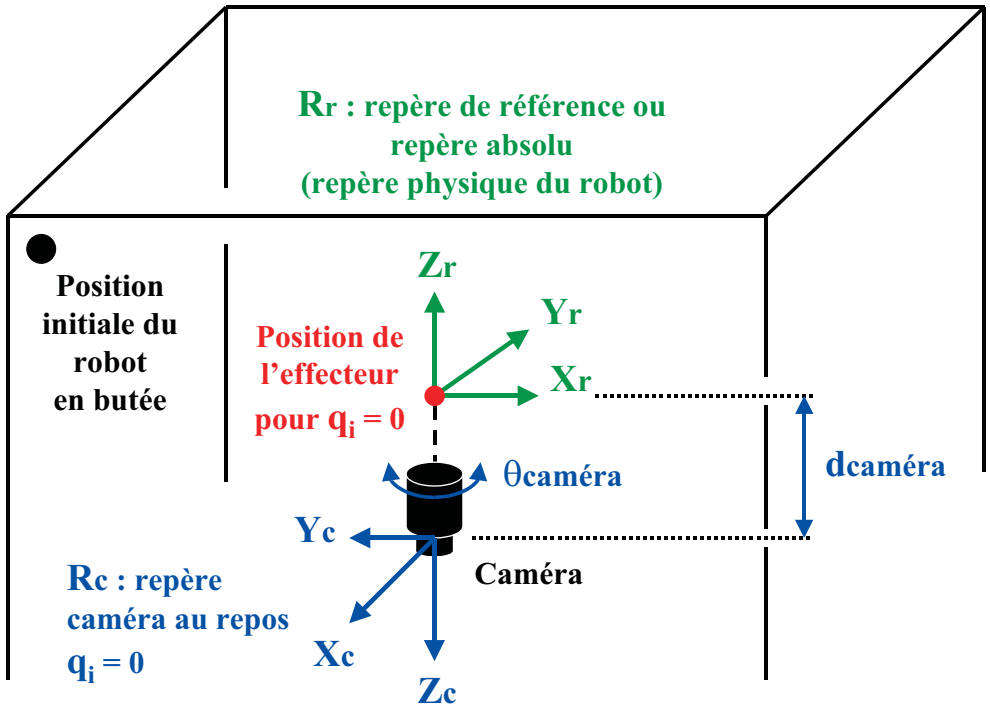


FIGURE C.6 – référence robot et repère caméra.

Le modèle géométrique direct

Le *modèle géométrique direct du robot* (noté MGD) est aussi appelé *modèle de connaissance*. Il permet de calculer l'attitude de l'effecteur $\underline{F}$ en connaissant une position quelconque des axes $\underline{q}$.

$$\underline{F} = MGD(\underline{q})$$

La méthode choisie pour calculer ce modèle utilise la *convention de Denavit-Hartenberg* avec des matrices (4×4). Chaque repère R_i d'une articulation i (avec $i = 0$ à $n-1$) doit respecter

certaines contraintes par rapport au repère précédent R_{i-1}.

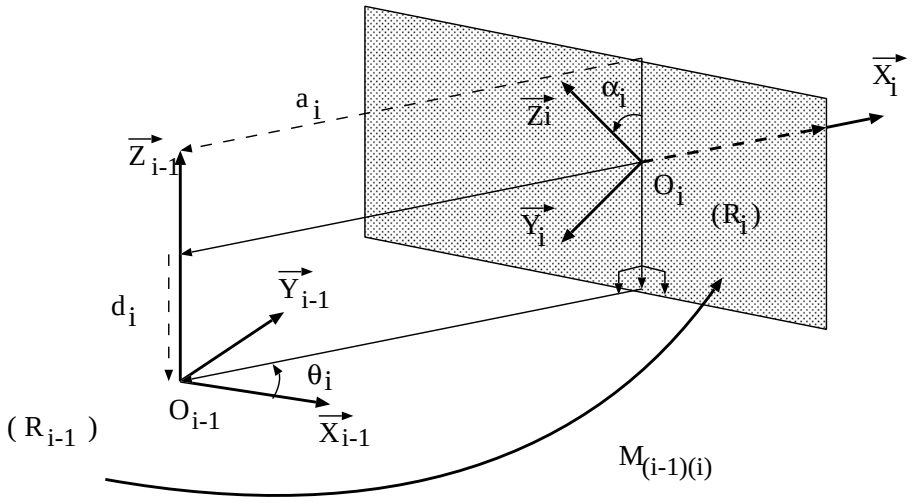


FIGURE C.7 – convention de Denavit-Hartenberg.

La transformation entre R_{i-1} et R_i se décompose en quatre opérations élémentaires ordonnées de la manière suivante :

1. rotation de X_{i-1} vers X_i d'un angle θ_i autour de Z_{i-1},
2. translation de X_{i-1} vers X_i d'une distance d_i selon Z_{i-1},
3. rotation de Z_{i-1} vers Z_i d'un angle α_i autour de X_i,
4. translation de Z_{i-1} vers Z_i d'une distance a_i selon X_i.

où θ_i, d_i, α_i et a_i sont les paramètres de Denavit-Hartenberg (figure C.7). Les paramètres α_i, a_i sont constants et dépendent de la géométrie du robot. θ_i est variable si l'axe de l'élément i du robot constitue une liaison *rotoïde* (d_i restant constant). d_i est variable si l'axe de l'élément i du robot constitue une liaison *prismatique* (θ_i restant constant). Notons que pour cette représentation, c'est l'axe Z_i de R_i qui porte l'axe de l'élément i du robot.

Nous obtenons alors la matrice de transformation homogène $M_{(i-1)(i)}$ entre les repères R_{i-1} et R_i :

$$M_{(i-1)(i)} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}_{R_{i-1}}$$

Le modèle géométrique direct du robot cartésien est calculé à partir de l'équation suivante :

$$MGD = M_{re} = M_{r5} = M_{r0}.M_{01}.M_{12}.M_{23}.M_{34}.M_{45}$$

Ainsi :

- en posant les *variables articulaires* $q_0 = d_0$, $q_1 = d_1$, $q_2 = d_2$, $q_3 = \theta_3$, $q_4 = \theta_4$ et $q_5 = \theta_5$,
- en notant les abréviations $\sin(q_i) = S_i$ et $\cos(q_i) = C_i$,
- en écrivant $H = a_5$ comme étant la longueur entre R_4 et R_5 et en supposant les autres paramètres nuls,

le modèle géométrique direct du robot cartésien est donné par la matrice :

$$MGD = \begin{bmatrix} -S_3.C_4.C_5 - C_3.S_5 & S_3.C_4.S_5 - C_3.C_5 & -S_3.S_4 & d_0 + H.C_3 \\ -C_3.C_4.C_5 + S_3.S_5 & C_3.C_4.S_5 + S_3.C_5 & -C_3.S_4 & d_1 - H.S_3 \\ S_4.C_5 & -S_4.S_5 & -C_4 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{R_r}$$

Rappelons enfin que le capteur de vision (la caméra) se trouve à l'extrémité de l'articulation q_5 et que le passage bras/caméra devra être prise en compte lors de la détermination de la loi de commande.

