Reconnaissance de gestes et actions pour la collaboration homme-robot sur chaîne de montage

Eva Coupeté

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à MINES ParisTech

Reconnaissance de gestes pour la collaboration homme-robot sur chaîne de montage

Ecole doctorale n°432

Sciences des Métiers de l'Ingénieur

Spécialité Informatique temps réel, robotique et automatique

COMPOSITION DU JURY :

Mme Catherine Achard
Université Pierre et Marie Curie, Rapporteur

M. Laurent Grisoni
Université Lille 1, Rapporteur

M. Sylvain Calinon
Institut de recherche Idiap, Examinateur

M. Philippe Fraisse
Université Montpellier 2, Examinateur

M. Sotiris Manitsaris
Mines ParisTech, Examinateur

M. Fabien Moutarde
Mines ParisTech, Examinateur

Soutenue par Eva Coupeté
le 10 novembre 2016

Directeur de thèse **Fabien Moutarde**
Maître de thèse **Sotiris Manitsaris**

MINES ParisTech | PSL RESEARCH UNIVERSITY PARIS

Table des matières

1 Contexte **7**
- 1.1 La robotique collaborative . . . 8
  - 1.1.1 Émergence des robots collaboratifs . . . 8
  - 1.1.2 Les enjeux de la robotique collaborative dans l'industrie . . . 8
- 1.2 La reconnaissance de gestes : nouvelle voie d'interaction . . . 10
  - 1.2.1 Interaction avec des systèmes intelligents . . . 10
  - 1.2.2 Les contraintes du milieu industriel . . . 10
- 1.3 Organisation du manuscrit . . . 11

2 La collaboration homme-robot **13**
- 2.1 Etat de l'art sur la collaboration homme-robot . . . 13
  - 2.1.1 Les robots collaboratifs . . . 14
  - 2.1.2 Le cas particulier de l'industrie . . . 19
- 2.2 Cas d'étude . . . 25
  - 2.2.1 Cas de co-présence . . . 26
  - 2.2.2 Cas de collaboration . . . 29
- 2.3 Conclusion . . . 32

3 La reconnaissance de gestes : état de l'art **35**
- 3.1 Le geste technique . . . 36
- 3.2 La reconnaissance de gestes : vue d'ensemble . . . 37
- 3.3 La reconnaissance de gestes avec des données vidéos . . . 39
  - 3.3.1 Les caméras . . . 39
  - 3.3.2 Méthodes basées sur des points d'intérêts . . . 43
  - 3.3.3 Méthodes basées sur des silhouettes . . . 45
  - 3.3.4 Analyse de la succession de postures . . . 47
- 3.4 La reconnaissance de gestes avec un équipement intrusif . . . 50
  - 3.4.1 Les capteurs inertiels . . . 50
  - 3.4.2 Les marqueurs optiques . . . 52
- 3.5 La reconnaissance des gestes avec un système hybride vision et inertiel . . . 53
- 3.6 Classification des gestes . . . 53
  - 3.6.1 Classification directe . . . 55
  - 3.6.2 Classification avec une approche temporelle . . . 60
- 3.7 Conclusion sur la reconnaissance de gestes . . . 67

4 Reconnaissance des gestes techniques avec un équipement intrusif **69**
- 4.1 Protocole d'acquisition de la base de données . . . 69
- 4.2 Méthodologie de reconnaissance des gestes . . . 70
  - 4.2.1 Présentation du *Gesture Follower* . . . 70
  - 4.2.2 Reconnaissance des gestes isolés . . . 71

4.2.3 Reconnaissance en temps réel . . . 72
4.3 Résultats . . . 72
4.3.1 Reconnaissance de gestes isolés . . . 72
4.3.2 Reconnaissance en temps réel . . . 73
4.4 Conclusion de l'étude . . . 76

5 Suivi des mains de l'opérateur 79
5.1 Description d'une image de profondeur . . . 79
5.2 Le suivi en plusieurs étapes . . . 80
5.2.1 Graphe 2D du torse de l'opérateur . . . 81
5.2.2 Trouver les mains . . . 84
5.3 Implémentation et résultats . . . 87
5.3.1 Choix d'implémentation . . . 87
5.3.2 Rapidité de l'algorithme . . . 88
5.3.3 Précision de l'algorithme . . . 89
5.4 Conclusion . . . 93

6 Reconnaissance des gestes techniques avec des capteurs non intrusifs 95
6.1 Acquisition d'une base de données . . . 96
6.1.1 Protocole d'acquisition de la base de données . . . 96
6.1.2 Analyse de la base de données . . . 97
6.2 Reconnaissance des gestes techniques . . . 101
6.2.1 Méthodologie de reconnaissance des gestes . . . 101
6.2.2 Choix du descripteur . . . 102
6.2.3 Critères d'évaluation . . . 104
6.2.4 Résultats . . . 107
6.2.5 Temps de calcul . . . 114
6.3 Adaptation de la base de données pour un nouvel opérateur . . . 115
6.3.1 Méthodologie . . . 115
6.3.2 Résultats . . . 116
6.3.3 Comparaison avec une base d'apprentissage mono-opérateur . . . 117
6.4 Utilisation du capteur inertiel sur les outils pour affiner la reconnaissance des gestes basée sur la vision . . . 118
6.4.1 Méthodologie . . . 119
6.4.2 Acquisition et traitement des données issues du capteur inertiel . . . 119
6.4.3 Résultats . . . 120
6.5 Gestion des gestes inattendus . . . 124
6.5.1 Présentation du problème . . . 124
6.5.2 Réaction de notre système de reconnaissance face à des gestes parasites . . . 126
6.6 Mise en place d'un démonstrateur . . . 128
6.7 Conclusion et synthèse . . . 129

7 Conclusions et perspectives 131

Bibliographie 134

Liste des figures

1.1 Evolution des robots industriels 8
1.2 Dessins sur la collaboration homme robot 9
1.3 Interaction avec des sytèmes intelligents via des gestes 10

2.1 Collaboration homme-robot 14
2.2 Exemple d'exosquelettes 15
2.3 Exemple de robots téléopérés 16
2.4 Exemple de robots assistant industriels 17
2.5 Exemple de bras robotiques 18
2.6 Exemple de robots antropomorphes 19
2.7 Robots-assitants de service 19
2.8 Gains financiers et en flexibilité grâce à l'arrivée de cellules hybrides sur les chaînes de montage 20
2.9 Capteurs anti-collision 21
2.10 Méthodes d'évitement de collision 22
2.11 Sécurité des robots assistants insutriels 22
2.12 Évaluation des risques de blessures liées à la collaboration avec un robot . . 23
2.13 INteraction avec des robots collaboratifs 25
2.14 Cas d'étude de co-présence 26
2.15 Cellule expérimentale pour l'étude de la co-presence 27
2.16 Objets à appliquer sur la porte de voiture 27
2.17 Schémas des quatre gestes à reconnaître dans le cas de la co-présence . . . 28
2.18 Capteurs inertiels 29
2.19 Cellule expérimentale pour l'étude de la collaboration 29
2.20 Pièces à assembler sur le poste collaboratif 30
2.21 Actions pour l'assemblage du raccord de moteur 31
2.22 Schémas des cinq gestes à reconnaître dans le cas d'étude collaboratif . . . 32
2.23 Configuration du poste de collaboration 32

3.1 Gestes techniques 36
3.2 Exemple de trame pour la reconnaissance de gestes avec des données vidéo 39
3.3 Modèle géométrique simplifié du système de stéréo-vision 40
3.4 Exemples de cartes de profondeurs 41
3.5 Illsutration d'une caméra temps de vol 42
3.6 Points d'intérêts issus des détecteurs Harris 3D et cuboïde 44
3.7 Illustration de trajectoires pour la reconnaissance d'actions 45
3.8 Extraction de silhouettes pour la reconnaissance de gestes 46
3.9 Construction de surface 3D 46
3.10 Illustration de *Shape flow* et *Motion History Volumes* 47

3.11 Exemple de squelettisation de personnes dans des images RGB grace à un méthode de *deep-learning* . . . 48
3.12 Squelettisation d'un personne filmée avec une caméra de profondeur (i) . . 48
3.13 Squelettisation d'un personne filmée avec une caméra de profondeur (ii) . . 49
3.14 Angles d'Euler . . . 50
3.15 "*Acitivity Recognition Chain*" . . . 51
3.16 Marqueurs optiques . . . 52
3.17 Illustration de la méthode des Sacs de Mots . . . 55
3.18 Séparateur à Vaste Marge (SVM) . . . 56
3.19 Méthode des k-Plus Proches Voisins . . . 57
3.20 Exemple d'une architecture d'un réseau de neurones à convolution . . . 57
3.21 Combinaison de trajectoires et de descripteurs trouvées par des CNN pour la reconnaissance d'actions . . . 59
3.22 Chemin optimal selon la méthode DTW . . . 60
3.23 Strcutures des HMMs . . . 62
3.24 Illustration d'un HMM . . . 63
3.25 Partitionnement de données avec un K-Means . . . 65

4.1 Temps moyen d'exécution des gestes dans le cas d'étude de la co-présence . 70
4.2 Le *Gesture Follower* . . . 71
4.3 Données brutes issues du *Gesture Follower* pour la reconnaissance en temps réel . . . 72
4.4 Application de différents seuils pour la reconnaissance de gestes en temps réel . . . 74
4.5 Utilisation de fenêtres glissantes de tailles variables pour la reconnaissance de gestes en temps réel . . . 75

5.1 Exemple d'une carte de profondeur sur le cas d'étude de collaboration . . . 80
5.2 Étapes de l'algorithme de suivi des mains . . . 81
5.3 Détection de la tête de l'opérateur . . . 81
5.4 Correction des artéfacts autour de la tête . . . 82
5.5 Illustration de la construction d'u graphe 2D . . . 83
5.6 Cas particulier des mains jointes dans la construction du graphe . . . 84
5.7 Calcul des distances géodésiques du haut du corps de l'opérateur . . . 84
5.8 Positions des mains, de la tête et les chemins les plus courts . . . 86
5.9 Logo d'OpenCV . . . 87
5.10 Suivi des mains sur 100 images consécutives . . . 91
5.11 Suivi des mains sur 160 images consécutives . . . 92

6.1 Temps d'éxécution moyen de chaque geste dans le cas d'étude collaboratif . 97
6.2 Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 1, attraper une pièce à gauche . . . 98
6.3 Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 2, attraper une pièce à droite . . . 98
6.4 Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 3, assembler deux pièces . . . 99
6.5 Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 4, visser . . . 99
6.6 Projection sur le plan XZ des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 3, assembler . . . 100

6.7 Projection sur le plan XZ des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 4, visser 100
6.8 Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 5, poser la pièce terminée dans une boîte 101
6.9 Méthodologie de la reconnaissance des gestes utilisant des K-Means et des HMMs 102
6.10 Echantillonage des chemins les plus courts 103
6.11 Positions des mains et de la tête 103
6.12 Choix du référentiel 104
6.13 Illsutration de nos critères d'évaluation des performances de reconnaissance de gestes isolés 105
6.14 Méthodologie de la reconnaissance des gestes en temps réel 106
6.15 Méthodologie pour adapter notre base de donées 116
6.16 Utilisation d'un capteur inertiel sur les outils pour affiner la reconnaissance de gestes 119
6.17 Données brutes issues d'un capteur inertiel 120
6.18 Exemples de gestes parasites 125
6.19 Projection, sur le plan XY, dans le repère lié à la scène, des trajectoires 3D des mains gauche et droite lors de l'exécution de gestes parasites 125
6.20 Projection, sur le plan XZ, dans le repère lié à la scène, des trajectoires 3D des mains gauche et droite lors de l'exécution de gestes parasites 125
6.21 Reconnaissance en temps réel et suivi de la position des mains 128

Liste des tableaux

3.1 Descripteurs pour la reconnaissance de gestes en fonction du/des capteurs utilisé(s) . . . 38
3.2 Comparaison de caractéristiques de caméras 3D . . . 42
3.3 Classification des gestes en fonctions des descripteurs utilisés . . . 54

4.1 Résultats avec les MotionPods . . . 73
4.2 Résultats avec l'Animazoo . . . 73
4.3 Précision et rappel pour chaque geste, seuls les gestes avec une probabilité d'au moins 0.7 sont prises en compte . . . 76
4.4 Précision et rappel pour chaque geste, seuls les gestes avec une probabilité d'au moins 0.9 sont prises en compte . . . 76

5.1 Précision de l'algorithme de localisation des mains . . . 89

6.1 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel de la caméra, 3 échantillons des chemins 3D les plus courts reliant la tête aux mains . . . 107
6.2 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel de la caméra, 7 échantillons des chemins 3D les plus courts reliant la tête aux mains . . . 108
6.3 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel de la caméra, 15 échantillons des chemins 3D les plus courts reliant la tête aux mains . . . 108
6.4 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel de la caméra, Positions 3D des mains et de la tête . . . 109
6.5 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Méthode de jackknife, référentiel de la caméra, Positions 3D des mains . . . 109
6.6 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel lié à l'opérateur, positions 3D des mains . . . 110
6.7 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel lié à la scène, positions 3D des mains . . . 111
6.8 Tableau récapitulatif des résultats de reconnaissance selon plusieurs référentiels et plusieurs vecteurs-descripteurs avec le critère jackknife . . . 111
6.9 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère 80%-20%, référentiel lié à la scène, positions 3D des mains . . . 112
6.10 Résultats de reconnaissance en temps réel en utilisant le critère jackknife . 113
6.11 Résultats de reconnaissance en temps réel en utilisant le critère 80%-20% . 114
6.12 Temps de calcul pour la reconnaissance des gestes . . . 114
6.13 Temps de calcul, en secondes, pour l'apprentissage des HMMs . . . 115

6.14 Évolution des taux de reconnaissance en fonction du nombre de sets ajoutés de l'opérateur "test" dans la base d'apprentissage 116
6.15 Évolution de la précision en fonction du nombre de sets ajoutés 117
6.16 Évolution du rappel en fonction du nombre de sets ajoutés 117
6.17 Reconnaissance mono-opérateur. Résultats en fonction du nombre de sets de gestes dans la base d'apprentissage 118
6.18 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife, référentiel lié à la scène, positions 3D des mains et utilisation des données issues du capteur inertiel 121
6.19 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère jackknife avec adaptation de la base de données avec 5 sets de gestes, référentiel lié à la scène, positions 3D des mains et utilisation des données issues du capteur inertiel 121
6.20 Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère 80%-20%, référentiel lié à la scène, positions 3D des mains et utilisation des données issues du capteur inertiel 122
6.21 Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* avec des données provenant de la vision et du capteur inertiel sur la visseuse 123
6.22 Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* et une adaptation de la base de données avec 5 sets de gestes, utilisation des données provenant de la vision et du capteur inertiel sur la visseuse 123
6.23 Résultats de reconnaissance en temps réel en utilisant le critère 80%-20% avec des données provenant de la vision et du capteur inertiel sur la visseuse 124
6.24 Sortie(s) notre système (geste reconnu) lors de l'exécution de gestes parasites avec des données provenant seulement de la vision en utilisant le critère d'évaluation *jackknife* 126
6.25 Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* sur des séquences de gestes comprenant des gestes parasites (RC : taux de reconnaissances correctes) 127
6.26 Résultats de reconnaissance en temps réel en utilisant le critère *jackknife*, avec une adaptation de la base d'apprentissage avec 5 sets de gestes, sur des séquences de gestes comprenant des gestes parasites en utilisant des données provenant de la vision et du capteur inertiel (RC : taux de reconnaissances correctes) 127
6.27 Tableau récapitulatif de nos résultats sur des séquences de gestes enchaînés avec le critère *jackknife* 129

Remerciements

Durant ces trois années, j'ai réalisé qu'effectuer un doctorat n'est pas un travail en solitaire, mais une expérience riche de rencontres et d'échanges. Je suis convaincue aujourd'hui que la thèse n'est pas simplement mettre en forme une ou plusieurs problématiques scientifiques pour y proposer des solutions mais également une formation bien plus large sur une nouvelle manière de penser et de réfléchir, une ouverture sur le monde, une remise en question permanente et un exercice pour apprendre à avoir confiance en soi et persévérer. Le travail effectué pendant cette période a été fortement influencé par les discussions passionnantes et les conseils avisés que j'ai pu recevoir et qui ont aidé à façonner mes idées. Je souhaite remercier les personnes qui ont fait partie de cette aventure.

Je tiens tout d'abord à remercier mes encadrants de thèse, Fabien Moutarde et Sotiris Manitsaris. Ils ont été une source d'écoute et de dialogue ainsi que de conseils pendant ces trois années. Merci de m'avoir guidée dans de bonnes directions tout en m'accordant assez de confiance pour me laisser aborder la thèse avec une grande liberté. Cela m'a permis de m'approprier les problématiques liées au sujet de la thèse et de chercher par moi-même des solutions. Merci pour votre esprit critique, j'ai beaucoup appris de nos discussions et du chemin que nous avons parcouru ensemble lors de cette période.