Le modèle géométrique inverse

Le *modèle géométrique inverse du robot* (noté $MGINV$) est aussi appelé *modèle de commande*. Ce modèle permet de calculer la position $\underline{q}$ des axes, si elle existe, à partir d'une attitude souhaitée $\underline{F}$.

$$\underline{q} = MGINV(\underline{F})$$

Une attitude $\underline{F}$ fait partie de l'ensemble des attitudes que peut atteindre le robot si, à partir d'une attitude souhaitée et en connaissant le modèle géométrique direct du robot considéré, une solution unique à l'équation précédente peut être trouvée mathématiquement. Mais il existe trois problèmes liés à l'inversion du modèle géométrique :

1. problème de géométrie “élémentaire” : si la position souhaitée est hors de l'espace de travail.
2. problème de mécanique “élémentaire” : de part sa construction, certaines articulations ne peuvent effectuer une rotation de 360°. Donc, il existe certains volumes de l'espace de travail que le robot ne peut pas atteindre.
3. problème mathématique : si la matrice homogène MGD n'est pas inversible.

Remarque : de la même façon, nous pouvons définir le *modèle Jacobien* et le *modèle Jacobien inverse* du robot qui lient les vitesses dans les espaces opérationnel et articulaire.

$$\underline{\dot{F}} = \mathcal{J}(q)\underline{\dot{q}} \quad \text{et} \quad \underline{\dot{q}} = \mathcal{J}^{-1}(q)\underline{\dot{F}}$$

C.4 Asservissement visuel

C.4.1 Capteurs proprioceptifs et extéroceptifs

L'objectif de l'asservissement visuel en robotique est de rendre le plus autonome possible le robot dans son environnement. Cet objectif nécessite l'utilisation de capteurs. On peut les classer en deux familles :

1. *les capteurs proprioceptifs* : ce type de capteur nous fournit des informations sur l'état du robot indépendamment de son environnement. Il nous donne donc des informations sur ”l'état interne” du robot. Ainsi une dynamo tachymétrique mesurant une vitesse ou une accélération fait partie de ce groupe de capteurs. Les variables représentant l'état du robot sont les *variables articulaires* $\underline{q}$. Elles représentent la position de chaque axe du robot par rapport à une référence. L'utilisation exclusive de ces capteurs montre rapidement des limites quant il s'agit de prendre en compte un environnement difficilement modélisable. Dans ce cas de figure, il semble indispensable d'avoir des informations sur la situation du robot dans son environnement, c'est l'utilisation de capteurs extéroceptifs qui va nous les donner.
2. *les capteurs extéroceptifs* : ce type de capteur va nous fournir des informations sur l'état du robot dans son environnement (espace cartésien). Six *variables opérationnelles* $\underline{F}$ décrivent totalement la position et l'orientation de l'effecteur dans un repère absolu. La caméra fait partie de ce groupe de capteurs et est capable de fournir des informations très riches.

C.4.2 Informations visuelles et matrice d'interaction

Les déplacements de la caméra, embarquée sur l'effecteur du robot, sont réalisés à l'aide des différents axes constituant le robot. La situation de la caméra ne dépend donc que de la valeur des coordonnées $\underline{F}(t)$ dans l'espace opérationnel du robot.

Les informations visuelles choisies $\underline{S}$ peuvent donc s'écrire : $\underline{S} = \underline{S}(\underline{F}(t), t)$ où la variable temporelle t permet de prendre en compte le mouvement des objets dans la scène. Ainsi on peut écrire :

$$\dot{\underline{S}} = \frac{d\underline{S}}{dt} = \frac{\partial \underline{S}}{\partial \underline{F}} \cdot \frac{d\underline{F}}{dt} + \frac{\partial \underline{S}}{\partial t} \tag{C.1}$$

Pour une cible immobile on obtient :

$$\dot{\underline{S}} = \frac{\partial \underline{S}}{\partial \underline{F}} \cdot \frac{d\underline{F}}{dt} = L_{\underline{S}} \cdot \dot{\underline{F}} = L_{\underline{S}} \cdot \underline{T} \tag{C.2}$$

où :

- $\frac{\partial \underline{S}}{\partial \underline{F}}$ est la *matrice d'interaction*, appelée aussi *Jacobien de tâche* ou *Jacobien d'image*. Elle est notée $L_{\underline{S}}$. Cette matrice traduit les variations des informations visuelles en fonction des différents déplacements de la caméra. Elle caractérise complètement les interactions entre le capteur et son environnement. Cependant, la détermination de cette matrice n'est pas toujours aisée et dépend du type de primitives utilisées. Si le nombre d'informations visuelles est supérieur au nombre de d.d.l du robot alors le Jacobien d'image constitue une *liaison virtuelle rigide* entre la caméra et son environnement. C'est à dire que n'importe quel mouvement du robot modifie au moins une primitive visuelle.
- $\underline{T}$ est le *torseur cinématique*. Il correspond à la vitesse de la caméra dans un repère absolu.

$$\underline{T} = \begin{pmatrix} \vec{V} \\ \vec{\Omega} \end{pmatrix} \tag{C.3}$$

  - $\vec{V} = (V_X, V_Y, V_Z)^T$ représente la vitesse de translation du centre du repère caméra dans un repère absolu.
  - $\vec{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)^T$ est le vecteur vitesse de rotation du repère caméra dans un repère absolu.

C.4.3 Notion de fonction de tâche

Dans un asservissement visuel, les informations extraites des images de la caméra embarquée sont utilisées pour faire exécuter au robot une tâche souhaitée.

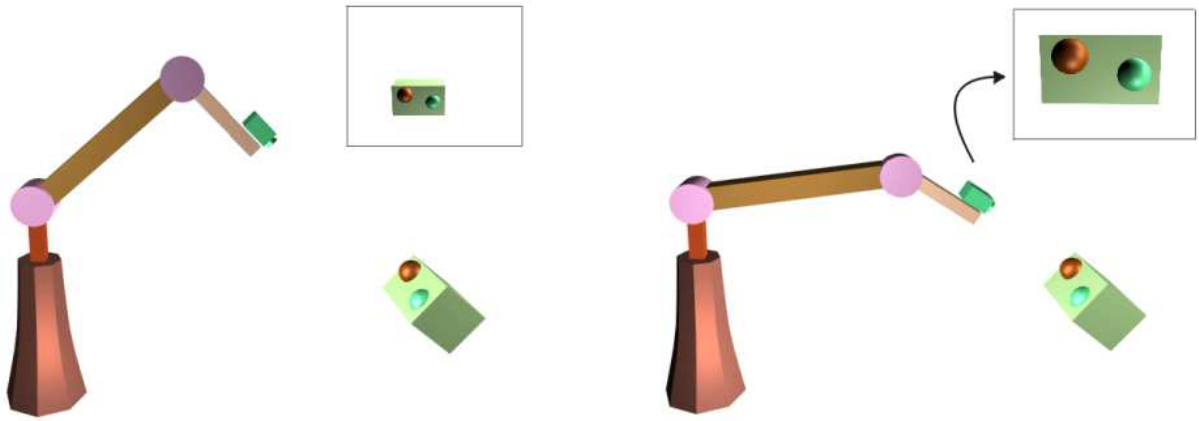
On distingue trois groupes principaux pour les *tâches robotiques* liées à l'utilisation d'informations visuelles :

1. *tâche de suivi* de cible mobile (figure C.8),
2. *tâche de positionnement* par rapport à une cible fixe (figure C.9),
3. *tâche de navigation* entre plusieurs cibles (figure C.10).