Dans un second temps je souhaiterais remercier Philippe Fraisse qui m'a fait l'honneur de présider mon jury de thèse. Je remercie également Catherine Achard et Laurent Grisoni qui ont accepté de rapporter cette thèse. Les remarques que vous avez faites suite à la lecture du manuscrit m'ont permis de prendre du recul sur mon travail et d'améliorer le document final. Je remercie Sylvain Calinon qui a accepté de faire partie de mon jury, s'est intéressé à mon sujet et s'est déplacé jusqu'à Paris le jour de la soutenance.

Ce travail de thèse a été fait au sein de la chaire «Robotique et Réalité Virtuelle» liant PSA Peugeot-Citroën et Mines ParisTech. Je souhaite donc remercier les membres de la chaire sans qui ce travail n'aurait pas existé. Je souhaiterais remercier l'ensemble du personnel de l'équipe NOVA (NOuvelles Voies d'Automatisation) de PSA qui a toujours été d'une grande aide lors des tests effectués à Vélizy. Nahid a été d'une grande écoute, sachant voir l'intérêt de mes travaux dans un contexte industriel sans oublier leur part de recherche académique. Je souhaite remercier Pascal Ligot et Alain Sonet qui ont été mes interlocuteurs privilégiés chez PSA et qui ont facilité mes travaux à Velizy. En dehors des équipes de PSA Peugeot-Citroën, des membres de Mines ParisTech ont également fait partie de la chaire. Tout d'abord Philippe Fuchs, président de la chaire, qui a suivi mes travaux et a toujours cru en leur pertinence. Je remercie également Vincent Weistroffer et Olivier Hugues qui ont été présents lors d'une grande majorité de mes voyages à Vélizy. Vincent a mis en place un scénario complet de tests sur la collaboration homme-robot qui a été une base pour mes travaux de thèse. Olivier m'a été d'une grande aide lorsque quatre mains étaient nécessaires pendant les tests avec les opérateurs. J'en profite également pour remercier tous les volontaires de PSA Peugeot Citroën qui se sont déplacés pour venir « travailler » avec notre robot collaboratif et qui m'ont permis d'avoir une base de données solide.

Ma thèse n'aurait pas été la même sans l'ambiance chaleureuse et les rencontres que j'ai pu faire dans mon laboratoire, le Centre de Robotique de Mines ParisTech (CAOR). Ces trois années ont été riches de rencontres qui m'ont permises de toujours garder le sourire lors de petites périodes de creux. Je garde en mémoire les bons moments passés au laboratoire, les soirées au Pantalon, les voyages au ski, les concours CCC et les repas au ministère qui ont rythmés cette période. Bien sûr, tous ces moments n'auraient pas été les mêmes sans les personnes avec qui je les ai partagés. Une première pensée va à Raoul

qui m'a guidée lors de mes premiers mois et avec qui j'ai eu des conversations ouvertes et très intéressantes sur mes travaux. J'espère que nous aurons toujours à l'avenir l'occasion de discuter sur des sujets qui nous passionnent tous le deux, scientifiques ou non. Ensuite l'équipe « gestes », Edgar et Yannick. Nous avons vu nos thèses «grandir» en même temps, partagé nos doutes, nos résultats, et nos questions et parfois incompréhensions. Une belle équipe qui, j'espère, perdurera ! Sans oublier Alina, toujours de bonne humeur et ouverte à la discussion. J'ai profité de cette période pour participer au Valeo Innovation Challenge, et je suis très heureuse d'avoir partagé cette aventure avec Philip, Florent, Jun et Sofiane, une super équipe ! Au cours de ces trois années j'ai lié des liens forts avec de nombreuses personnes qui, j'espère, perdureront : Vincent, Sylvain, JE, Jef, Etienne, Hassan, Xavier, Marion, Mathieu, Daniele, Arthur, Amaury, Cyril, Axel, Manu, David, Raphaëlle, Thomas, Clémentine, Pierre... J'espère que l'ambiance continuera de rester aussi agréable et bonne enfant au CAOR après mon départ.

En dehors du laboratoire j'ai pu compter sur mes amis et ma famille qui m'ont encouragée pendant ces trois années. Leur soutien et leur intérêt pour mes travaux ont été une force qui m'a permis de continuer et d'avoir confiance en la voie que j'avais choisie.

Résumé

Les robots collaboratifs sont de plus en plus présents dans nos vies quotidiennes. En milieu industriel, ils sont une solution privilégiée pour rendre les chaînes de montage plus flexibles, rentables et diminuer la pénibilité du travail des opérateurs. Pour permettre une collaboration fluide et efficace, les robots doivent être capables de comprendre leur environnement, en particulier les actions humaines.

Dans cette optique, nous avons décidé d'étudier la reconnaissance de gestes techniques afin que le robot puisse se synchroniser avec l'opérateur, adapter son allure et comprendre si quelque chose d'inattendu survient.

Pour cela, nous avons considéré deux cas d'étude, un cas de co-présence et un cas de collaboration, tous les deux inspirés de cas existant sur les chaînes de montage automobiles.

Dans un premier temps, pour le cas de co-présence, nous avons étudié la faisabilité de la reconnaissance des gestes en utilisant des capteurs inertiels. Nos très bons résultats (96% de reconnaissances correctes de gestes isolés avec un opérateur) nous ont encouragés à poursuivre dans cette voie.

Sur le cas de collaboration, nous avons privilégié l'utilisation de capteurs non-intrusifs pour minimiser la gêne des opérateurs, en l'occurrence une caméra de profondeur positionnée avec une vue de dessus pour limiter les possibles occultations.

Nous proposons un algorithme de suivi des mains en calculant les distances géodésiques entre les points du haut du corps et le haut de la tête. Nous concevons également et évaluons un système de reconnaissance de gestes basé sur des Chaînes de Markov Cachées (HMM) discrètes et prenant en entrée les positions des mains. Nous présentons de plus une méthode pour adapter notre système de reconnaissance à un nouvel opérateur et nous utilisons des capteurs inertiels sur les outils pour affiner nos résultats. Nous obtenons le très bon résultat de 90% de reconnaissances correctes en temps réel pour 13 opérateurs.

Finalement, nous formalisons et détaillons une méthodologie complète pour réaliser une reconnaissance de gestes techniques sur les chaînes de montage.

Mots clefs Robotique collaborative, Interaction homme-robot, Reconnaissance de gestes en temps réel, Apprentissage artificiel, Vision par ordinateur

Abstract

Collaborative robots are becoming more and more present in our everyday life. In particular, within the industrial environment, they emerge as one of the preferred solution to make assembly line in factories more flexible, cost-effective and to reduce the hardship of the operators' work. However, to enable a smooth and efficient collaboration, robots should be able to understand their environment and in particular the actions of the humans around them.

With this aim in mind, we decided to study technical gestures recognition. Specifically, we want the robot to be able to synchronize, adapt its speed and understand if something unexpected arises.

We considered two use-cases, one dealing with copresence, the other with collaboration. They are both inspired by existing task on automotive assembly lines.

First, for the co-presence use case, we evaluated the feasibility of technical gestures recognition using inertial sensors. We obtained a very good result (96% of correct recognition with one operator) which encouraged us to follow this idea.

On the collaborative use-case, we decided to focus on non-intrusive sensors to minimize the disturbance for the operators and we chose to use a depth-camera. We filmed the operators with a top view to prevent most of the potential occultations.

We introduce an algorithm that tracks the operator's hands by calculating the geodesic distances between the points of the upper body and the top of the head.

We also design and evaluate an approach based on discrete Hidden Markov Models (HMM) taking the hand positions as an input to recognize technical gestures. We propose a method to adapt our system to new operators and we embedded inertial sensors on tools to refine our results. We obtain the very good result of 90% of correct recognition in real time for 13 operators.

Finally, we formalize and detail a complete methodology to realize technical gestures recognition on assembly lines.

Keywords Collaborative robotics, Human-Robot interaction, Real-time gesture recognition, Machine learning, Computer vision

Chapitre 1

Contexte

Sommaire

1.1 La robotique collaborative 8

1.1.1 Émergence des robots collaboratifs 8

1.1.2 Les enjeux de la robotique collaborative dans l'industrie 8

1.2 La reconnaissance de gestes : nouvelle voie d'interaction 10

1.2.1 Interaction avec des systèmes intelligents 10

1.2.2 Les contraintes du milieu industriel 10

1.3 Organisation du manuscrit 11

Les robots collaboratifs sont de plus en plus présents dans notre quotidien. Certains sont guides dans des musées ou des aéroports, d'autres accueillent des clients dans des magasins ou aident des personnes en perte d'autonomie. En milieu industriel, les opérateurs vont également être amenés à travailler avec des robots collaboratifs. Bien que les interactions avec ces robots restent minimes pour une grande partie de la population, elles vont être amenées à s'accroître dans les prochaines années. L'homme va donc devoir apprendre à communiquer et interagir avec ces robots.

L'étude de l'interaction avec des robots collaboratifs en milieu industriel est un sujet de recherche novateur et en phase avec les enjeux auxquels sont confrontés les entreprises. Il convient, d'une part, d'assurer la sécurité des opérateurs et de permettre aux robots de comprendre leur environnement en les dotant d'intelligence. D'autre part, du point de vue de l'opérateur, il est nécessaire d'évaluer dans quel cadre l'introduction de ces robots peut être acceptée. Ces deux problématiques constituent les deux axes de recherche de la chaire « PSA Peugeot Citroën - Robotique et Réalité Virtuelle » [1] établie entre PSA Peugeot Citroën, Mines ParisTech et Fondation Mines ParisTech.

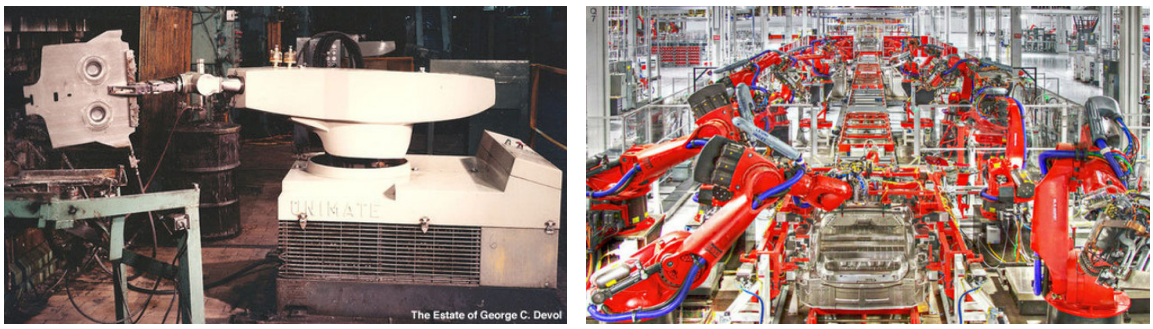
Ce manuscrit présente les résultats d'une thèse inscrite dans le cadre de la chaire. Le sujet de la thèse est la reconnaissance de gestes techniques effectués par un opérateur afin de permettre au robot collaboratif, travaillant avec ou à côté de l'opérateur, de se synchroniser, d'adapter sa vitesse et de comprendre si quelque chose d'anormal survient. Les cas d'étude de la thèse sont tirés de postes d'usine de PSA Peugeot Citroën.

1. http ://chaire-rrv.fr/

1.1 La robotique collaborative

1.1.1 Émergence des robots collaboratifs

L'apparition des premiers robots utiles pour l'homme a eu lieu sur les chaînes de montage où les robots pouvaient effectuer des tâches répétitives, pénibles et dangereuses pour les opérateurs. Le robot Unimate est connu pour être le premier robot industriel programmable, datant de 1954, voir Figure 1.1a. Il s'agit d'un bras articulé pouvant transporter une pièce d'un endroit à un autre. Depuis, les robots industriels ont pris un place de plus en plus importante sur les chaînes de montage 1.1b.



(a) (b)

FIGURE 1.1 – Evolution des robots industriels. (a) : Robot Unimate en 1954, (b) : Usine automobile actuelle

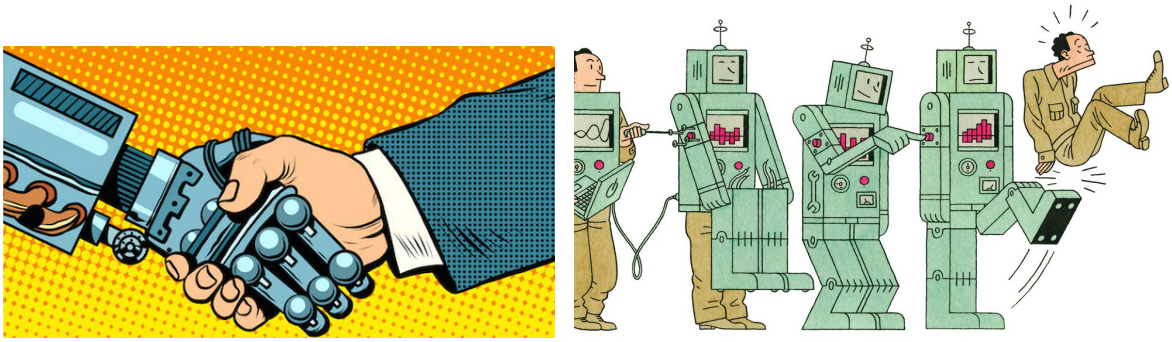
Ces robots sont la plupart du temps dans des zones dédiées où les opérateurs ne peuvent pas se rendre lorsqu'ils sont en fonctionnement. Cette règle permet d'assurer la sécurité des opérateurs mais rend également peu flexible l'organisation d'une chaîne de montage si l'on souhaite par la suite modifier ou changer le produit qui y est fabriqué. De plus, il existe des tâches qui requirent des compétences hybrides entre celles d'un robot (répétitivité, force, précision) et celles d'un opérateur (intelligence, adaptabilité). L'utilisation de robots collaboratifs est donc une solution de plus en plus étudiée par les industriels. Ces robots évoluent dans un espace commun avec l'opérateur, ils peuvent travailler sur une même tâche, en collaboration, ou sur deux tâches distinctes mais en co-présence. Ils illustrent l'usine du futur, plus performante, plus efficace et productive, Figure 1.2a.

1.1.2 Les enjeux de la robotique collaborative dans l'industrie

1.1.2.1 Des robots acceptés

L'étude de l'acceptabilité par les opérateurs constitue le premier axe de recherche de la chaire PSA « PSA Peugeot Citroën - Robotique et Réalité Virtuelle ». Ce fut le sujet de thèse de Vincent Weistroffer [137] soutenue en décembre 2014.

L'acceptation par les opérateurs est primordiale pour créer une collaboration efficace. Pourtant, de nos jours, les opérateurs ont été habitués à travailler dans des zones dépourvues de robots. Les robots industriels étaient alors présentés comme des machines potentiellement dangereuses dont il fallait éviter tout contact. L'introduction de robots collaboratifs va à l'encontre de ce discours. De plus, les opérateurs pourraient avoir l'impression que les robots collaboratifs vont les remplacer dans les usines, voir Figure 1.2b. En réalité, comme expliqué plus haut, l'objectif est de mettre en place une complémentarité entre



(a) (b)

FIGURE 1.2 – Dessins sur la collaboration homme robot. (a) : Illustration sur la collaboration homme-robot évoquant le gain et la réussite issue de cette collaboration efficace, (b) : Dessin satirique sur la peur du remplacement des emplois des opérateurs par les robots (*MIT Technology Review*).

humains et robot, et celle-ci devrait même diminuer la pénibilité du travail des opérateurs et leur permettre de se concentrer sur des tâches à plus grande valeur ajoutée.

Au cours de sa thèse, Vincent Weistroffer a utilisé la réalité virtuelle pour déterminer quels critères d'acceptabilité peuvent y être évalués. Ces critères seront ensuite utilisés pour la configuration de postes de travail collaboratifs afin qu'ils soient mieux acceptés par les opérateurs.

1.1.2.2 Des robots intelligents

Lorsqu'un robot évolue dans un environnement où il peut rencontrer des hommes, plusieurs problèmes se posent : comment assurer la sécurité des personnes autour du robot ? Comment permettre aux personnes et aux robots de communiquer et collaborer ?