Pour ce qui nous concerne, la tâche finale que nous avons à réaliser fait partie du premier groupe. Le but, commun aux trois tâches robotiques de base, est de placer/déplacer la caméra du robot dans/selon une situation/trajectoire souhaitée par rapport aux objets qui l'entourent.

Chaque type de tâche robotique souhaité peut se traduire en termes mathématiques sous la forme d'une *fonction de tâche*. Par définition, une fonction de tâche permet de déterminer une loi de commande directement exploitable par la partie commande du robot. La fonction de tâche peut être définie, selon [18], dans n'importe quel espace de travail : cartésien, articulaire, plan image caméra...

FIGURE C.8 – tâche de suivi de cible mobile.



(a) pose initiale. (b) pose finale.

FIGURE C.9 – tâche de positionnement par rapport à une cible fixe.

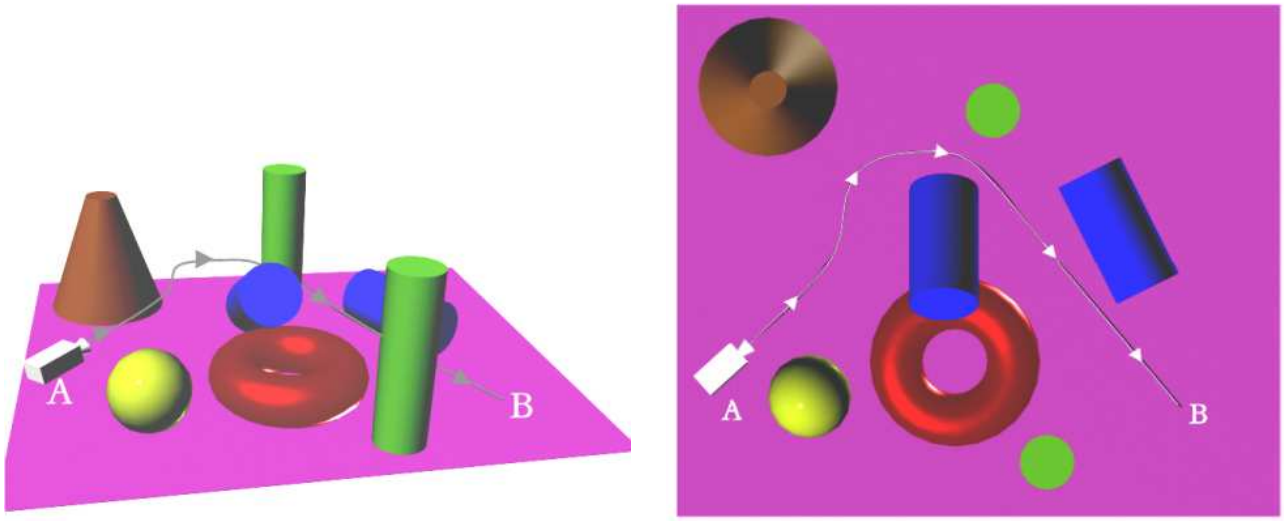


FIGURE C.10 – tâche de navigation entre plusieurs cibles.

La régulation de la fonction de tâche se fera correctement si la loi de commande possède certaines propriétés nécessaires à l'admissibilité de la tâche à réaliser [9] [18]. Ces propriétés permettent de borner le domaine de convergence de la fonction de tâche et donc de déterminer la robustesse de l'asservissement.

Il existe deux techniques d'asservissement utilisant d'une part, des informations visuelles (dont le traitement est directement intégré dans la boucle classique de commande) et d'autre part, la notion de fonction de tâche. Elles se différencient par l'espace de travail utilisé pour les calculs. Autrement dit, cette différence se fait sur les types de consigne/mesure sur lesquels sont réalisés les calculs de la fonction de tâche. Ces deux techniques nommées *asservissement en situation* et *asservissement dans l'image* seront détaillées par la suite. Mais auparavant,

présentons succinctement le formalisme de la Commande Référencée Capteur.

C.4.4 Formalisme de la Commande Référencée Capteur

Au début des années 90, C. Samson, B. Espiau et M. Le Borgne [98] ont proposé une approche appelée *Commande Référencée Capteur* (CRC), dont les premiers travaux ont été menés par B. Espiau lors de développements sur la commande proximétrique.

Dans ce formalisme, toute *tâche robotique* peut se mettre sous la forme d'une *régulation à zéro d'une fonction de tâche* $\underline{E}$ qui exprime *l'erreur entre la configuration souhaitée et la configuration courante* :

$$\underline{E}(\underline{F}, t) = C \cdot (\underline{S}(\underline{F}, t) - \underline{S}^*) \tag{C.4}$$

où :

- C est une *matrice de combinaison*. Elle permet de prendre en compte un nombre d'informations visuelles supérieur au nombre de degrés de liberté contraint par la liaison souhaitée entre le robot et la caméra.
- $\underline{S}$ est un vecteur composé des primitives choisies pour décrire la situation courante relative entre le capteur et l'objet. Celui-ci sera composé de primitives 2D dans le cas d'un asservissement dans l'image (par exemple, les coordonnées d'un point dans le plan image) ou de primitives 3D dans le cas d'un asservissement dans l'espace cartésien (par exemple, les coordonnées d'un point dans le repère caméra).
- $\underline{S}^*$ est la valeur prise par $\underline{S}$ lorsque l'objectif de l'asservissement est atteint.

C.4.5 Les différentes approches en asservissement visuel

A.C. Sanderson et L.E. Weiss [1] ont introduit une importante classification dans le domaine de l'asservissement visuel. Celle-ci prend en compte deux critères :

1. l'espace de contrôle (2D ou 3D),
2. l'utilisation ou non d'une boucle interne.

Dans cette classification, deux approches ont fait l'objet d'importants développements :

1. *l'asservissement en situation (3D)* : à partir de primitives extraites de l'image et via un modèle de la cible, on détermine la position et l'orientation de celle-ci par rapport à la

caméra. L'erreur entre la configuration désirée et la configuration courante est exprimée dans l'espace cartésien.

2. *l'asservissement dans l'image (2D)* : la loi de commande est directement exprimée dans l'espace du capteur.

La seconde approche a donc l'avantage sur la première d'éviter l'étape d'estimation de la pose relative entre la cible et la caméra. En contrepartie, elle est exprimée dans un espace qui n'est pas celui où évolue l'effecteur.

On voit donc apparaître les deux grandes différences entre l'asservissement dans l'image et celui en situation.

C.4.6 Asservissement en situation ou asservissement dans l'espace opérationnel du robot

Les types d'information que l'on peut extraire d'une image sont nombreux et de plus ou moins haut niveau selon la complexité des algorithmes de traitement d'images utilisés. Cela va de la simple information sur l'intensité lumineuse d'un pixel jusqu'à la reconnaissance et la localisation de la forme d'un objet dans l'image. Il est nécessaire de localiser un objet dans l'espace cartésien (noté *localisation 3D*), pour ensuite pouvoir bouger le robot en conséquence. Pour réaliser cette localisation et reconstruire la scène 3D, il existe plusieurs manières de procéder avec le capteur vision :

- par vision stéréoscopique [21] voir aussi des systèmes à trois caméras,
- par vision dynamique ou active : une caméra effectuant des mouvements connus entre chaque acquisition et donnant des points de mesure de l'objet dans l'image,
- par fusion de données avec des capteurs proximétriques ou des télémètres renseignant sur la distance entre la caméra et l'environnement [74],
- par la connaissance *à priori* de la scène grâce à des *modèles* (notamment la connaissance de la distance scène/caméra ou de la taille des objets) associée à une localisation dans l'image (notée *localisation 2D*).

Grâce à la modélisation des objets, mais aussi à la modélisation de la caméra et du robot, le principe de cet asservissement donné figure C.11 est défini par P.I. Corke [26] selon la structure hiérarchique de la figure C.12.

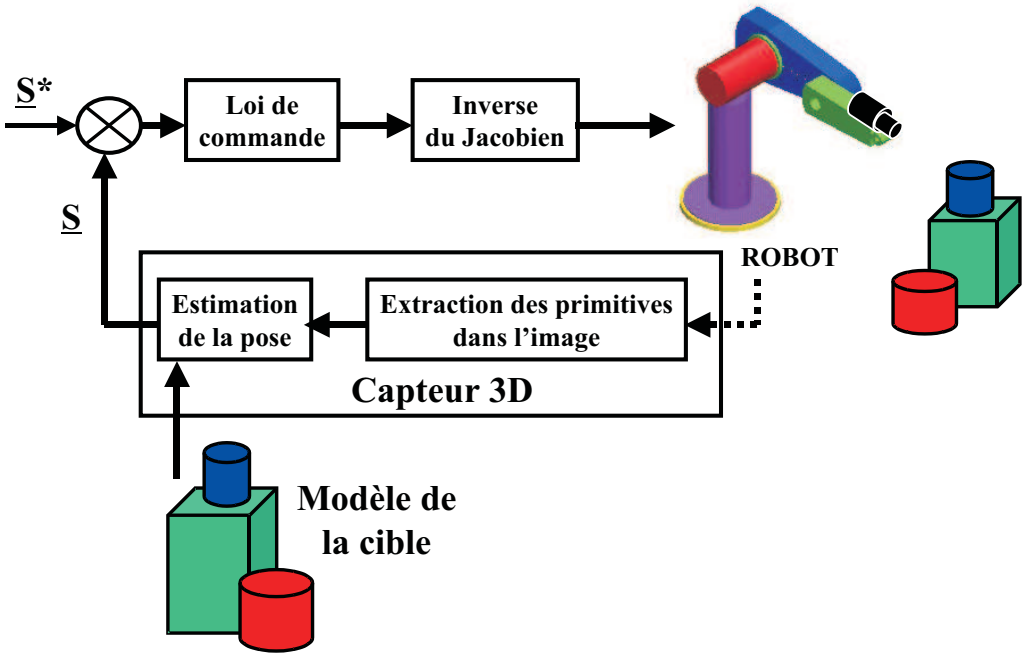


FIGURE C.11 – asservissement en situation (3D) pour une commande en vitesse.

On observe que cet asservissement peut être résumé de la manière suivante :

1. acquisition de l'image,
2. extraction des primitives visuelles,
3. estimation de la situation du l'effecteur du robot par rapport à l'environnement,
4. commande du robot.

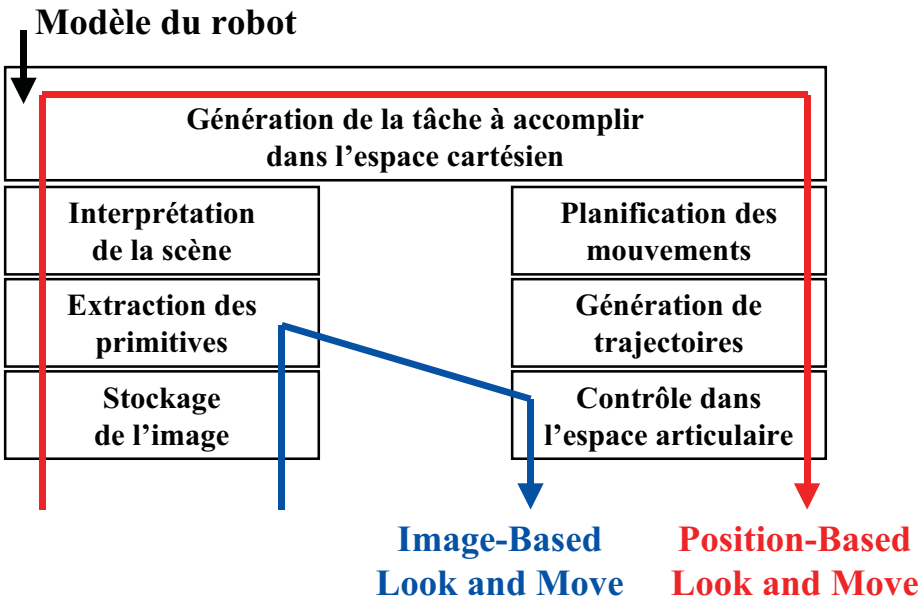


FIGURE C.12 – comparaison Position/Image - Based Look and Move.

Pour cet asservissement, les consignes de la tâche robotique à réaliser sont exprimées sous la forme d'une attitude de la caméra ($\underline{A}^*$) ou de l'effecteur ($\underline{F}^*$). D'où la fonction de tâche $\underline{E}$ suivante :

$$\underline{E} = C \cdot (\underline{A}(t) - \underline{A}^*) \quad \text{ou} \quad C \cdot (\underline{F}(t) - \underline{F}^*) \tag{C.5}$$

L'interprétation des informations visuelles nous donne la nouvelle position du robot dans son environnement et ceci directement dans l'espace cartésien. Le fait de travailler dans l'espace d'évolution réelle de l'effecteur constitue le principal avantage de cet asservissement. On constate que c'est typiquement un *asservissement en situation* et dans la littérature, il est identifié par le terme : **Position-Based Look and Move** (figure C.12). Cette technique se caractérise par les deux approches suivantes.