Aujourd'hui, pour des applications industrielles, il existe de nombreuses solutions pour assurer la sécurité d'un opérateur qui se trouve dans le même espace et à proximité d'un robot. Ces robots sont eux-mêmes équipés de capteurs de force ou de proximité pour éviter de blesser les opérateurs. Des systèmes à base de lasers ou de caméras sont également utilisés pour détecter si le robot et l'opérateur sont trop proches. De nos jours les robots collaboratifs en milieu industriel sont donc de plus en plus sûrs. Mais cette sécurité n'est pas suffisante pour permettre une collaboration fluide et efficace. En effet, le robot doit être capable de comprendre son environnement afin de s'y adapter. Afin de collaborer, les actions du robot et de l'opérateur doivent être synchronisées. Nous pensons que la reconnaissance de geste pourrait permettre aux robots collaboratifs de comprendre quelle tâche est effectuée par l'opérateur. Avec cette information, le robot pourrait synchroniser ses actions avec celles de l'opérateur, mais également adapter sa vitesse et s'arrêter si quelque chose d'inattendu survient.

Cette problématique constitue le second axe de recherche de la chaire « PSA Peugeot Citroën - Robotique et Réalité Virtuelle », cette thèse porte sur la reconnaissance des gestes techniques effectués par les opérateurs.

1.2 La reconnaissance de gestes : nouvelle voie d'interaction

1.2.1 Interaction avec des systèmes intelligents

Aujourd'hui, l'interaction avec des machines intelligentes devient un domaine de plus en plus étudié. C'est dans cette optique que la reconnaissance de gestes s'est développée afin d'améliorer et de faciliter l'expérience utilisateur, voir Figure 1.3. D'autres domaines ont également aidé à et profité de l'expansion de ce domaine, comme la vidéo surveillance par exemple, ou également lors d'installations artistiques, comme présenté dans l'étude de Bremard et al. [18]. Comme expliqué par Norman [95], bien que les premiers systèmes basés sur la reconnaissance de gestes datent des années 50, les récentes avancées ont permis de s'affranchir de verrous technologiques, réduisant les coûts, augmentant la robustesse, et facilitant l'utilisation de ces nouvelles méthodes d'interaction.

(a) (b)

FIGURE 1.3 – Interaction avec des sytèmes intelligents via des gestes. (a) : Avec un écran dans un bloc opératoire, (b) : Avec un gestionnaire de commandes dans l'habitacle d'une voiture

La reconnaissance de gestes touche à plusieurs domaines comme, entre autres, la modélisation, l'apprentissage artificiel, la vision par ordinateur. Elle désigne l'ensemble des étapes nécessaires à la compréhension d'un geste, allant de la capture à l'aide de capteurs jusqu'à son interprétation.

Dans le cadre de cette thèse, nous allons nous intéresser plus particulièrement à la reconnaissance de gestes techniques effectués par des opérateurs qui travaillent avec des robots collaboratifs. Selon Clark [28], pour pouvoir communiquer les participants doivent avoir des connaissances communes. Pour une communication entre hommes, cela peut-être la langue parlée ou le langage des signes par exemple. Lors d'une collaboration homme-robot, cette connaissance commune n'est pas évidente. La reconnaissance des gestes de l'opérateur permettra d'établir une voie de communication allant de l'opérateur vers le robot.

1.2.2 Les contraintes du milieu industriel

Sur les chaînes de montage, le travail est très codifié. Les opérateurs doivent travailler à une cadence donnée et ont des gestes rapides et précis. Ils ne peuvent prendre du temps pour être certain que le robot, qui travaille avec ou à côté d'eux, a bien compris quelle tâche vient d'être effectuée. Ils n'ont pas non plus le temps d'envoyer des messages, via l'appui sur des boutons par exemple, pour tenir informé le robot de l'avancement de l'exécution des tâches. La collaboration doit donc se faire de la manière la plus naturelle

possible, comme si le robot était un autre opérateur capable de comprendre par lui-même son environnement et s'adapter à son coéquipier.

Dans ce contexte, le système de reconnaissance des gestes doit être d'une part performant pour ne pas interrompre l'opérateur dans son travail, mais également le moins intrusif possible. C'est en cela que se différencie la reconnaissance des gestes en milieu industriel des systèmes conçus pour le grand public. Dans ce dernier cas, là où des erreurs peuvent être tolérées, demandant à l'utilisateur de refaire son geste pour avoir la réaction attendue du système avec lequel il interagit, cela n'est pas possible en milieu industriel.

De plus, il faut éviter d'obliger les opérateurs à porter un équipement supplémentaire, de capteurs intrusifs. Cela pourrait les gêner dans leurs gestes et interférer avec les normes de sécurité sur les chaînes de montage. Nous devons donc privilégier l'utilisation de capteurs non intrusifs.

Pour prendre en compte toute ces contraintes, nous devons donc nous orienter vers un système robuste, basé sur des capteurs non intrusifs pour l'opérateur et qui peuvent être utilisés sur une chaîne de montage.

1.3 Organisation du manuscrit

La thèse se concentre sur la reconnaissance de gestes techniques pour la collaboration homme-robot sur chaîne de montage. Elle s'articule en six chapitres. Tout d'abord, dans le Chapitre 2 nous allons présenter le domaine de la collaboration homme-robot, les particularités de son utilisation en milieu industriel, et les cas d'étude sur lesquels nous avons basé nos travaux. Dans le Chapitre 3 nous allons présenter la reconnaissance de gestes, les différentes méthodes existantes allant de l'extraction d'informations de capteurs jusqu'à la classification des gestes.

Dans le Chapitre 4 nous nous sommes intéressés à une première étude de faisabilité de la reconnaissance de gestes en utilisant un équipement intrusif. Nous présenterons des premiers résultats et les conclusions que nous en retirons.

Dans le Chapitre 5 nous présentons une méthode pour suivre les mains d'une personne filmée en une vue de dessus par une caméra de profondeur. Ces données seront ensuite utilisées, dans le Chapitre 6 pour faire une reconnaissance de gestes avec un équipement non intrusif. Nous présenterons notre méthodologie de reconnaissance de gestes techniques et plusieurs résultats.

Enfin, dans le Chapitre 7, nous détaillerons nos conclusions sur les études menées lors de cette thèse, puis présenterons différentes perspectives.

Chapitre 2

La collaboration homme-robot

Sommaire

2.1 Etat de l'art sur la collaboration homme-robot 13
2.1.1 Les robots collaboratifs 14
2.1.1.1 Les cobots 15
2.1.1.2 Les exosquelettes 15
2.1.1.3 Les robots télé-opérés 16
2.1.1.4 Les robots collaboratifs industriels 16
2.1.1.5 Les robots-assistants de service 18
2.1.2 Le cas particulier de l'industrie 19
2.1.2.1 La sécurité 20
2.1.2.2 Partage du travail 23
2.1.2.3 Interaction avec un robot industriel 24
2.2 Cas d'étude 25
2.2.1 Cas de co-présence 26
2.2.1.1 Présentation du cas d'étude 26
2.2.1.2 Choix des gestes 27
2.2.1.3 Choix des capteurs 28
2.2.2 Cas de collaboration 29
2.2.2.1 Présentation du cas d'étude 29
2.2.2.2 Choix des gestes 30
2.2.2.3 Choix des capteurs 31
2.3 Conclusion 32

Dans ce chapitre, nous allons dans un première partie présenter un état de l'art sur la collaboration homme-robot, appliqué au milieu industriel. Cela nous permettra de mettre en évidence les enjeux qui restent à relever pour rendre ce nouveau type de collaboration sûre et efficace. Dans une seconde partie, nous présenterons deux cas d'étude que nous avons étudié pendant cette thèse, un cas de co-présence et un cas de collaboration.

2.1 Etat de l'art sur la collaboration homme-robot

Les robots quittent peu à peu les cadres qui leur sont dédiés pour faire leur entrée dans nos environnements. Ils ne doivent donc plus être simplement en mode automatique,

mais aussi être capable de s'adapter, communiquer et collaborer avec l'homme. Une collaboration entre un homme et un robot sous-entend que l'homme et le robot travaillent ensemble avec un but commun, qu'ils forment une équipe, Figure 2.1. Contrairement à la collaboration homme-homme, pour la collaboration homme-robot les voies de communications ne sont pas naturelles. Selon la norme ISO-8373 :2012, un robot peut être défini par :

> Mécanisme programmable actionné sur au moins deux axes avec un degré d'autonomie, se déplaçant dans son environnement, pour exécuter des tâches prévues.

Un robot, bien qu'il puisse être mobile, doté d'une certaine autonomie, et pourvu de capacité pour effectuer des tâches prévues, n'a pas (encore) les capacités cognitives d'un humain. De nouveaux codes de collaboration doivent donc être utilisés.

FIGURE 2.1 – Illustration de la collaboration homme-robot

Comme expliqué dans Bauer et al. [7], la collaboration homme-robot peut être réalisée par des actions individuelles qui permettent aux deux partenaires de remplir un objectif commun. Dans la plupart des cas, le robot joue le rôle d'assistant tandis que l'humain le dirige et effectue les actions les plus délicates ou nécessitant le plus de valeur ajoutée.

Dans cette partie nous allons présenter plusieurs types de robots collaboratifs, et expliquer les différents enjeux de la collaboration homme-robot en milieu industriel.

2.1.1 Les robots collaboratifs

Les robots collaboratifs, c'est à dire des robots conçus pour avoir une interaction directe avec l'être humain, peuvent être classés en deux grandes familles :

— Les robots industriels : robots manipulateurs, multi-applications, reprogrammables, commandés automatiquement, programmables sur trois axes ou plus, qui peuvent être fixés sur place ou mobiles, destinés à être utilisé dans des applications d'automatisation industrielle

— Les robots de service : robots qui effectuent des tâches utiles à l'être humain, mais qui ne servent pas à automatiser un processus industriel.

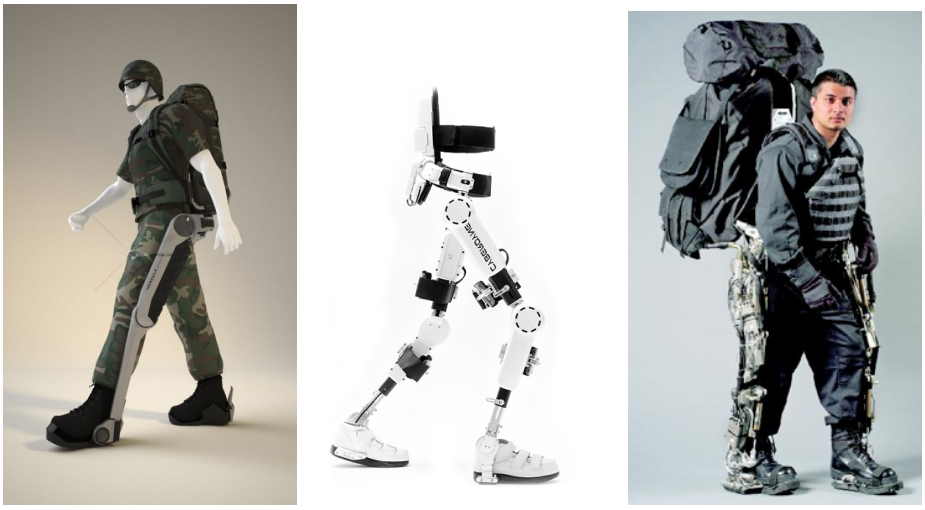
Nous allons détailler ci-dessous plusieurs types de robots collaboratifs, allant des robots avec une application industrielle aux robots dit "compagnons", qui sont utilisés par des particuliers.

2.1.1.1 Les cobots

Les cobots, concaténation des mots *collaborative* et *robot*, ont été introduits en 1996 par Colgate et al. [29]. Ces robots permettent de guider les mouvements d'un opérateur pour manipuler un objet. Ils sont intrinsèquement passifs, c'est à dire qu'ils ne fournissent pas d'énergie à l'opérateur qui travaille avec. Seul l'opérateur fournit de la puissance. Cette caractéristique fait des cobots des robots sécurisés pour la collaboration homme-robot. Une application de cobot chez *General Motors* est présentée par Wannasuphoprasit et al. [134]. Dans ce cas d'étude, un cobot est utilisé pour enlever les portes de voitures sur une chaîne de montage.

2.1.1.2 Les exosquelettes

Les exosquelettes sont des robots qui sont directement portés par l'utilisateur. Ils ont la forme d'un squelette humain et s'adaptent à ses mouvements. Les exosquelettes sont de plus en plus étudiés avec des applications dans différents domaines comme la santé, l'industrie ou le militaire. Des exosquelettes spécifiques aux parties basses du corps ont été utilisés pour augmenter la force musculaire de personnes valides ou pour aider des personnes à mobilité réduite. En 2002, Lee et Sankai [81] présentent un exosquelette pour l'aide à la marche, HAL (*Hybrid Assistive Leg*), développé depuis par Cyberdyne [1], voir Figure 2.2b. Cet exosquelette capte des signaux électriques traduisant l'activité musculaire de l'utilisateur afin de le synchroniser sur les mouvements des jambes.



(a) (b) (c)

FIGURE 2.2 – (a) : Exosquelette Hercule, (b) : Exosquelette HAL, (c) : Exosquelette BLEEX

Banala et al. [5] proposent un exosquelette, appelé ALEX, pour apprendre à nouveau à marcher à des personnes ayant subit un accident vasculaire cérébral. Kazerooni et al [64] ont développé un exosquelette, BLEEX (*Berkeley Lower Extremity Exoskeleton*), permettant de porter des poids importants, voir Figure 2.2c. La Direction Générale de l'Armement à mis au point l'exosquelette Hercule en partenariat avec l'entreprise RB3D [2], le CEA-LIST [3] et l'ESME Sudria [4]. Hercule devrait aider les militaires lors de la manipulation de charges lourdes ainsi qu'augmenter leur endurance, voir Figure 2.2a.

1. http ://www.cyberdyne.jp/english/products/HAL/
2. http ://www.rb3d.com/produits/exosquelettes/
3. Commissariat à l'Energie Atomique - Laboratoire d'Intégration des Systèmes et Technologies
4. Ecole Spéciale de Mécanique et d'Electricité

2.1.1.3 Les robots télé-opérés

Les robots télé-opérés sont en grande partie utilisés pour explorer des milieux qui sont hostiles à l'homme ou difficile d'accès. L'humain peut contrôler ou guider le robot à distance, tandis que celui-ci récolte des informations ou effectue une tâche délicate.

L'entreprise Toshiba a mis au point un robot pour démanteler la centrale nucléaire Fukushima Daiichi. Ce robot à trois bras devra retirer des barres de combustibles dans un des réacteurs endommagés par le tsunami de 2011.

(a) (b) (c)

FIGURE 2.3 – (a) : Robot Da Vinci, (b) : Robot K10, (c) : Robnaute

Les robot télé-opérés sont également utilisés en chirurgie. Ils permettent d'augmenter la précision et la visibilité du chirurgien. Le robot Da Vinci, Figure 2.3a, commercialisé par l'entreprise Intuitive Surgical [5], est déjà présent dans de nombreux hôpitaux. Le chirurgien utilise des télémanipulateurs qui commandent les bras du robot. Ceux-ci reproduisent les gestes à l'identique en minimisant les possibles tremblements. Okamura [99] propose une méthode pour améliorer les retours haptiques de ces robots. Il existe également le robot Neuroarm qui est aussi un robot d'assistance chirurgical spécialisé dans la neurochirurgie.

La NASA a également développé un robot télé-opéré, le robot K10, pour explorer des planètes, voir Figure 2.3b. Fong et al. [40] présentent plusieurs robots télé-opérés développés pour explorer de nouveaux environnements, comme le Robnaute, un robot humanoïde ayant les mêmes capacités de manipulation que les astronautes , voir Figure 2.3c. Il permet de réduire la quantité de consommables à emporter durant les missions spatiales et peut utiliser les outils développés pour les astronautes.

2.1.1.4 Les robots collaboratifs industriels

De nouveaux robots collaboratifs sont développés pour assister les opérateurs sur les chaînes de montage. Ils travaillent dans la même zone que l'opérateur, sur une tâche commune, en collaboration ou en coprésence. Ces robots sont de plus en plus, sûrs, autonomes et robustes.

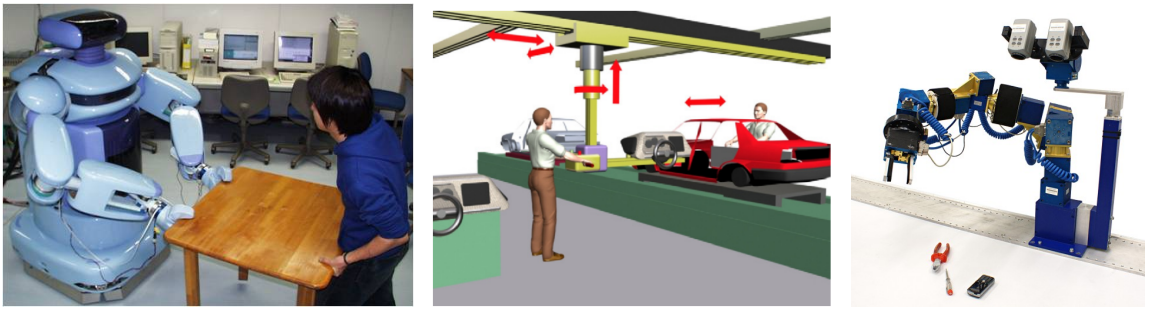
La notion de "robots (ou actionneurs) *compliant*" a été introduite dans les années 2000 par Ham et al. [48], elle permet de décrire des robots plus sûrs et plus adaptés à la collaboration homme-robot en milieu industriel. Contrairement aux robots *stiff* (rigides) qui se déplacent vers une position précise ou suivent un trajectoire pré-définie, les robots *compliant* peuvent dévier de leur position d'équilibre, c'est à dire la position où le robot ne génère aucune force ou aucun couple, en fonction des forces extérieures qui lui sont appliquées. Le but de ces robots est de reproduire des systèmes biologiques en terme de

5. http ://www.intuitivesurgical.com/

capacité à prendre en compte un contact inattendu avec un humain ou l'environnement, permettant une interaction sûre et une exécution des tâches efficace.

Une première catégorie de robots collaboratifs industriels sont les robots co-manipulateurs, nous allons en présenter deux ci-dessous. En 2000, Kosugue et al. [69] ont proposé un robot, appelé *Mr Helper*, Figure 2.4a, permettant de co-manipuler des objets avec des opérateurs. Les auteurs proposent également un algorithme de contrôle pour partager le poids porté par l'opérateur et le robot. L'opérateur commande le mouvement du robot en appliquant des forces à l'objet conjointement porté. Krüger et al. [72] ont également proposé un robot à retour d'effort pour guider et aider à porter des charges lourdes, appelé *Intelligent Power Assist Device*, Figure 2.4b. Il a été utilisé pour le projet européen PISA, [71].

Les robots que nous allons maintenant présenter sont des robots assistants qui permettent d'aider ou d'accompagner des opérateurs sur des chaînes de montage. Ils permettent de faire des tâches à faible valeur ajoutée, tandis que l'opérateur peut se concentrer sur des opérations plus délicates. En 2002, Iossifidis et al. [56] ont présenté le robot CORA, *COoperative Robot Assistant*, un bras articulé d'apparence anthropomorphe, pour aider les opérateurs sur des tâches d'assemblage, Figure 2.4c. Il est doté de nombreuses fonctionnalités telle que la reconnaissance vocale, la détection d'objets ou la reconnaissance de gestes simples, comme "pointer" par exemple. En 2005, Scharft et al. [110] ont proposé un robot stationnaire, *PowerMate*, pour assister un opérateur sur une tâche commune. Il permet de porter des pièces lourdes tandis que l'opérateur travaille dessus. Le contact physique entre l'opérateur et le robot étant possible sur ce type de travail collaboratif, les auteurs ont été particulièrement attentifs aux normes de sécurité lors du développement de ce robot.



(a) (b) (c)

FIGURE 2.4 – (a) : Le robot Mr Helper, (b) : Robot co-manipulateur ©Krüger et al. [72], (c) : Le robot CORA

L'entreprise allemande KUKA [6] a développé le robot collaboratif KUKA LWR, voir Figure 2.5b, présenté par Bischoff et al. [13]. Il s'agit d'un bras à 7 axes permettant de porter des charges de 7 kg. Il a été créé pour travailler avec des opérateurs, peut faire des tâches de manière automatique mais est capable de s'adapter s'il rencontre un obstacle dans sa trajectoire. D'autres robots collaboratifs sous forme de bras sont commercialisés, comme par exemple les bras UR3, UR5 et UR10 d'Universal Robots [7], Figure 2.5a, les robot Roberta de ABB [8], Figure 2.5c, ou les bras CR de Fanuc [9]. Le projet européen *SMErobotis* [10] (2005-

6. http ://www.kuka-robotics.com/france/fr/
7. http ://www.universal-robots.com/fr/
8. http ://www.abb.com/
9. http ://www.fanuc.eu/pt/en
10. http ://www.smerobotics.org/

2009) a utilisé plusieurs de ces bras robotisés pour des cas d'études industriels, comme des assemblages de pièces, du *bin-picking*.

(a) (b) (c)

FIGURE 2.5 – (a) : Les robots UR3 et UR5 et UR10, (b) : Le KUKA LWR, (c) : Le robot Roberta d'ABB

Les robots bi-bras, ou anthropomorphes, font également leur apparition sur les chaînes de montage. Un des plus connus est le robot Baxter de Rethink Robotics [11], Figure 2.6c. il a notamment été utilisé par Fitzgerald [39] sur des cas d'étude industriels occupant des tâches à faible valeur ajoutée : ranger des pièces dans des cartons ou emballer des lots de gobelets par exemple. L'entreprise japonaise Kawada [12] propose également des robots bi-bras anthropomorphes, dont le robot Nextage, Figure 2.6b. Il a été développé pour l'assemblage de pièces, ou *picking*, une tâche répétitive pour les opérateurs. Ce robot travaille principalement en co-présence en usine, mais pas en collaboration directe. En février 2016, le laboratoire JRL (Joint Robotics Laboratory) du CNRS et Airbus Group ont lancé un programme de recherche sur 4 ans [13] visant à développer des robots humanoïdes pouvant évoluer dans des usines. Le but du programme est de permettre à ces robots de se déplacer dans des environnements exigus que l'on peut rencontrer sur les lignes d'assemblage aéronautique. Ce sont des robots Kawada qui seront utilisés, les modèles HRP-2 et HRP-4, Figure 2.6a.

2.1.1.5 Les robots-assistants de service

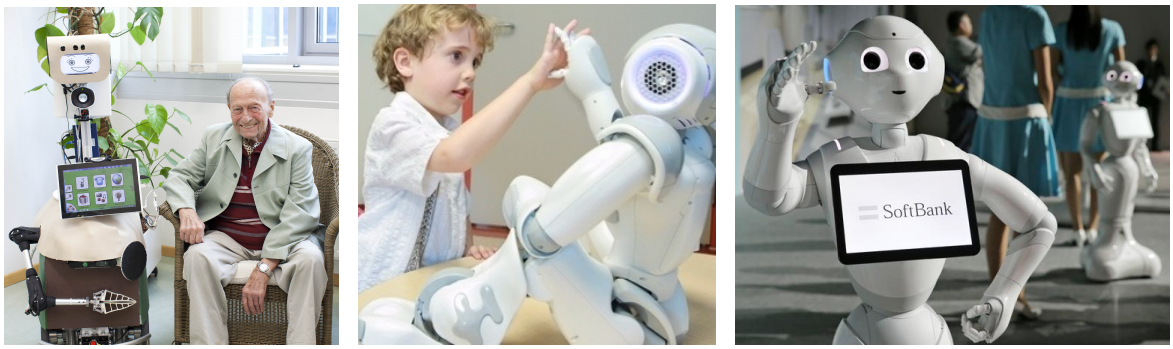
Les robots assistants de service ont dans un premier temps été utilisés pour aider les personnes âgées et les personnes handicapées qui avaient besoin d'une assistance. Chandrasekaran et al. [24] donnent une liste non exhaustive de ces robots. Vincze et al. [129] présentent le robot HOBBIT qui aide les personnes âgées à pouvoir rester plus longtemps chez elles avant d'aller dans des instituts adaptés à leur manque d'autonomie. Dans le même esprit, le robot Care-O-bot, présenté par Hans et al. [49] donne son assistance aux personnes âgées et handicapées. Il peut apporter des objets, aider à la mobilité et interagir via un écran. Il a aussi été utilisé pour faire guide dans des musées. Le projet Juliette [14] a pour but de permettre à des robots existants, NAO et ROMEO de l'entreprise SoftBank Robotics Europe [15], anciennement Aldebaran, de comprendre son environnement afin

11. http ://www.rethinkrobotics.com/baxter/
12. http ://global.kawada.jp/
13. http ://www2.cnrs.fr/presse/communique/4413.htm
14. http ://projetjuliette.com/Juliette/
15. https ://www.ald.softbankrobotics.com/fr

(a) (b) (c)

FIGURE 2.6 – (a) : HRP-4 de Kawada, (b) : Nextage de Kawada, (c) : Nexter de Rethink Robotics

d'être d'une meilleure aide. Ils pourront interagir via des commandes gestuelles, comprendre lorsqu'une personne a chuté et plus généralement comprendre l'activité courante de l'utilisateur. Cette même entreprise a également développé le robot Pepper qui sert de guide ou renseigne des clients dans des magasins.



(a) (b) (c)

FIGURE 2.7 – Exemples de robots-assitants de service (a) : le robot HOBBIT, (b) : le robot NAO, (c) : le robot Pepper

Les robots ont également été utilisés pour interagir avec des enfants autistes afin de renforcer leur comportement social. La société Aldebaran a mis en place le programme ASK *Autism Solution for Kids* et utilisent le robot NAO pour travailler avec ces enfants. Le programme a été étendu à l'apprentissage pour tous les enfants, toujours avec le robot NAO.

2.1.2 Le cas particulier de l'industrie

En milieu industriel, l'utilisation de robots collaboratifs peut apporter des gains financiers et une plus grande flexibilité dans les usines, comme l'a présenté Hägele et al. [47] en 2002, Figure 2.8. Dans cette étude faite dans le cadre du projet MORPHEA, les auteurs comparent les lignes de montages manuelles avec des lignes de montage automatisées. Les cellules de travail hybrides semblaient dès alors améliorer la productivité de la chaîne de montage par rapport à des cellules de travail manuel.

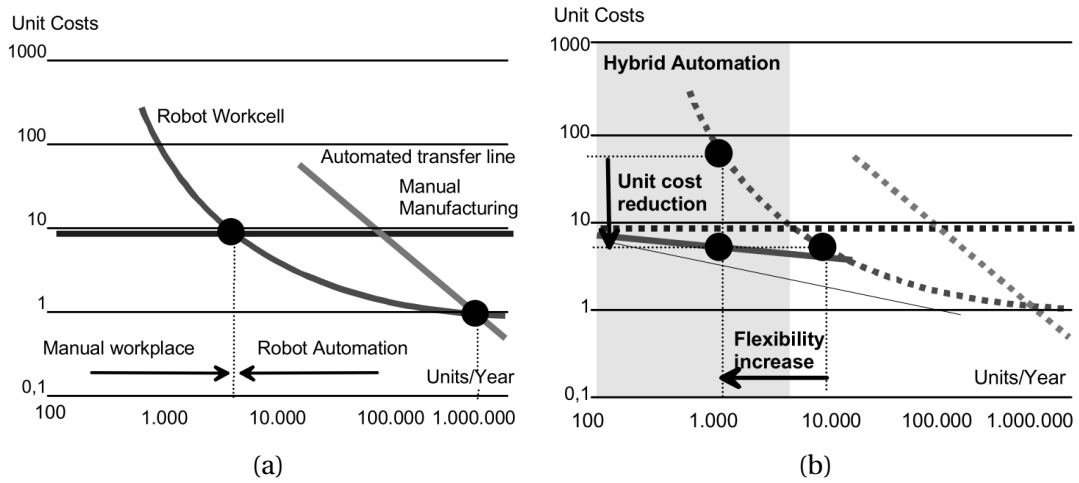


FIGURE 2.8 – Gains financiers et en flexibilité grâce à l'arrivée de cellules hybrides sur les chaînes de montage. Comparaison entre des postes manuels 2.8a et des postes automatisés 2.8b. ©Hägele et al. [47]

Des programmes ont depuis été mis en place pour permettre notamment aux PME françaises d'être accompagnées dans l'automatisation de leur production, nous pouvons citer Robot Start PME [16]. L'automatisation des usines fait également partie du plan "Usine du Futur" présenté dans le programme "Nouvelle France industrielle" mis en place par le gouvernement français en septembre 2013 puis repris sous le nom de "Industrie du futur" en mai 2015 [17].

De nouvelles problématiques se posent pour rendre ce nouveau type de collaboration efficace. Nous allons donc nous intéresser aux enjeux liés à la sécurité des opérateurs, au partage du travail et à la gestion de l'interaction avec le robot.

2.1.2.1 La sécurité

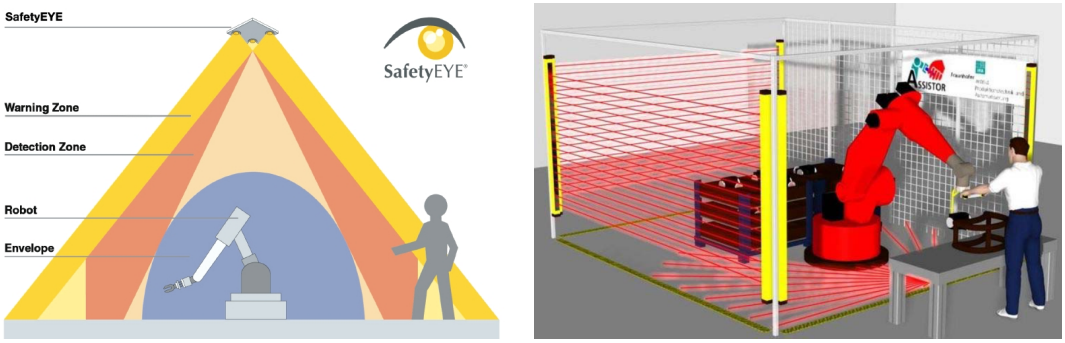
En milieu industriel, la notion de sécurité est régie par des normes et des certifications. Pour la collaboration homme-robot, c'est la norme ISO-10218 éditée en 2011, qui délimite les conditions de sécurité. Elle préconise plusieurs contraintes sur le comportement du robot collaboratif. Sa vitesse ne doit pas excéder les 250 mm/s, et sa puissance doit être inférieure à 80 W, ou sa force à 150 N. Les robots *compliant* permettent de respecter cette contrainte. La norme ISO-13855 ajoute des contraintes sur les zones de sécurité dans l'espace de la collaboration entre l'homme et le robot. Depuis début 2016, la nouvelle norme ISO-15066 complète la norme ISO-10218, elle détaille plusieurs modes de collaboration et ajoute une "échelle de blessure" ou *injury level*. Pour chaque partie du corps exposée à un robot, les industriels disposent d'un seuil maximal de force ou pression qui peut y être appliqué.

Il existe des dispositifs pour garantir la sécurité des opérateurs à proximité des robots collaboratifs. On peut les classer, comme l'a fait Krüger et al. [73], en deux familles : les dispositifs pré-collision et les dispositifs post-collision.

Dispositifs pré-collision En 2007, Kulic et Croft [75] ont proposé une revue sur les stratégies de sécurité pré-collision pour l'interaction homme robot. Deux éléments clefs sont

16. http ://www.robotstartpme.fr/
17. http ://www.economie.gouv.fr/lancement-seconde-phase-nouvelle-france-industrielle

détaillés pour garantir la sécurité de l'opérateur : planifier la trajectoire du robot et permettre au robot de percevoir son environnement. Pour rendre la collaboration plus flexible et adaptative, les stratégies de sécurité se sont surtout intéressées à la compréhension de l'environnement du robot. Un dispositif simple à mettre en place est l'installation de barrières lasers, Figure 2.9b, à la place de barrières physiques, permettant à l'opérateur et au robot de travailler dans le même espace. Lorsque l'opérateur franchit cette barrière pour rentrer dans la zone du travail du robot, celui-ci s'arrête automatiquement s'il est en fonctionnement. L'entreprise SICK[18] propose plusieurs capteurs multi-faisceaux de ce type. Le capteur SafetyEye de Pilz[19] permet de créer plusieurs enveloppes virtuelles 3D autour du robot, Figure 2.9a, chacune étant associée à un niveau de sécurité croissant lorsqu'on se rapproche du robot. En fonction de la présence d'un opérateur dans une enveloppe, le comportement du robot est modifié, allant d'un ralentissement de sa vitesse jusqu'à un arrêt complet.