Asservissement en situation dit "statique"

Le fonctionnement consiste en un enchaînement *séquentiel* des traitements de l'information visuelle et de la commande. On est ici dans le cas d'une pseudo-boucle ouverte car il faut attendre la fin du déplacement du robot pour refaire un traitement sur l'estimation de la pose. Cela entraîne de faibles performances en terme de temps de réponse du système, et une utilisation de cet asservissement pour des objets uniquement statiques.

Asservissement en situation dit "dynamique"

Dans cette approche, les étapes de traitement et de commande s'effectuent en *parallèle*, mais à des cadences différentes. Cette structure autorise par conséquent la situation d'objets en mouvement car les traitements peuvent s'effectuer dès que les informations visuelles sont disponibles. Cependant, les temps de calcul des différents traitements doivent rester compatibles avec la vitesse de l'objet de façon à ce que l'objet reste dans le champ de vision de la caméra entre deux acquisitions.

En théorie, il suffit d'une seule itération pour passer de la situation courante (quelconque) à la situation voulue. Toutefois en pratique, ceci n'est pas réalisable à cause, non seulement de la dynamique du robot (qui limite l'amplitude de la commande), mais aussi des erreurs introduites par les éléments suivants :

- algorithmes d'extraction des informations visuelles avec les problèmes inhérents aux approximations de tout traitement d'images,
- modélisation de la caméra (problèmes de calibration),
- modélisation du robot (approximation du modèle).

L'accumulation de ces erreurs peut amener à ce que la situation réelle finale soit différente de la situation souhaitée alors que la convergence de la fonction de tâche est effectivement obtenue ($\underline{E} = 0$). Cela nécessite donc l'emploi d'algorithmes de vision itératifs. Ils utilisent des méthodes *d'extraction et de suivi de primitives visuelles* pour déterminer en permanence l'attitude de la caméra par rapport à la scène.

C.4.7 Asservissement dans l'image

L'avantage de cette technique est que la tâche à réaliser est directement définie dans l'espace image. Ceci diminue l'influence des problèmes liés à la modélisation de la caméra sur la boucle de commande. Autrement dit, l'interprétation des informations visuelles (figure C.12) pour reconstruire la pose courante de la caméra par rapport à son environnement, n'est plus nécessaire [50][59].

L'asservissement dans l'image donné figure C.13 est identifié dans la littérature par le terme : **Image-Based Look and Move** (figure C.12).

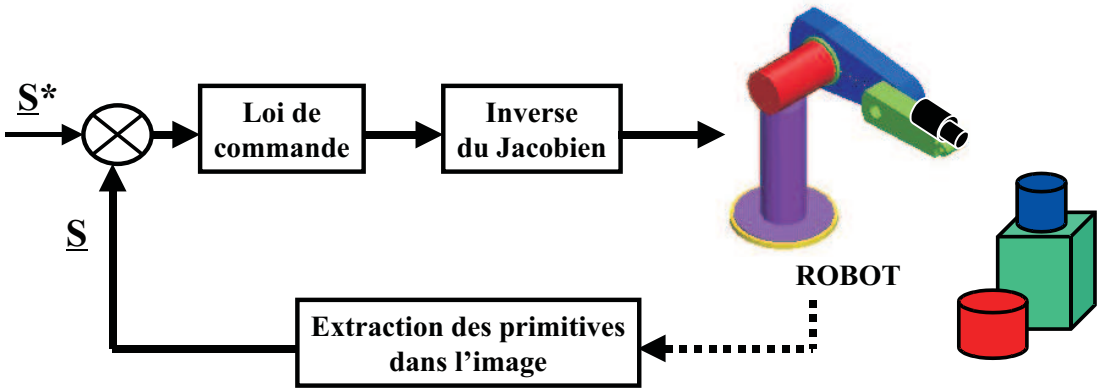


FIGURE C.13 – asservissement dans l'image (2D) pour une commande en vitesse.

Les consignes de la tâche robotique à réaliser ne sont plus exprimées sous la forme d'une attitude de la caméra $\underline{A}^*$ ou de l'effecteur $\underline{F}^*$, mais sous la forme d'informations visuelles $\underline{I}^*$ appelées *motif d'objet dans l'image*. D'où la fonction de tâche $\underline{E}$ suivante :

$$\underline{E} = C \cdot (\underline{I}(t) - \underline{I}^*) \tag{C.6}$$

Comme pour l'asservissement en situation, cet asservissement nécessite lui aussi l'emploi d'algorithmes de vision itératifs pour atteindre l'attitude désirée.

C.5 Plate-forme robotique du LASMEA

L'installation robotique du laboratoire est constituée de trois parties principales :

- le robot portique de la société "AFMA Robots",
- le rack VME (avec l'armoire électrique et la console),
- la station de travail Silicon Graphics O_2.

C.5.1 Le robot "AFMA Robots"

FIGURE C.14 – vue générale de la plate-forme robotique du LASMEA.

Le robot possède une structure cartésienne. Sa base est formée de deux ensembles de montants métalliques horizontaux et verticaux soutenant une pièce verticale qui porte l'effecteur (figure C.14). L'extrémité de ce dernier élément est munie d'un logement destiné à recevoir une

caméra vidéo (figure C.15).

Le robot possède six degrés de liberté qui se décomposent en trois axes de translation orthogonaux et trois rotations. Les directions des axes des translations $\vec{X_r}$, $\vec{Y_r}$ et $\vec{Z_r}$ sont définies à partir des montants du portique. Les articulations de l'effecteur permettent d'orienter la caméra suivant trois rotations autour d'axes $\vec{X_\alpha}$, $\vec{Y_\beta}$ et $\vec{Z_\gamma}$ respectivement.

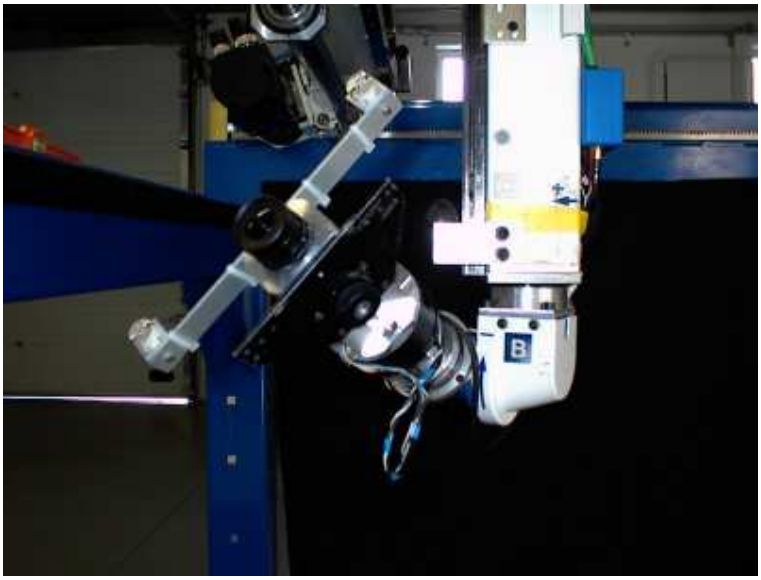


FIGURE C.15 – caméra embarquée.