(a) (b)

FIGURE 2.9 – (a) : Le Safety Eye, (b) : Barrières lasers ©Schraft et al. [110]

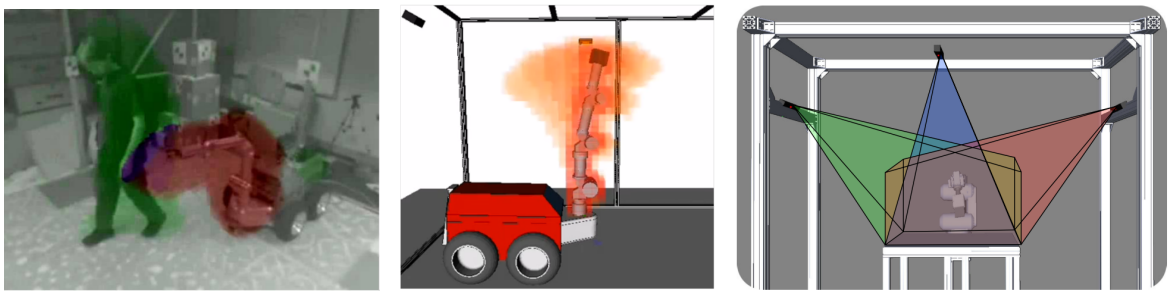
En 2005, Krüger et al. [74] ont proposé un système de vision stéréoscopique basé sur trois caméras pour suivre les mouvements de l'opérateur et du robot. En déterminant les coordonnées 3D des deux partenaires, ce système permet d'éviter les collisions en réduisant la vitesse du robot ou en l'arrêtant.

Les caméras de profondeur ont également été utilisées pour garantir la sécurité des opérateurs. Fischer et Henrich [38] ont fusionné les données de plusieurs caméras 3D pour déterminer les zones occupées dans la zone de travail partagée par le robot et l'opérateur. Ils mettent en place un algorithme de détection de collisions. Un seuil de distance entre le robot et les autres objets de la zone a été établi pour permettre de moduler la vitesse du robot.

En 2012, les auteurs Rybski et al. [109] se sont intéressés à la fusion de données pour assurer la sécurité dans une cellule de travail où un opérateur et un robot coexistent, Figure 2.10a. Pour cela, ils utilisent plusieurs caméras 3D. Connaissant la position du robot et en fusionnant les données des caméras pour gérer les occultations, ils déterminent en temps réel lorsqu'une personne s'approche trop près du robot. Ils utilisent deux types de zones autour du robot : une zone de "sécurité" et une zone de "danger". Ces zones évoluent en fonction du mouvement du robot et de sa vitesse. L'entrée d'une personne dans une des zones entraîne un ralentissement ou un arrêt du robot.

Anderson-Sprecher et al. [4] calculent une grille d'accessibilité, *reachability grid*, composée de voxels, c'est à dire un équivalent tri-dimensionnels des pixels, autour du robot,

18. https ://www.sick.com/fr/fr/
19. https ://www.pilz.com/fr-FR



(a) (b) (c)

FIGURE 2.10 – Méthodes d'évitement de collision (a) : En rouge, zone de danger, en vert, zone sûre (©Rybski et al.[109]), (b) : En teintes de rouges, les zones accessibles par le robot (©Anderson-Sprecher et al.[4], (c) : Fusion des caméra 3D (©Lenz et al. [82])

Figure 2.10b. A chaque voxel est associé un temps minimum dont a besoin le robot pour l'atteindre. Cette étude prend en compte la vitesse du robot et son nombre de degrés de liberté. Le calcul se fait en temps réel et donne une information précise sur les limites de mouvements d'un robot pour garantir la sécurité des personnes autour de lui.

Dans le cadre du projet JAHIR, Lenz et al. [82] ont travaillé sur une fusion de données de caméras 3D pour comprendre l'environnement du robot, , Figure 2.10c. Les objets inconnus dans la scène sont labellisés en tant qu'objets statiques ou mobiles.

Dispositifs post-collision Lorsque les interactions entre l'opérateur et le robot sont rapprochées, les systèmes basés sur la vision ne sont pas toujours suffisants. Les nouveaux robots collaboratifs sont *compliant* et ont en plus des capteurs intégrés qui permettent de les rendre intrinsèquement sûrs.

(a) (b)

FIGURE 2.11 – Sécurité des robots assistants insutriels (a) : Système de coque MRK), (b) : Crash test sur un mannequin avec le robot LWR de KUKA,

Les robots KUKA LWR ont des capteurs de couple sur chacun des sept axes, comme expliqué par Bischoff et al. [13]. Cela leur permet de repérer toutes les collisions. Le robot Netxer de Rethink Robotics a des capteurs d'effort à chaque articulation et des *"Series Elastic Actuators"* qui minimisent les forces lors des impacts. La société allemande MRK-Systeme [20] a développé des coques pour rendre des robots non sécurisés capables de s'arrêter en cas de contact. Ce système, appelé *Safe Interaction* est composé d'une mousse, à

20. http ://www.mrk-systeme.de/

appliquer autour du robot, qui comporte des capteurs tactiles et capacitifs permettant de détecter des présences jusqu'à 30cm.

En 2007, Oberer et Scharft [97] et Haddadin et al. [46] ont fait des études pour évaluer les risques de blessures lors d'un choc avec un robot collaboratif. Ils ont appliqué les tests utilisés pour les crashs de voitures. Ils utilisent notamment l'indice HIC, *Head Injuriy Criterion*, qui évalue l'accélération de la tête lors d'un impact. On peut observer Figure 2.12 les résultats, par Haddadin et al. [46], de HIC en fonction de la vitesse du robot. Pour des vitesses allant jusqu'à 2 m/s, le niveau de HIC, évalué pour plusieurs robots, atteint au maximum la valeur de 70 sur une échelle allant de à 0 plus de 1000, ce qui équivaut à de très faibles blessures. Par comparaison, la norme ISO-10218 préconise des vitesse en dessous de 0.25m/s.

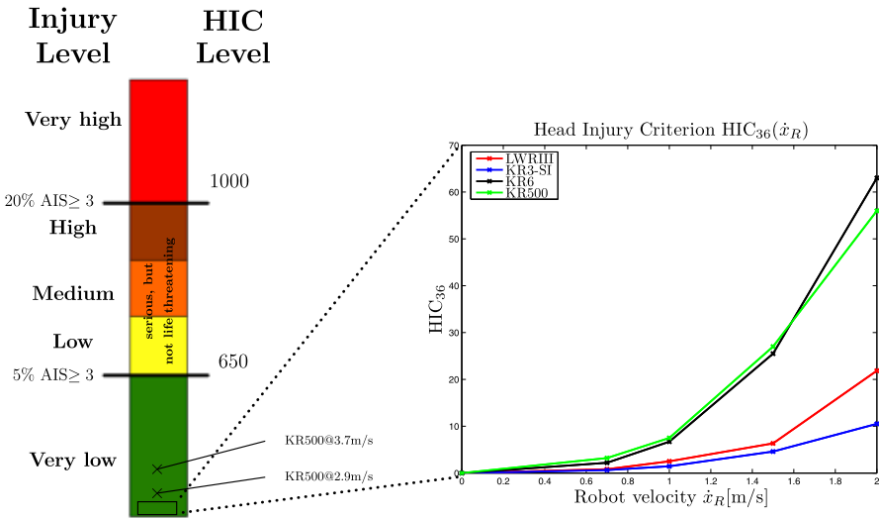


FIGURE 2.12 – Évaluation des risques de blessures liées à la collaboration avec un robot. Différents niveaux de HIC en fonction de la vitesse du robot (©Haddadin et al.[46])

Oberer et Scharft [97] insistent sur le fait que les critères utilisés pour déterminer les blessures lors de chocs en voiture ne sont pas tous applicables aux tests de collisions avec un robot. Il en ressort néanmoins qu'en suivant les directives des normes associées à la collaboration homme-robot, les impacts représentent très peu de danger pour les opérateurs.

2.1.2.2 Partage du travail

Dans les espaces de travail partagés en milieu industriel, il peut y avoir deux types de division de travail. L'opérateur et le robot peuvent travailler en réelle collaboration, c'est à dire travailler sur la même tâche, ou juste en co-présence, c'est à dire travailler sur deux tâches distinctes mais dans le même espace.

Depuis la disparition progressive des barrières matérielles pour laisser place aux barrière immatérielles, les postes de co-présence sont devenus de plus en plus nombreux sur les chaînes de montage. En 2012, Shi et al. [116] décrivent trois niveaux de co-présence homme-robot dans les usines automobiles du futur. Le niveau le plus bas ne permet pas de contact entre le robot et l'opérateur ; leurs zones de travail sont distinctes. Même si il n'y a pas de barrières entre eux, l'opérateur n'a pas à entrer en contact avec le robot ou avec des pièces que le robot pourrait porter. Au niveau intermédiaire, le robot est en mode automatique, l'opérateur et le robot travaillent chacun dans une zone spécifique.

Si l'opérateur entre dans la zone réservée au robot, sa vitesse est ralentie jusqu'à un possible arrêt. Pour le niveau de plus haute collaboration, le robot est en mode automatique et se synchronise, vitesse et mouvements, sur l'opérateur afin de travailler sur une même tâche. C'est dans ce dernier mode que le robot et l'opérateur sont réellement des partenaires de travail.

Avec les nouvelles méthodes mises en places pour garantir la sécurité des opérateurs, les tâches collaboratives se sont multipliées. Le contact physique entre l'homme et le robot est la plupart du temps nécessaire, mais les avancées technologiques ont permis de rendre ces postes sûrs et donc de les développer. Des études ont été menées pour étudier leur faisabilité de ces nouvelles tâches sur les chaînes de montage. En 2012, Corrales et al. [31] ont présentés trois cas de collaboration homme-robot pour le montage et le démontage de pièces. Dans le cadre du projet *ICARO*, Cherubini et al. [26] ont travaillé sur un système collaboratif pour le montage d'un joint de transmission, source de troubles musculo-squelettiques au niveau du poignet. Dans ce cas d'étude, le robot oriente la pièce pour permettre à l'opérateur de travailler plus facilement dessus. Le robot et l'opérateur sont en contact permanent et le robot doit synchroniser ses actions en fonction de l'avancement de la tâche effectuée par l'opérateur.

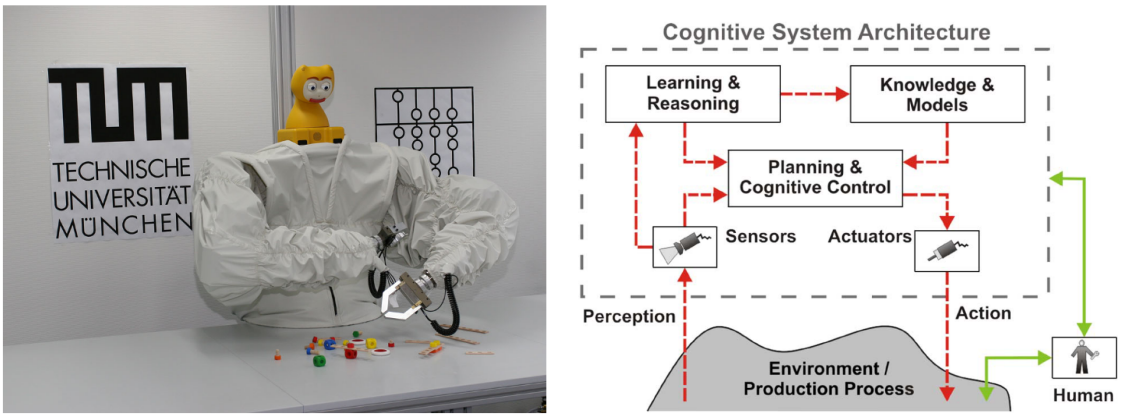
2.1.2.3 Interaction avec un robot industriel

La collaboration entre un opérateur et un robot demande une adaptation de l'opérateur à son nouveau collègue. En effet, alors que la collaboration entre deux opérateurs est simple car la communication entre eux est naturelle, la collaboration entre un homme et un robot demande à l'opérateur d'utiliser de nouveaux codes de communication.

Une étude psychologique de Knoblich et Jordan [67] met en avant la nécessité, lors d'un travail en groupe, d'anticiper sur les actions de ses partenaires pour avoir une collaboration efficace.

Pour contourner le problème, les robots ont longtemps été programmés sur des modes automatiques qui ne permettent pas une synchronisation des tâches avec l'opérateur. Plus récemment, des études se sont intéressées aux actions jointes entre un opérateur et un robot. Hoffman et Breazeal [53] se sont intéressés à l'effet de l'anticipation des tâches sur l'efficacité et la fluidité d'une collaboration homme-robot. Ils comparent un mode où le robot anticipe sur ses actions en fonction de ce qu'il perçoit chez l'opérateur, modélisés par un processus de Markov, avec un mode où le robot enchaîne ses actions automatiquement. Il en ressort que les opérateurs ont préféré travailler avec le robot qui anticipait sur les actions et le travail leur a semblé plus fluide. En 2005, Schrempf et al. [111] ont proposé une méthode pour synchroniser les actions d'un robot en déterminant les intentions de l'opérateur avec des réseaux bayésiens dynamiques. Rickert et al. [108] ont présenté un robot collaboratif, voir Figure 2.13a prenant en compte des informations provenant de l'opérateur, grâce notamment à une reconnaissance des paroles, des objets ainsi qu'un suivi des mains et de la tête, et des informations sur l'état d'avancement de la tache collaborative pour déclencher la prochaine action du robot.

Travailler avec un robot collaboratif est une nouvelle expérience pour les opérateurs, et celle-ci peut parfois être appréhendée. Afin de pouvoir mieux évaluer la réaction d'un opérateur face à un robot collaboratif sur un nouveau poste, la réalité virtuelle a été utilisée. Grâce à cet outil, un poste collaboratif peut être testé en virtuel pour mieux apprécier le comportement d'un opérateur face à un robot. Weistroffer et al. [138] ont travaillé sur ces problématiques pour évaluer l'acceptabilité d'un opérateur à travailler avec un robot en co-présence.



(a) (b)

FIGURE 2.13 – (a) : Robot utilisé par ©Rickert et al. [108]), (b) : Structure d'une "*Cognitive Factory*" (©Bannat et al. [6]),

L'équipe RIS, Robotique et InteractionS, du laboratoire LAAS-CNRS[21] de Toulouse s'est largement intéressée à la problématique d'intention jointe dans la collaboration entre un homme et un robot, surtout appliquée aux robots de service, notamment dans l'étude de Alami et al. [3].

Au sein du projet *ICARO*, Cherubini et al. [27] ont également travaillé sur un projet collaboratif où un robot et un opérateur travaillaient ensemble sur des tâches de vissage. Le robot devait s'adapter au comportement de l'opérateur avec lequel il collaborait.

Dans le cadre du projet *JAHIR*, Lenz et al. [83] ont également travaillé sur l'élaboration d'une cellule de travail collaborative. Ils utilisent un robot Mitsubishi et équipent la cellule de plusieurs capteurs pour permettre au robot de comprendre son environnement. Ils travaillent sur la reconnaissance de la parole, la reconnaissance de gestes simples, comme "pointer", la reconnaissance et le suivi d'objets. L'idée d'équiper les usines de capteurs pour doter les robots d'intelligence a été reprise en 2011 par Bannat et al. [6] en introduisant le terme de "*Cognitive Factories*". C'est à dire des environnements industriels où les machines sont équipés de capacité cognitives qui leur permet de devenir plus autonomes. Pour cela, les "*Cognitive Factory*" doivent être pourvues avec des capacités en perception, en apprentissage et en prise de décision, voir Figure 2.13b.

Afin de créer une nouvelle voie de communication et pour que le robot comprenne son environnement, nous nous sommes intéressés à la reconnaissance des gestes de l'opérateur lorsqu'il effectue son travail. Grâce à ces informations, le robot pourrait se synchroniser sur l'opérateur, adapter son allure et comprendre lorsque quelque chose d'inattendu survient. Nous allons ci-dessous détailler deux cas d'étude de collaboration et co-présence homme-robot en milieu industriel auxquels nous nous sommes intéressés dans le cadre de cette thèse. Nous souhaitons y ajouter un système de reconnaissance de gestes pour fluidifier la collaboration de l'opérateur et du robot.

2.2 Cas d'étude

Dans cette partie, nous allons présenter deux cas d'étude : un cas de co-présence, l'opérateur et le robot travaillent dans le même espace mais pas sur les même tâches, et un cas de collaboration, l'opérateur et le robot travaillent ensemble sur une tâche commune.

21. https ://www.laas.fr

FIGURE 2.14 – Cas d'étude de co-présence

Nous allons détailler leur fonctionnement, les gestes que nous souhaitons reconnaître et les capteurs que nous allons utiliser à cette fin.

2.2.1 Cas de co-présence

2.2.1.1 Présentation du cas d'étude

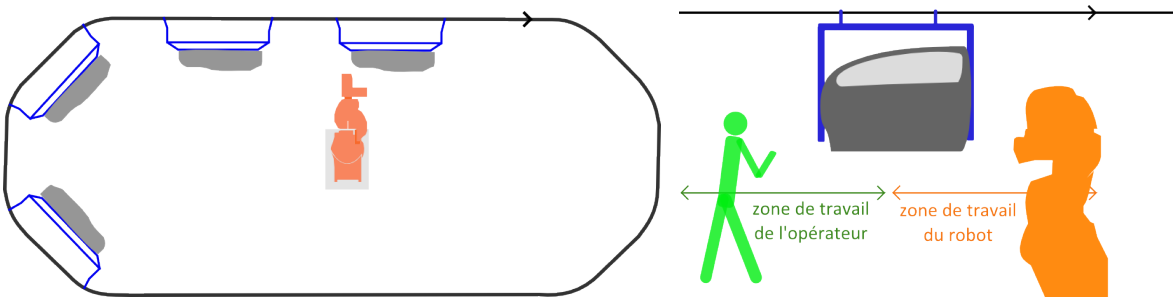
Le robot et l'opérateur partagent une même zone de travail, sans barrière physique pour les séparer. Ils peuvent entrer en contact lorsqu'ils sont proches l'un de l'autre, mais ce n'est pas nécessaire pour la réalisation de leur tâches individuelles.

Le cas d'étude a été imaginé par des roboticiens et des ergonomes de PSA Peugeot Citroën. Il s'inspire d'un poste de travail existant sur les chaînes de montage. Sur ce poste, l'opérateur doit apposer des obturateurs, des clips et une feuille d'étanchéité sur une porte de voiture. Le robot applique ensuite une roulette sur le contour de la feuille d'étanchéité pour la fixer solidement à la porte, Figure 2.14. L'opérateur et le robot ont chacun un pas de travail spatial qui leur est propre, voir Figure 2.15b, mais qui peuvent se superposer. La porte est sur une balancelle mobile qui avance à une vitesse constante, passant devant l'opérateur puis devant le robot.

Le passage de la roulette sur les feuilles d'étanchéité est une tâche source éventuellement de trouble musculosquelettiques (TMS) pour les opérateurs. Utiliser un robot collaboratif industriel pour cette tâche permet de conserver la structure préexistante de la chaîne de montage et de diminuer des blessures chez les opérateurs. Ils pourront alors se concentrer sur des tâches avec une valeur ajoutée.

Le robot utilisé dans ce cas d'étude est un robot collaboratif de la marque KUKA, le modèle KR5, équipé de la coque MRK. Elle est dotée de capteurs capacitifs et tactiles et elle permet de rendre le robot sécurisé. Grâce aux capteurs capacitifs, le robot peut détecter la présence d'un opérateur autour de lui à une distance inférieure à 30 cm. Les capteurs tactiles lui permettent de détecter les collisions. Lorsque l'opérateur est à moins de 30 cm du robot, celui-ci s'arrête puis s'écarte de la porte. Les balancelles continuent d'avancer. Il reprend son travail une fois que l'opérateur a libéré sa zone de travail. Lors d'une collision forte, l'arrêt d'urgence est activé, stoppant le robot et les balancelles.

Dans la cellule expérimentale de PSA, pour des raisons pratiques, quatre balancelles tournent en boucle, 2.15a. L'opérateur et le robot travaillent sur la porte qui défile devant eux 2.15b.



(a) (b)

FIGURE 2.15 – Cellule expérimentale pour l'étude de la co-presence (a) : cellule vue de haut, (b) : cellule vue de face avec les zones du travail de l'opérateur et du robot

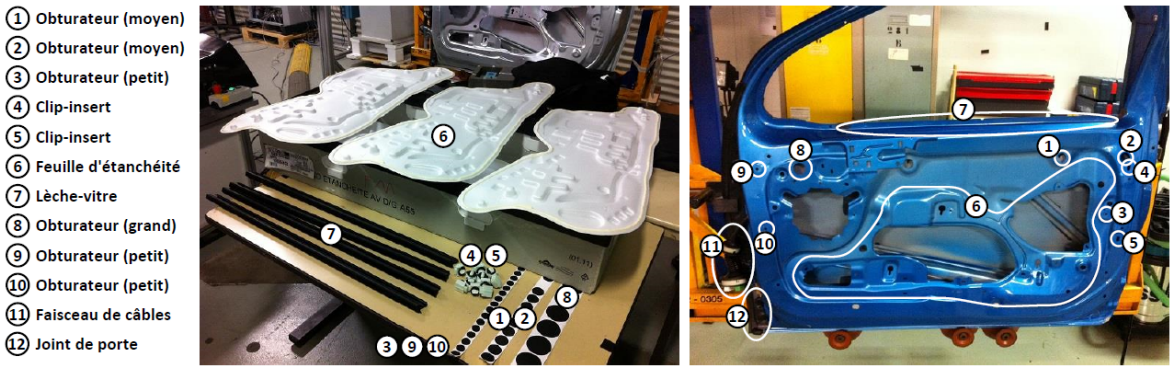


FIGURE 2.16 – Objets à appliquer sur la porte de voiture. Image issue de la thèse de Vincent Weistroffer [137]

2.2.1.2 Choix des gestes

Sur ce poste de co-présence, l'opérateur doit effectuer plusieurs actions :

- enlever l'adhésif de la feuille d'étanchéité,
- poser la feuille d'étanchéité sur la porte,
- pré-coller la feuille d'étanchéité sur la porte (petites pressions sur les bords de la feuille pour qu'elle reste collée sur la porte jusqu'à ce que le robot passe sa roulette),
- poser des clips sur la porte,
- poser des obturateurs sur la porte,
- mettre le lèche vitre,
- sortir les faisceaux de câbles de la porte.

Ces actions ne sont pas soumises à un ordre particulier. Néanmoins, les opérateurs commencent par les actions sur la droite de la porte pour finir par celles sur la gauche afin de suivre le mouvement de la porte sur la balancelle. Une liste des objets à apposer sur la porte avec leur emplacement est représentée Figure 2.16. Nous voulons pouvoir synchroniser le robot avec l'opérateur afin qu'il commence à appliquer la roulette pour fixer durablement la feuille d'étanchéité lorsque celle-ci a déjà été apposée sur la porte par l'opérateur.

Reconnaître toutes les actions de l'opérateur n'est pas nécessaire pour permettre cette synchronisation. Nous souhaitons plutôt privilégier la reconnaissance d'actions relatives à la pose de la feuille d'étanchéité. De plus, lorsque l'opérateur pose le lécheur sur la porte, il se place devant la feuille d'étanchéité. Nous souhaiterions donc que cette action soit

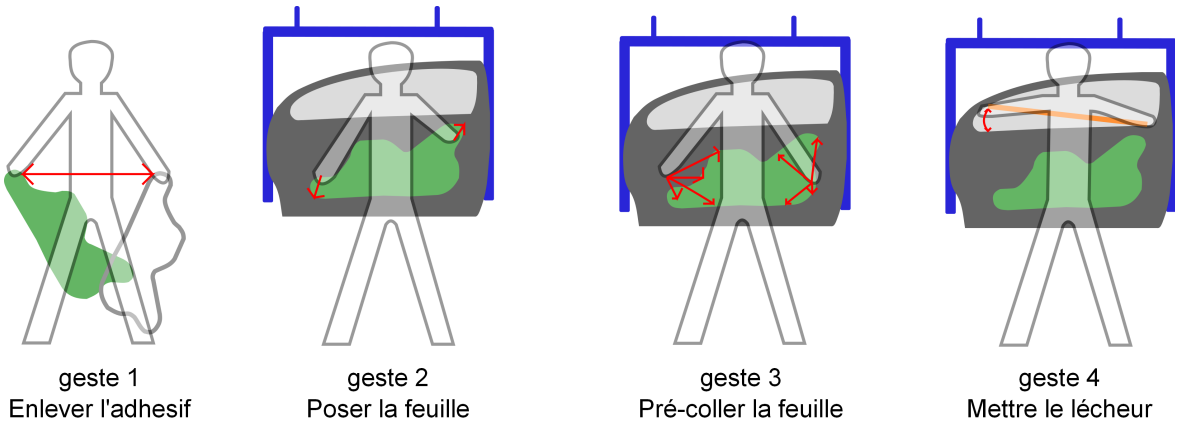


FIGURE 2.17 – Schémas des quatre gestes à reconnaître dans le cas de la co-présence. Geste 1 : enlever l'adhésif de la feuille d'étanchéité ; Geste 2 : poser la feuille sur la portière ; Geste 3 : pré-coller la feuille sur la porte ; Geste 4 : mettre le lécheur sur la portière.

également reconnue. Ainsi, le robot peut commencer son cycle lorsque le lécheur est posé pour éviter que l'opérateur et le robot ne soient gênés. Connaître l'évolution des actions relatives à la feuille d'étanchéité permet de déclencher le début du cycle du robot au bon moment. Nous avons donc sélectionné quatre gestes à reconnaître :

geste 1 : enlever l'adhésif de la feuille d'étanchéité

geste 2 : poser la feuille d'étanchéité sur la porte

geste 3 : pré-coller la feuille d'étanchéité sur la porte

geste 4 : mettre le lécheur

La Figure 2.17 illustre ces quatre gestes.

2.2.1.3 Choix des capteurs

Nous souhaitons, dans ce premier cas d'étude, vérifier la faisabilité de la reconnaissance de gestes techniques. Nous nous sommes donc intéressés aux capteurs inertiels qui donnent des mesures fiables nécessitant peu de traitement. Un des autres avantages de ces capteurs est qu'ils ne sont pas soumis aux occultations. Nous disposons de deux types de capteurs inertiels, Figure 2.18, les capteurs *MotionPod* de MOVEA, depuis racheté par l'entreprise InvenSense [22] , et une veste Animazoo de l'entreprise Synertial [23].

Un *MotionPod*, Figure 2.18b, donne ses rotations en temps réel à une fréquence pouvant aller jusqu'à 200 Hz et a une portée allant jusqu'à 30 m. Nous avons utilisé deux *MotionPods*, un sur chaque poignet de l'opérateur.

La veste Animazoo , Figure 2.18a, est composée de 11 capteurs inertiels positionnés aux articulations du haut du corps l'opérateur. Chacun des capteurs donne ses rotations en temps réel.

Nous ne comptons pas utiliser les deux types de capteurs simultanément. Mais nous allons pouvoir comparer leur performance dans la reconnaissances des gestes techniques.

22. http ://www.invensense.com/
23. http ://www.synertial.com/

(a) (b)

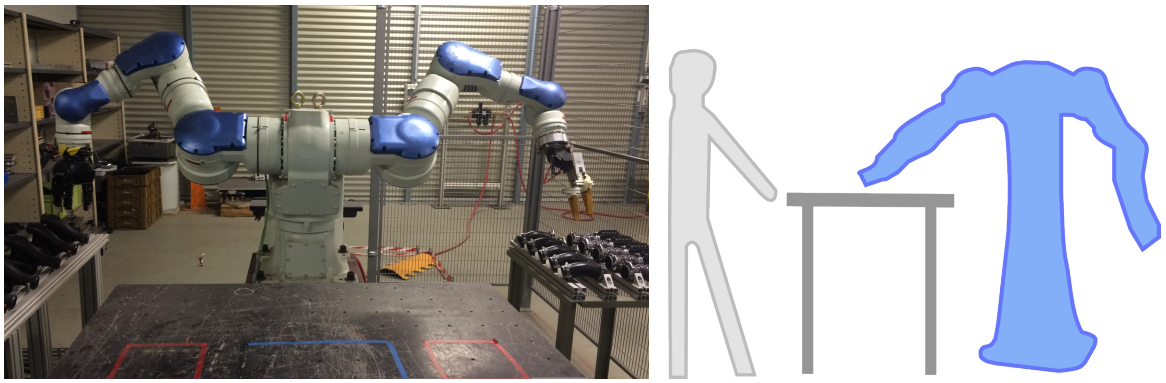
FIGURE 2.18 – Capteurs inertiels, (a) : vêtement Animazoo, (b) : *MotionPods*

2.2.2 Cas de collaboration

2.2.2.1 Présentation du cas d'étude

Dans ce cas d'étude, nous nous intéressons à un scénario de collaboration entre un robot et un opérateur. Ils travaillent alors simultanément sur une tâche commune dans le même espace et sont amenés à entrer en contact lors de l'exécution de leurs tâches.

Pour les scénarios de co-présence, le robot peut effectuer une tâche autrefois attribuée à un opérateur. Ces scénarios peuvent facilement être mise en place sur les chaînes de montage actuelles. Les scénarios collaboratifs demandent quant à eux la mise en place d'une nouvelle organisation dans la distribution des tâches.



(a) (b)

FIGURE 2.19 – Cellule expérimentale pour l'étude de la collaboration (a) : Robot Motoman SDA20 , (b) : l'opérateur et le robot sont séparés par une table

Pour l'élaboration de ce poste collaboratif, nous nous sommes intéressés à une tâche minimisant la réorganisation d'une chaîne de montage et pour laquelle la collaboration avec un robot peut être bénéfique. Les postes de préparation manuels, aussi appelés *kitting*, dans les industries automobiles semblent être de bons candidats. Ils ne sont pas directement sur les chaînes de montage mais se situent en bout de ligne et sont destinés à la préparation de pièces automobiles. Les opérateurs doivent prendre des pièces dans différents bacs pour ensuite les assembler. Les pièces assemblées sont ensuite amenées par des convoyeurs sur les chaînes de montage pour être installées sur les véhicules en préparation. Ces postes ont plusieurs inconvénients. Le premier est la surcharge cognitive de l'opérateur qui doit chercher la bonne pièce dans un grand nombre de bacs, ce qui peut mener à de multiples erreurs. Le second est la perte de temps due aux déplacements de

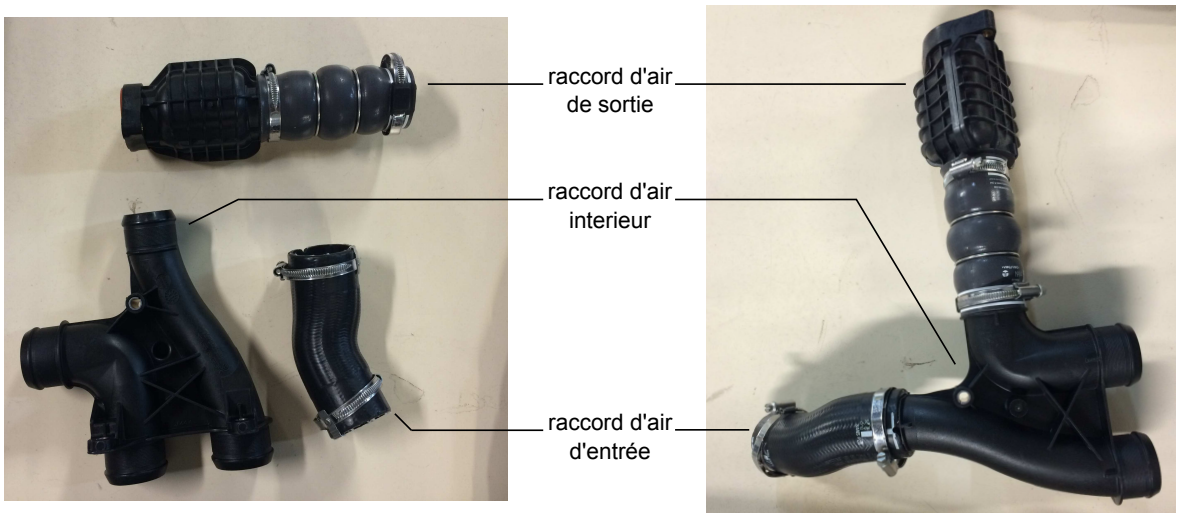


FIGURE 2.20 – Pièces à assembler sur le poste collaboratif, avant et après l'assemblage

l'opérateur pour aller chercher ces pièces. Dans ce contexte, nous pensons qu'une tâche collaborative où un robot apporte les pièces à l'opérateur permet de soulager l'opérateur et de gagner en productivité.

Nous nous sommes plus particulièrement intéressé au montage de raccords d'air de moteur. Trois pièces sont nécessaires à l'assemblage : le raccord d'air d'entrée, le raccord d'air de sortie et le raccord d'air intérieur, Figure 2.20. Des colliers à serrer doivent être vissés pour fixer les différentes pièces entre elles.

Tout comme le poste présenté partie 2.2.1.1, ce poste a été conçu par des roboticiens et des ergonomes de PSA Peugeot Citroën. Le robot et l'opérateur sont séparés par une table et le robot passe les pièces à l'opérateur qui doit les assembler 2.19b. Pour faciliter le passage des pièces, nous avons choisi d'utiliser un robot bi-bras, le robot Motoman SDA20, Figure 2.19a. Contrairement au robot choisi pour le cas d'étude de la co-présence, le Motoman SDA20 n'est pas intrinsèquement sûr. Néanmoins, du fait de sa disposition par rapport à l'opérateur, le robot ne peut l'atteindre qu'avec le bout de ses effecteurs. Il a de plus été programmé pour ne pas dépasser un plan imaginaire perpendiculaire à la table entre l'opérateur et le robot. Lors des enregistrements ou des tests avec un opérateur, un coordinateur avait toujours une main sur le bouton d'arrêt d'urgence pour pouvoir stopper le robot si nécessaire.