Les caractéristiques générales de ce robot sont les suivantes :

	en Translation :	en Rotation :
Vitesse	1 m/s	2 rad/s (114.6°/s)
Accélération	4 m/s^2	8 rad/s^2
Débattement	X : 1.5 m	A (α) : 187.5°
Débattement	Y : 1.5 m	B (β) : 130°
Débattement	Z : 1 m	C (γ) : 130°
Précision	Statique : ± 0.5 mm	Dynamique± 1 mm
Répétabilité : ± 1 mm		

C.5.2 Le rack VME et la station de travail Silicon Graphics O_2

Pour les applications de vision, le robot est commandé à partir des informations collectées par la caméra vidéo embarquée sur l'effecteur. L'algorithme assurant le suivi de l'objet dans l'image et l'élaboration de la commande est exécuté sur une station de travail Silicon Graphics O_2 sous un environnement Unix multi-tâches. Ce type de machine a la particularité de posséder une entrée vidéo et une carte vidéo intégrée. Cet équipement permet de stocker les images en provenance de la caméra du robot.

Cette station de travail communique avec le rack VME via le réseau Ethernet local (figure C.16).

FIGURE C.16 – rack VME.

Le rack gère les dialogues entre la station et l'armoire de commande du robot par l'intermédiaire d'une application *temps réel* s'exécutant dans un environnement VxWorks.

Bibliographie

[1] Sanderson A.C. and Weiss L.E. Image-based visual servo control using relational graph error signals. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1074–1077, 1980.

[2] Adelson and Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, pages 1–20, 1991.

[3] Y. Adini, Y. Moses, and S. Ullman. Face recognition : the problem of compensating for changes in illumination direction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7) :721–732, July 1997.

[4] P. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. Tech. rep. cucs-035-90, Columbia Univ., Pittsburg, PA, September 1990.

[5] Pentland A.P. Local shading analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(2) :170–187, 1984.

[6] Arulampalam, Maskell, Gordon, and Clapp. A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing, Special Issue on Monte Carlo Methods*, 50(2) :174–188, February 2002.

[7] S. Basu and A. Pentland. A three-dimensional model of human lip motions trained from video. Technical Report 441, M.I.T. Media Laboratory Perceptual Computing Section, 1999.

[8] M. Ben-Ezra, S. Peleg, and M. Werman. Real-time motion analysis with linear programming. *Computer Vision and Image Understanding*, 78(1) :32–52, Apr 2000.

[9] F. Berry. *Contournement d'objet par asservissement visuel*. Thèse de doctorat à l'Ecole Doctorale Sciences pour l'Ingénieur de Clermont-Ferrand, 1999.

[10] F. Berry, P. Martinet, and J. Gallice. Trajectory generation by visual servoing. In *IEEE/RSJ International Conference on Intelligent Robot and Systems*, pages 1066–1072, Grenoble, France, September 1997.

[11] B. Bhanu and W. Burger. Qualitative understanding of scene dynamics for moving robots. In *the Int. J. Robotics Research*, volume 9, pages 74–90, 1990.

[12] Horn B.K.P. Understanding image intensities. *Artificial Intelligence*, 8 :201–231, 1977.

[13] Michael J. Black and Allan D. Jepson. Eigentracking : Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1) :63–84, 1998.

[14] A. Blake, R. Curwen, and A. Zisserman. A framework for spatio-temporal control in the tracking of visual contour. *International Journal of Computer Vision*, 11(2) :127–145, 1993.

[15] M. Bober and J. Kittler. A hough transform based hierarchical algorithm for motion segmentation and estimation. In *Intl Workshop on Time -Varying Image Processing and Moving Object Recognition*, pages 335–342, Florence, Italie, Juillet 1993.

[16] François Bérard. *Vision par ordinateur pour l'interaction homme-machine fortement couplé*. PhD thesis, Université Joseph Fourier Grenoble I, Grenoble, France, 1999.

[17] R. Brunelli and T. Poggio. Template matching : Matched spatial filter and beyond. A.I. Memo 1549, M.I.T., October 1995.

[18] F. Chaumette. *La relation vision-commande : théorie et application à des tâches robotiques*. Thèse de doctorat de l'Institut de Formation Supérieure en Informatique et Communication de Rennes, 1990.

[19] H. Chen, P. Belhumeur, and D. Jacobs. In search of illumination invariants. In *the CVPR00*, volume 1, pages 254–261, 2000.

[20] Sung-Il Chien and Si-Hun Sung. Adaptive window method with sizing vectors for reliable correlation-based target tracking. *Pattern Recognition*, 33 :237–249, 2000.

[21] J. Chung and N. Ohinishi. Self-organizing reaching opération in a simulated robot via interaction of binocular and arm movements. In *IEEE/RSJ International Conference on Intelligent Robot and Systems*, pages 37–42, Grenoble, France, september 1997.

[22] M. Cole. Algorithmic skeletons : structured management of parallel computations. *Pitman/MIT Press*, 1989.

[23] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1) :38–59, 1995.

[24] T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 227–232, Grenoble, France, 2000.

[25] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appareance models. In *the European Conference on Computer Vision*, volume 2, pages 484–498, 1998.

[26] P. I. Corke. *Visual Control of Robots : high-performance visual servoing.* Professor J.Billingsley, 1996.

[27] R. Coudarcher. *Composition de squelettes algorithmiques : application au prototypage rapide d'applications de vision.* Thèse de doctorat à l'Ecole Doctorale Sciences pour l'Ingénieur de Clermont-Ferrand, décembre 2002.

[28] Rémi Coudarcher and Florent Duculty. Algorithmic skeleton nesting with skipper-2 : A case study. Technical report, LASMEA UMR 6602 du CNRS, 2002.

[29] Rémi Coudarcher, Florent Duculty, Jocelyn Sérot, Frédéric Jurie, Jean-Pierre Dérutin, and Michel Dhome. Algorithmic skeleton nesting parallelisation of a 3d face tracking from appearance with the skipper-2 parallel programming environment. In *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, volume 5 (Parallel and Distributed Systems), Orlando, Florida (USA), July 2002.

[30] Rémi Coudarcher, Florent Duculty, Jocelyn Sérot, Frédéric Jurie, Jean-Pierre Dérutin, and Michel Dhome. Parallelisation of a face tracking algorithm with the skipper-2 parallel programming environment. In *International Workshop on Computer Architectures for Machine Perception (CAMP)*, New Orleans (USA), May 2003.