2.2.2.2 Choix des gestes

Sur ce poste collaboratif, l'opérateur doit monter cinq raccords d'air de moteur. Pour chaque montage, il a plusieurs actions à faire :

- prendre une pièce dans la pince gauche du robot,
- prendre une pièce dans la pince droite du robot (deux fois),
- assembler deux pièces ensemble (deux fois),
- visser (une ou deux fois),
- poser la pièce assemblée dans une boîte à côté de l'opérateur.

L'ordre des actions n'est pas imposé, mais une organisation logique des tâches est inhérente au montage des pièces. L'opérateur commence par saisir les deux premières pièces, une dans chaque pince du robot, puis les assemble. Il doit ensuite prendre la troisième pièce dans la pince droite du robot, l'assembler aux deux premières. L'opérateur

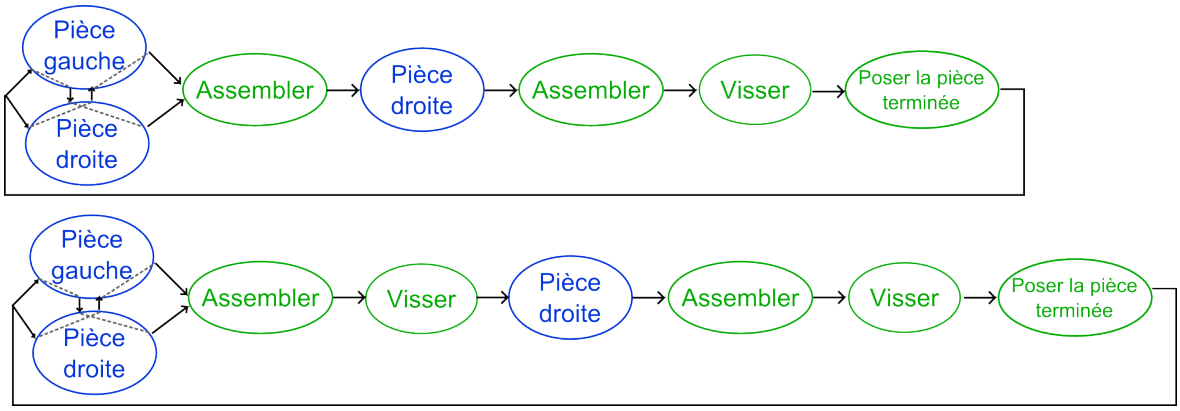


FIGURE 2.21 – Différents ordres possibles des actions pour l'assemblage du raccord de moteur ; En bleu les actions nécessitant la collaboration avec le robot, en vert les actions propres à l'opérateur.

peut choisir : il visse après chaque assemblage ou bien il visse les deux colliers l'un après l'autre lorsque les trois pièces sont assemblées. Une fois toutes ces actions effectuées, il pose la pièce terminée dans la boîte. La Figure 2.21 illustre les possibles enchaînements de ces actions.

Nous souhaitons synchroniser le robot avec l'opérateur pour qu'il lui donne la pièce voulue au bon moment. Les actions nécessitant une action du robot sont écrites en bleu sur la Figure 2.21. Dans le cas où l'opérateur décide de ne visser qu'une seule fois (ligne du haut), il est nécessaire de reconnaître les actions "assembler" et "poser la pièce terminée dans la boîte" pour synchroniser le robot. Dans le cas où l'opérateur choisit de visser après chaque assemblage (ligne du bas), il faut au moins reconnaître les actions "visser" et "poser la pièce terminée dans la boîte" pour assurer la synchronisation du robot. Dans les deux cas, il est nécessaire de savoir quand l'opérateur prend la pièce dans la pince du robot pour pouvoir actionner son ouverture et rendre la collaboration fluide.

Nous avons donc choisi de reconnaître les cinq gestes suivants :

geste 1 : attraper une pièce dans la pince gauche du robot

geste 2 : attraper une pièce dans la pince droite du robot

geste 3 : assembler deux pièces

geste 4 : visser

geste 5 : poser la pièce terminée dans la boîte

La Figure 2.22 illustre ces cinq gestes.

2.2.2.3 Choix des capteurs

Nous souhaitons utiliser des capteurs non intrusifs pour l'opérateur afin qu'il ne soit pas gêné dans ses mouvements. L'utilisation d'une caméra a donc été privilégiée. De plus, pour être robuste aux changements de luminosité, nous nous sommes plutôt intéressés aux caméras de profondeur. Comme les cas de collaboration avec un robot industriel se déroulent la plupart du temps en intérieur, nous n'avons pas de problème avec les possibles interférences entre les rayons infrarouges du soleil et ceux de la caméra. Nous avons choisi d'utiliser une caméra Kinect, une caméra de profondeur peu couteuse ayant une plage d'acquisition allant de 40 cm à 4 m.

De plus, le choix est fait d'utiliser une configuration qui puisse être généralisable à d'autres cas d'étude pour la collaboration avec un robot industriel. Sur les chaînes de

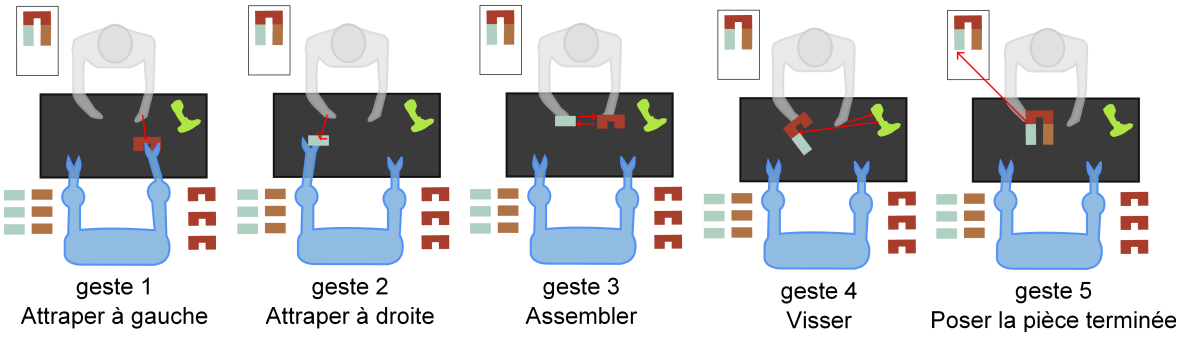


FIGURE 2.22 – Schémas des cinq gestes à reconnaître dans le cas d'étude collaboratif. Geste 1 : Attraper une pièce dans la pince gauche du robot ; Geste 2 : Attraper une pièce dans la pince droite du robot ; Geste 3 : Assembler deux pièces ensemble ; Geste 4 : Visser ; Geste 5 : POser la pièce terminée dans une boîte

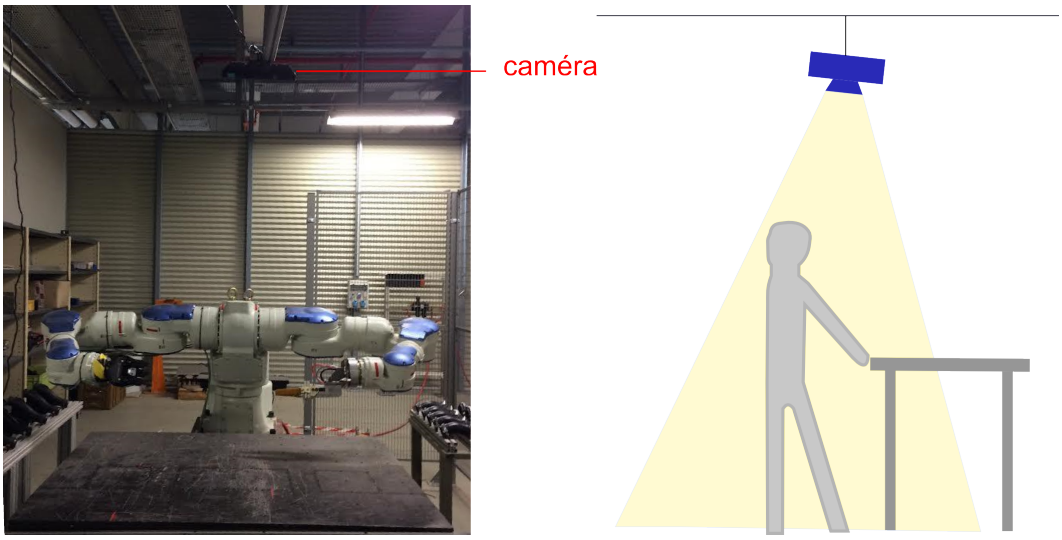


FIGURE 2.23 – Configuration du poste de collaboration avec la caméra de profondeur

montage, des pièces imposantes peuvent passer face à l'opérateur. Sur les côtés de l'opérateur, d'autres personnes sont susceptibles de travailler et de se déplacer. Dans ces deux configurations, une caméra est sujette à de nombreuses occultations. Nous avons donc choisi de mettre une caméra avec une vue de haut, voir Figure 2.23.

Dans les usines automobiles, les outils utilisés par les opérateurs sont "connectés" : par exemple il est possible de savoir quand un opérateur utilise une visseuse. Dans le poste créé pour notre étude, les outils ne sont pas connectés. Donc, pour simuler ces connexions et savoir quand les outils sont utilisés, nous allons mettre un capteur inertiel, de type *MotionPod* comme présenté dans la partie 2.2.1.3, sur la visseuse.

2.3 Conclusion

La première partie de ce chapitre nous a permis de mieux comprendre les enjeux de la collaboration homme-robot en milieu industriel. Les zones de travail partagées entre un robot et un opérateur pourront permettre une plus grande flexibilité et des gains financiers pour les entreprises, mais des contraintes sont également à prendre en compte. En matière de sécurité de l'opérateur, les avancées technologiques permettre d'avoir des robots intrinsèquement sûrs. Le principal enjeu est maintenant de permettre au robot et

à l'opérateur de collaborer de manière fluide et naturelle. Notre réflexion nous a conduit à considérer la reconnaissance des gestes techniques, que nous détaillerons dans le prochain chapitre, pour essayer de répondre à ces problématiques.

Nous avons présenté dans la seconde partie de ce chapitre deux cas d'étude : un cas d'étude de co-présence et un cas d'étude de collaboration.

Pour la co-présence, nous avons présenté un cas portant sur un montage de porte de voiture. Nous avons retenu des gestes permettant de synchroniser le robot avec l'opérateur afin qu'il commence sa tâche lorsque les éléments sur la portière dont il a besoin soient installés et que l'opérateur ait libéré l'espace devant eux. Nous avons choisi d'utiliser des capteurs inertiels. Cela nous permet de vérifier la faisabilité de la reconnaissance des gestes techniques avec des données fiables et non soumises aux occultations.

Pour la collaboration, nous nous sommes intéressés à une tâche de *kitting*. Le robot donne des pièces à l'opérateur qui doit les assembler, les visser et les déposer dans une boîte une fois qu'elles sont montées. Nous avons décomposé l'action de montage de pièces en plusieurs sous-actions que nous allons reconnaître individuellement afin que le robot puisse passer la bonne pièce au bon moment à l'opérateur. Nous avons choisi d'utiliser un capteur non intrusif : une caméra de profondeur. Afin que cette configuration minimise les occultations et soit généralisable, nous avons choisi de positionner la caméra avec une vue de haut.

Chapitre 3

La reconnaissance de gestes : état de l'art

Sommaire

3.1 Le geste technique 36
3.2 La reconnaissance de gestes : vue d'ensemble 37
3.3 La reconnaissance de gestes avec des données vidéos 39
3.3.1 Les caméras 39
3.3.1.1 Caméra 2D 39
3.3.1.2 La stéréo-vision 39
3.3.1.3 Caméra 3D 40
3.3.2 Méthodes basées sur des points d'intérêts 43
3.3.2.1 Détecteurs 43
3.3.2.2 Descripteurs 44
3.3.2.3 Trajectoires 45
3.3.3 Méthodes basées sur des silhouettes 45
3.3.4 Analyse de la succession de postures 47
3.4 La reconnaissance de gestes avec un équipement intrusif 50
3.4.1 Les capteurs inertiels 50
3.4.1.1 Les angles d'Euler 50
3.4.1.2 Les quaternions 51
3.4.1.3 Utilisation pour la reconnaissance de gestes 51
3.4.2 Les marqueurs optiques 52
3.5 La reconnaissance des gestes avec un système hybride vision et inertiel . 53
3.6 Classification des gestes 53
3.6.1 Classification directe 55
3.6.1.1 Sac de Mots 55
3.6.1.2 Séparateur à Vaste Marge (SVM) 55
3.6.1.3 k-Plus Proches Voisins *(k-NN)* 56
3.6.1.4 Le *deep-learning* pour la reconnaissance de gestes 57
3.6.2 Classification avec une approche temporelle 60
3.6.2.1 *Dynamic Time Wraping* 60
3.6.2.2 Modèles de Markov Cachés 61
3.7 Conclusion sur la reconnaissance de gestes 67

Dans le chapitre précédent nous avons introduit les enjeux et problématiques liés à la collaboration entre un opérateur et un robot en milieu industriel. Nous proposons d'utiliser la reconnaissance de gestes pour permettre une collaboration plus fluide et naturelle entre eux. Nous avons également présenté deux cas d'étude pour tester des systèmes de reconnaissance de gestes technique en milieu industriel.

Dans ce chapitre, nous allons présenter des méthodes existantes de reconnaissance de gestes. Dans le cadre de notre étude, nous nous sommes principalement intéressés aux gestes techniques, nous expliquons leurs spécificités ci-dessous. Nous présentons ensuite différentes méthodes pour acquérir et traiter des données provenant de différents capteurs : les caméras 2D et 3D et les capteurs inertiels. Puis nous détaillons des algorithmes de classification pouvant être appliqués à la reconnaissance de gestes.

3.1 Le geste technique

Comme l'expliquent Mitra et Acharya [91], un geste est un mouvement du corps ayant du sens. Un geste peut provenir de mouvements des doigts, des mains, des jambes ou du corps tout entier, avec l'intention de communiquer des informations et/ou d'interagir avec l'environnement. Les gestes constituent donc un sous-espace des mouvements du corps humain. Nous différencions le geste de la posture par le fait qu'un geste est un mouvement dynamique, tandis qu'une posture est une pose statique.

Dans ce manuscrit nous nous intéressons plus particulièrement aux gestes techniques. Selon Cadoz [22], le geste a trois fonctions :

— une fonction ergodique : fonction visant à appliquer une force sur un objet de manière à le modifier ou le déplacer. On peut aussi appeler cette fonction "instrumentale".

— une fonction épistémique : fonction à percevoir de l'information depuis l'environnement, telle une exploration haptique.

— une fonction sémiotique : fonction qui permet de communiquer avec l'environnement afin de transmettre une information.

Dans le cas des gestes techniques, nous nous intéressons plus spécifiquement à la fonction ergodique du geste.

On retrouve le geste technique dans de nombreux domaines comme par exemple le sport, l'artisanat, les arts et, bien entendu, sur les chaînes de montage industrielles, Figure 3.1.

FIGURE 3.1 – Exemples de gestes techniques : à gauche, dans l'industrie ; au centre : artistique ; à droite : dans l'artisanat

Peu d'études ont porté sur le geste technique, néanmoins l'anthropologue André Leroi-Gourhan, dans ses ouvrages [85] et [84], parle du geste technique comme l'objet de la technologie. Ce n'est pas l'outil, mais le geste qui rend la technique évoluée.

> *"L'outil n'est réellement que dans le geste qui le rend techniquement efficace"*, André Leroi-Gourhan

Dans son étude sur les gestes techniques Blandine Bril [19] s'intéresse aux méthodes de description du geste technique. Selon elle les gestes techniques sont composés de mouvements élémentaires de base et de mouvements dérivés. Les mouvements de base sont considérés comme des structures autour desquelles l'activité peut être décrite. L'étude s'est plus particulièrement portée sur l'exemple du pliage et une grande répétabilité des gestes entre les différentes exécutantes a été remarquée.

Dans une autre étude, Bril et Roux [20] expriment le fait qu'un geste technique est par définition un geste appris. De plus, la qualité d'un geste sera évaluée par son impact sur l'environnement.

Un geste technique est donc caractérisé par une première phase d'apprentissage du geste puis ensuite une grande répétabilité, précision et rapidité. Un geste technique peut aussi être qualifié de réussi, ou non, en fonction des objectifs qui lui sont fixés.