[31] Rémi Coudarcher, Jocelyn Sérot, and Jean-Pierre Dérutin. Implementation of a skeleton-based parallel programming environment supporting arbitrary nesting. In *the 6th International Workshop on High-Level Parallel Programming Models and Supportive Environments*, volume LNCS 2026, pages 71–85, San Francisco, California, USA, April 2001.

[32] J.L. Crowley and J.M. Bedrune. Integration and control of reactive visual processes. In *Proc. European Conference on Computer Vision*, pages 47–58, Stockholm, Sweeden, 1994.

[33] Curwen and Blake. Dynamic contours : Real-time active splines. *Active Vision, Blake and Yuille, eds., MIT Press*, pages 39–58, 1992.

[34] J. Darlington, Y.-K. Guo, Hing Wing To, and J. Yang. Functional skeletons for parallel coordination. *Lecture Notes in Computer Science*, 966 :55–65, 1995.

[35] T. Darrell, I.A. Essa, and A.P. Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(12) :1236–1242, 1996.

[36] Vincent Colin de Verdière and James L. Crowley. Visual recognition using local appearance. In *the Fifth European Conference on Computer Vision*, pages xx–yy, 1998.

[37] D. DeCarlo and D. Metaxas. Deformable model-based face shape and motion estimation. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 146–150, 1996.

[38] K. Deguchi and T. Noguchi. Visual servoing using eigenspace method and dynamic calculation of interaction matrices. In *ICPR96*, page A7E.3, 1996.

[39] Frank Dellaert and Robert Collins. Fast image-based tracking by selective pixel integration. In *ICCV Workshop on Frame-Rate Vision*, pages xx–yy, Greece, September 1999.

[40] Joachim Denzler and Dietrich W. R. Paulus. Active motion detection and object tracking. In *the Proceedings on the 1st International Conference on Image Processing*, volume 3, Austin,Texas, November 13th-16th 1994.

[41] Dubuisson, Lakshmanan, and Jain. Vehicle segmentation and classification using deformable templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(3) :293–308, 1996.

[42] Florent Duculty, Michel Dhome, and Frédéric Jurie. Suivi de motif en temps réel dans des séquences d'images. In *Premières Rencontres des Sciences et Technologies de l'Information ASTI*, Cité des Sciences et de l'Industrie, La Villette, Paris, April 2001.

[43] Florent Duculty, Michel Dhome, and Frédéric Jurie. Suivi d'objets 3d basé sur l'apparence. In *Congrès francophone de Vision par Ordinateur ORASIS*, pages 491–500, Cahors, June 2001.

[44] Florent Duculty, Michel Dhome, and Frédéric Jurie. Suivi d'objets 3d basé sur l'apparence. In *Premières Rencontres Laboratoires-Entreprises VISIOMIP (Vision par Ordinateur en Midi-Pyrénées)*, pages 16–17, Cahors, June 2001.

[45] Florent Duculty, Michel Dhome, and Frédéric Jurie. Suivi efficace d'objets 3d basé sur l'apparence (efficient tracking of 3d objects from appearance). *Traitement du Signal*, 18(5-6) :403–417, 2001.

[46] Florent Duculty, Michel Dhome, and Frédéric Jurie. Tracking of 3d objects from appearance. In *the 12th Scandinavian Conference on Image Analysis*, pages 515–522, Bergen, Norway, June 2001.

[47] Florent Duculty, Michel Dhome, and Frédéric Jurie. Real time 3d face tracking from appearance. In *the IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 581–584, Rochester, New York (USA), September 2002.

[48] Florent Duculty, Michel Dhome, and Frédéric Jurie. Suivi temps réel d'objets 3d basé sur l'apparence (real time tracking of 3d objects from appearance). In *13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA)*, volume 1, pages 67–76, Angers, January 2002.

[49] J. H. Duncan and T. Chou. Temporal edges : the detection of motion and the computation of optical flow. In *the Proc. Second Int. Conf. Comput. Vision*, pages 374–382, Tampa, FL, December 1988.

[50] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. In *IEEE Transactions on Robotics and Automation*, volume 8, pages 313–326, June 1992.

[51] H. Farid and E. Adelson. Separating reflections and lighting using independent components analysis. In *the CVPR99*, volume 1, pages 262–267, 1999.

[52] David J. Fleet, Michael J. Black, Yaser Yacoob, and Allan D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3) :171–193, 2000.

[53] Jr. G. Legters and T. Young. A mathematical model for computer image tracking. *IEEE Trans. on PAMI*, 4(6) :583–594, 1982.

[54] Dominique Ginhac, Jocelyn Sérot, and Jean-Pierre Dérutin. Fast prototyping of image processing applications using functional skeletons on a mimd-dm architecture. In *IAPR Workshop on Machine Vision and Applications*, pages 468–471, Chiba, Japan, November 1998.

[55] M. Gleicher. Projective registration with difference decomposition. In *CVPR97*, pages 331–337, 1997.

[56] S. Gong, S. McKenna, and A. Psarrou. Dynamic vision : From images to face recognition. *World Scientific Publishing and Imperial College Press*, pages xx–yy, April 2000.

[57] S. Gong, A. Psarrou, I. Katsouli, and P. Palavouzis. Tracking and recognition of face sequences. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, pages 96–112, Hamburg, Germany, 1994.

[58] G. Hager. Calibration-free visual control using projective invariance. In *IEEE International Conference on Computer Vision*, pages 1009–1015, Cambridge, USA, June 1995.

[59] G. Hager, S. Hutchinson, and P. Corke. A tutorial on visual servo control. In *IEEE International Conference on Robotics and Automation*, Minneapolis, USA, april 1996.

[60] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on PAMI*, 20(10) :1025–1039, 1998.

[61] Barrow H.G. and Tenenbaum J.M. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17 :75–116, 1981.

[62] Joachim Hornegger, Heinrich Niemann, and Robert Risack. Appearance-based object recognition using optimal feature transforms. *Pattern Recognition*, 33 :209–224, 2000.

[63] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1) :5–28, 1998.

[64] D. Jacobs, P. Belhumeur, and R. Basri. Comparing images under variable illumination. In *Proceedings of the CVPR98*, pages 610–617, 1998.

[65] Koenderink J.J. and Van Doorn D. The shape of smooth objects and the way contours end. *Perception*, 11 :129–137, 1982.

[66] F. Jurie and M. Dhome. Un algorithme efficace de suivi d'objets dans des séquences d'images. In *Congrès francophone RFIA*, volume 1, pages 537–546, Paris, February 2000.

[67] F. Jurie and M. Dhome. Real time template matching : an efficient approach. In *the 12th International Conference on Computer Vision*, pages 544–549, Vancouver, Canada, July 2001.

[68] Frédéric Jurie and Michel Dhome. Hyperplane approximation for template matching. *IEEE Trans. on PAMI*, 24(7) :996–1000, 2002.

[69] Nakayama K., Shimojo S., and Silverman G.H. Stereoscopic depth : its relation to image segmentation, grouping, and the recognition of occluded objects. *Perception*, 18 :55–68, 1989.

[70] R. E. Kalman. A new approach to linear filtering and prediction problems. In *Trans ASME*, volume 82, pages 34 – 45, 1973.

[71] Kass, Witkin, and Terzopoulos. Snakes : Active contour models. *International Journal of Computer Vision*, 1(4) :321–331, 1987.

[72] D. Koller, K. Daniilidis, and H.H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3) :257–281, 1993.

[73] H. Kollnig and H.H. Nagel. 3d pose estimation by directly matching polyhedral models to gray value gradients. *International Journal of Computer Vision*, 23(3) :283–302, 1997.

[74] I. Konukseven, B. Kaftanoglu, and T. Balkan. Multisensor controlled robotic tracking and automatic pick and place. In *IEEE/RSJ International Conference on Intelligent Robot and Systems*, pages 1356–1362, Grenoble, France, september 1997.

[75] A. Kuchinsky, C. Pering, M. L. Creech, D. Freeze, B. Serra, and J. Gwizdka. Fotofile : a consumer multimedia organization and retrieval system. In *Proc. ACM HCI'99 Conference*, 1999.

[76] M. La Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination : An approach based on registration of textured-mapped 3d models. *PAMI*, 22(4) :322–336, April 2000.

[77] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C. Malsburg, R. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3) :300–311, 1993.

[78] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic tracking, coding and reconstruction of human faces, using flexible appearance models. *IEEE Electron. Lett.*, 30 :1578–1579, 1994.

[79] Yongmin Li, Shaogang Gong, and Heather Liddell. Support vector regression and classification based multi-view face detection and recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 300–305, Grenoble, France, 2000.

[80] H. G. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. In *the Proc. Roy. Soc. London, B*, volume 208, pages 385–397, 1980.

[81] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2) :113–122, 1992.

[82] Riley M.D. The representation of image texture. AI-TR 649, M.I.T. AI-LAB, September 1981.

[83] Metaxas and Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6) :580–591, 1993.

[84] B. Moghaddam and A. Pentland. Face recognition using view-based and modular eigenspaces. *Automatic systems for the identification and inspection for humans*, 2277, 1994.

[85] B. Moghaddam and A. Pentland. A subspace method for maximum likelihood taget detection. Technical report, M.I.T. Media Laboratory Perceptual Computing Section, 1995.

[86] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14 :5–24, 1995.

[87] Don Murray and Anup Basu. Motion tracking with an active camera. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 16(5) :449–459, May 1994.

[88] Shree K. Nayar and Ruud M. Bolle. Reflectance based object recognition. *International Journal of Computer Vision*, 17(3) :219–240, 1996.

[89] Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase. Subspace methods for robot vision. *IEEE Transactions on Robotics and Automation on Vision-Based Control of Robot Manipulators*, 12(5) :750–758, october 1996.

[90] Hieu T. Nguyen, Marcel Worring, and Rein van den Boomgaard. Occlusion robust adaptive template tracking. In *the 12th International Conference on Computer Vision*, volume 1, pages 678–683, Vancouver, Canada, July 2001.

[91] N. Papanikolopoulos, P. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot : A combination of control and vision. *IEEE Trans. on Robotics and Automation*, 9 :14–35, 1993.

[92] Soo-Chang Pei and Lin-Gwo Liou. Vehicle-type motion estimation by the fusion of image point and line features. *Pattern Recognition*, 31(3) :333–344, 1998.

[93] Massimiliano Pontil and Alessandro Verri. Support vector machines for 3d object recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 20(6) :637–646, June 1998.

[94] Arthur R. Pope and David G. Lowe. Learning 3d object recognition models from 2d images. In *AAAI-MLCV96*, 1996.

[95] Arthur R. Pope and David G. Lowe. Learning appearance models for object recognition. In *ORCV96*, 1996.

[96] K. Prazdny. Determining the instantaneous direction of motion from optic flow generated by a curvilinearly moving observer. *Computer Vision Graphics, Image Processing*, 17 :238–248, 1981.

[97] S. Romdhani, S. Gong, and A. Psarrou. A multi-view non-linear active shape model using kernel pca. In *British Machine Vision Conference*, pages 483–492, Nottingham, UK, 1999.

[98] C. Samson, B. Espiau, and M. Le Borgne. Robot control : The task function approach. *Oxford University Press*, 1991.

[99] B. G. Schunk. The image flow constraint equation. *Computer Vision Graphics, Image Processing*, 35 :20–40, 1986.

[100] Jocelyn Sérot, Dominique Ginhac, Roland Chapuis, and Jean-Pierre Dérutin. Fast prototyping of parallel-vision applications using functional skeletons. *Machine Vision and Applications*, 12 :271–290, 2001.

[101] D. B. Skillicorn. Architecture-independent parallel computation. *IEEE Computer*, 23(12) :38–50, December 1990.

[102] A. Smolic, J.-R. Ohm, and T. Sikora. Object-based global motion estimation using a combined feature matching and optical flow approach. In *Proc. VLBV'98*, Urbana, IL, USA, October 8-9 1998.

[103] C. Stiller and J. Konrad. Estimating motion in image sequences : A tutorial on modeling and computation of 2d motion. *IEEE Signal Process. Mag.*, 16 :70–91, July 1999.

[104] J. Strom, T. Jebara, S. Basu, and A. Pentland. Real time tracking and modeling of faces : An ekf-based analysis by synthesis approach. Technical Report 506, M.I.T. Media Laboratory Perceptual Computing Section, 1999.

[105] Kanade T. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17 :409–460, 1981.

[106] I. Takahashi and K. Deguchi. Image-based control of robot and target objet motion by eigen space method. In *IARP Workshop on Machine Vision Applications*, pages 472–475, Makuhari, Chiba, Japan, November 1998.

[107] Peng-Seng Tho and Andrew K. Forrest. Occlusion detection in early vision. In *Int. Conf. on Comp. Vision*, pages 126–132, Osaka, Japan, December 1990.

[108] M. Tonkoand, J. Schurmann, K. Schafer, and H. Nagel. Visually gripping of a ued car battery. In *IEEE/RSJ International Conference on Intelligent Robot and Systems*, pages 49–54, Grenoble, France, september 1997.

[109] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1) :71–86, 1991.

[110] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 19(7) :733–742, 1997.

[111] Thompson W.B. and Whillock R.P. Occlusion-sensitive matching. In *Proc. Int. Conf. on Comp. Vision*, pages 285–289, 1988.

[112] J. J. Wu, R. E. Rink, T. M. Caelli, and V. G. Gourishankar. Recovery of the 3-d location and motion of a rigid object through camera image. *International Journal of Computer Vision*, 3 :373–394, 1989.

[113] O. Yamaguchi, K. Fukui, and K. Maeda. Face recognition using temporal image sequence. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 318–323, Nara, Japan, 1998.

[114] G.S. Young and R. Chellappa. 3d motion estimation using a sequence of noisy stereo images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 710–716, Ann Arbor, Michigan, 1988.

[115] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8 :99–111, 1992.