Nous allons donc détailler ci-dessous plusieurs méthodes existantes pour la reconnaissance de gestes techniques.

3.2 La reconnaissance de gestes : vue d'ensemble

Dans un premier temps, il faut extraire des données de l'environnement grâce à des capteurs. Ces données peuvent être traitées pour en extraire seulement des informations utiles pour la reconnaissance des gestes. Leur description est ensuite uniformisée pour pouvoir les comparer, pour cela nous utilisons des descripteurs. Un système d'apprentissage et de reconnaissance est ensuite utilisé pour pouvoir labéliser les nouvelles données.

Un système de reconnaissance de gestes est donc caractérisé par plusieurs paramètres :

— le capteur utilisé : les caméras 2D (3.3.1.1), les caméras 3D (3.3.1.3), les systèmes de stéréo-vision (3.3.1.2), les marqueurs optiques (3.4.2) ou les capteurs inertiels (3.4.1)

— le choix du descripteur : les points d'intérêts 3D(3.3.2.1), les trajectoires (3.3.2.3), les silhouettes (3.3.3), les postures (3.3.4), les angles d'Euler (3.4.1.1) ou les quaternions (3.4.1.2)

— le choix du classifieur : les sacs de mots (3.6.1.1), les SVM (3.6.1.2), les k-Plus Proches Voisins (3.6.1.3), les CNN (3.6.1.4), les DTW (3.6.2.1) ou les HMM (3.6.2.2)

Nous allons, dans la suite de ce chapitre, décrire l'ensemble de ces méthodes.

Le Tableau 3.1 répertorie les descripteurs que nous détaillons ci-dessous en fonction des capteurs utilisés.

TABLEAU 3.1 – Descripteurs pour la reconnaissance de gestes en fonction du/des capteurs utilisé(s)

	Points d'intérêts 3D	**Trajectoires**	**Silhouettes**	**Postures**	**Angles**
Caméra 2D	Laptev et Lindberg [76] Schuldt et al. [112] Willems et al. [139] Dollár et al. [35] Niebles et al. [94] Bregonzio et al. [17] Rapantzikos et al. [105] Oikonomopoulos et al. [98] Tang et al. [124] Wang et al. [132] Liu et al. [88] Lee et al. [80]	Wang et al. [130] Wang et Schmid [131] Jain et al. [57]	Bobick et Davis [15] Blank et al. [14] Achard et al. [1] Yilmaz et Shah [142] Weinland et al. [136] Jiang et Martin [57] Ke et al. [65] *Pons-Moll et al.*[101] *Ó Conaire et al.* [96] Yamato et al. [141]	Fujiyoshi et Lipton [41] Bourdev et Malik [16] Dantone et al. [33] Tompson et al. [126] Tompson et al. [125] *Liu et al.* [87] Gavrila et al. [43]	
Caméra 3D				Chen at al. [25] Plagemann et al. [100] Shotton et al. [118] Holt et al. [54] Schwarz et al. [113] Siddiqui et Medioni [119] Migniot et Absabsa [90] Ganapathi et al. [42] *Helten et al.* [51] Sempena et al. [115] Reyes et al. [107] Zhu et al. [143] Xia et al. [140]	
Capteurs inertiels					Bulling et al. [21] Dong et al. [36] Pylvänäinen [103] *Pons-Moll et al.*[101] *Helten et al.* [51] *Ó Conaire et al.* [96] *Liu et al.* [87]

3.3 La reconnaissance de gestes avec des données vidéos

La reconnaissance de gestes avec des données issues de caméra est un domaine qui a été largement étudié. Il a plusieurs domaines d'applications, comme la télésurveillance, la compréhension de la langue des signes ou bien sûr l'interaction homme-machine. L'utilisation de données vidéo permet d'avoir une bonne compréhension de la scène. Néanmoins ces données sont sensibles aux occultations. Des revues, notamment celles de Weinland et al. [136] et de Aggarwal et Ryoo [2] ont fait un état de l'art détaillé des méthodes pour reconnaître des actions via des données vidéos.

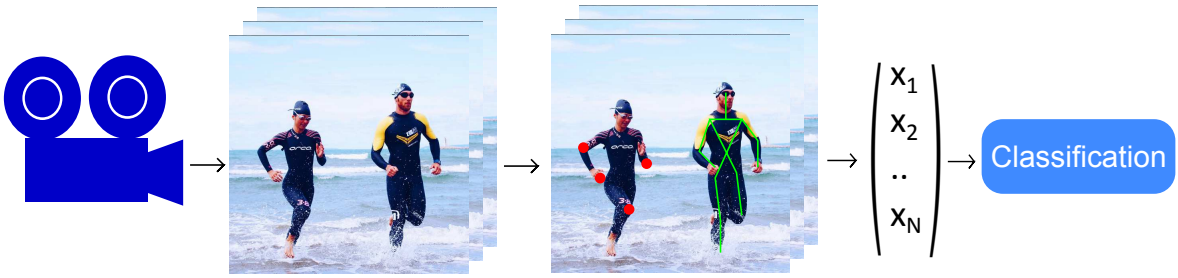


FIGURE 3.2 – Exemple de trame pour la reconnaissance de gestes avec des données vidéo. La caméra acquiert des images, dont on extrait des informations, par exemple des points d'intérêt (points rouges), ou un squelette (segments verts). Ces informations sont ensuite uniformisées d'une image à une autre pour permettre une classification.

Sur la Figure 3.2, on peut voir une trame illustrant les différentes étapes de la reconnaissance d'actions à partir de données vidéo. Dans cette partie, nous allons dans un premier temps présenter deux types de caméras, puis détailler différentes méthodes pour extraire des informations de vidéos en vue de reconnaître des actions.

3.3.1 Les caméras

Pour la reconnaissance de gestes, la vision par ordinateur permet une captation non-intrusive. Deux familles de capteurs, les caméras 2D et les caméras 3D, sont présentées ci dessous. Nous introduisons également la stéréo-vision qui permet de calculer la profondeur des objets dans une scène lorsque celle-ci est filmée par deux caméras.

3.3.1.1 Caméra 2D

Une caméra 2D permet de faire l'acquisition d'images représentant une scène de manière analogue à ce que perçoit un œil humain : pour chaque pixel de l'image, on dispose d'une information sur l'intensité ou la couleur de l'objet filmé. Une image est donc une matrice de pixels et donne des informations sur l'apparence et la texture des objets dans la scène. Ces caméras peuvent donner des images en niveaux de gris, ou alors en couleurs. Chaque pixel est alors codé par sa composition de trois couleurs primaires : le rouge, le vert et le bleu. On peut appeler les images couleur, images RGB (*Red, Green, Blue*) .

3.3.1.2 La stéréo-vision

La stéréo-vision permet de reconstruire une structure 3D en utilisant des données de deux caméras 2D. Elle est fortement inspirée de la vision humain qui permet à l'homme de percevoir la profondeur en utilisant les images issues de chacun de ses yeux. En effet, si deux images d'une même scène sont acquises avec des angles différents, on peut observer

des différences entre ces deux images. Pour estimer la profondeur d'un objet dans une scène, on calcule une valeur appelée *disparité*. Un modèle géométrique simplifié, dérivé du *pinhole model* est illustré Figure 3.3. On peut y voir :

— c_1 et c_2, les deux caméras
— f, la focale des caméras
— p, un point de l'objet dans la scène
— b, la baseline
— p_1 (respectivement p_2), la projection du point p dans l'image issue de la caméra c_1 (respectivement c_2)

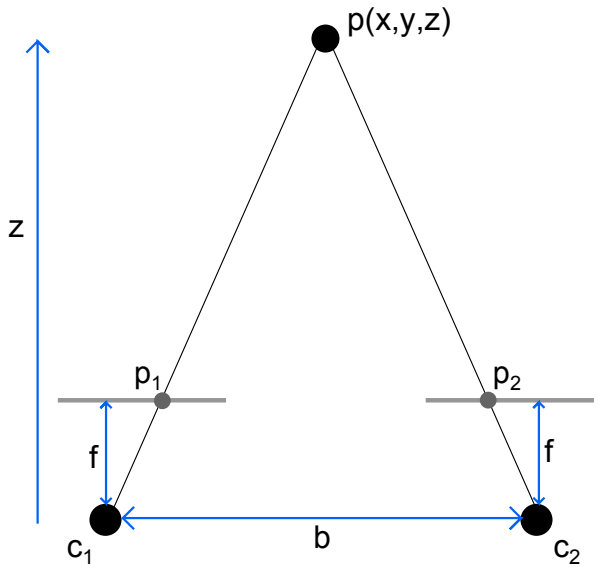


FIGURE 3.3 – Modèle géométrique simplifié du système de stéréo-vision

Selon le principe de la triangulation, la disparité peut être calculée comme suivant :

$$\frac{z}{f} = \frac{x}{x_1} = \frac{x-b}{x_2} = \frac{y}{y_1} = \frac{y}{y_2} \tag{3.1}$$

On peut en déduire la disparité d :

$$d = \frac{fb}{z} \tag{3.2}$$

La valeur de la disparité augmente lorsque la distance entre l'objet et la caméra diminue, et augmente lorsque la distance s'accroît. Une première étape avant de calculer la disparité est d'apparier des points d'intérêt dans les deux images, plusieurs méthodes existent, nous n'allons pas les détailler ici.

3.3.1.3 Caméra 3D

Bien que les informations données par les caméras 2D peuvent s'avérer très intéressantes pour la reconnaissance de gestes, elles restent sensibles aux changements de luminosité, ce qui diminue leur robustesse.

Les caméras 3D, ou caméras de profondeur, ont la particularité de donner des informations de distance entre la scène filmée et la caméra. Pour ces caméras, la valeur de

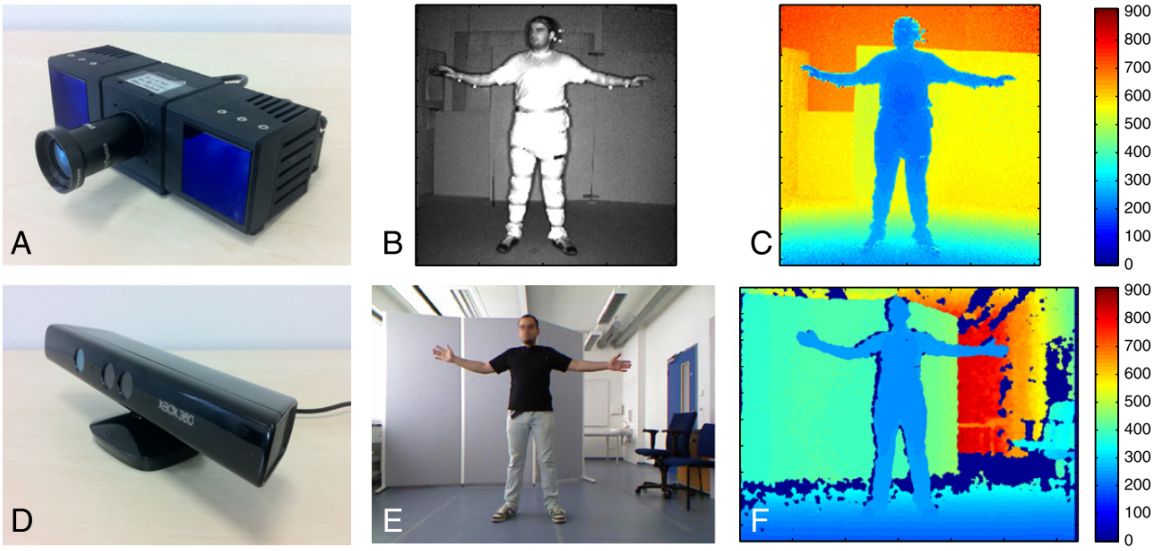


FIGURE 3.4 – Cartes de profondeurs issues d'une caméra technologie "temps de vol" et d'une caméra technologie "lumière structurée". A : caméra temps de vol, B : amplitude de l'image, C : carte de profondeur issue de la caméra temps de vol, D : caméra projetant de la lumière structurée, E : image RGB, F : carte de profondeur issue de la caméra projetant de la lumière structurée. Image issue de ©Schwarz et al.[113].

chaque pixel nous informe sur la distance entre la caméra et l'objet filmé. L'image résultante est appelée carte de profondeur ou *depth map* en anglais. Le grand avantage de ces caméras est qu'on peut directement extraire, des cartes de profondeur, la géométrie de la scène. De plus, ces caméras ne sont pas ou peu sensibles aux changements de luminosité. Il existe deux grandes familles de caméras de profondeur : les caméras basées sur une technologie de temps de vol et les caméras projetant des lumières structurées, voir Figure 3.4. Le tableau 3.2, présente un récapitulatif de différentes caméras de profondeur avec leurs caractéristiques.

Caméras temps de vol La technologie des caméras temps de vol est décrite par Klob et al. [68]. Pour estimer la distance entre un objet et la caméra, les caméras utilisent des raies de lumière infra-rouge modulées. En mesurant le retard de phase de la lumière réfléchie sur l'objet filmé, ils en déduisent la distance parcourue par la lumière, voir Figure 3.5. Connaissant le décalage de phase $\Delta\varphi$, on en déduit la distance parcourue, D, avec l'équation :

$$\mathrm{D} = \frac{\Delta\varphi c}{4\pi f} \tag{3.3}$$

sachant que :

$$\Delta\varphi = 2\pi f \Delta t \tag{3.4}$$

avec c la vitesse de la lumière, f la fréquence du signal et Δt le décalage temporel entre le signal émis et le signal reçu.

La plupart des caméras de profondeur basées sur la technologie de temps de vol ont une faible résolution. L'utilisation simultanée de plusieurs caméras avec la même modulation de ce type peut fausser les cartes de profondeur à cause d'interférences. De plus, des réflexions multiples d'une même raie émise par le capteur ainsi que les surfaces peu réfléchissantes (absorbantes ou transparentes) peuvent également être sources d'erreurs.

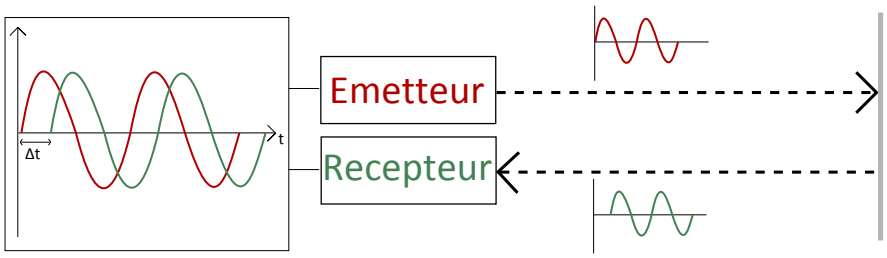


FIGURE 3.5 – Schéma illustrant le fonctionnement d'une caméra 3D temps de vol

Caméra projetant de la lumière structurée Les caméras utilisant de la lumière structurée projettent un motif infra rouge sur la scène filmée. La profondeur des objets dans la scène est estimé en évaluant la distorsion locale de ce motif. Le calcul de distance se fait en utilisant les positions connues de l'émetteur, du récepteur, et de la forme du signal émis dans la scène. Cette technique permet une résolution plus élevée de la carte de profondeur, mais, tout comme les caméra temps-de-vol, certaines surfaces, ayant une couleur foncée ou étant au contraire transparentes, peuvent être source d'erreur de mesure. De plus, le motif doit être visible par le récepteur et l'émetteur, ce qui résulte en des zones non-mesurées sur les cotés des objets filmés. Enfin, tout comme pour les caméras temps de vol, si l'éclairage de la scène contient une forte intensité de la même longueur d'onde que celle du motif projeté (la lumière du soleil par exemple), le motif peut devenir indétectable. C'est pourquoi ce type de caméra 3D n'est généralement utilisable qu'en intérieur, contrairement aux caméras temps de vol qui peuvent fonctionner en extérieur.

TABLEAU 3.2 – Comparaison de caractéristiques de caméras 3D

Nom de la caméra	Technologie	Résolution (pixels)	Plage d'acquisition (mètre)	FPS	FOV	Prix
ASUS® XtionPro ™Live	lumière structurée	640x480	0.8 - 3.5	30	58°H 45°V	180 €
Microsoft® Kinect ™1.0	lumière structurée	640x480	0.4 - 4	30	57°H 43°V	130 €
Microsoft® Kinect ™2.0	temps de vol	512x424	0.5 - 4.5	30	70°H 60°V	150 €
PMD PhotonICs®	temps de vol	160x120	0 - 2	90	90°H 68°V	?
IFM® Efector™03D303	temps de vol	176x132	0.3 - 8	25	45°H 60°V	1029 €
Ensenso® N35-606-16-BL	lumière structurée	1280x1024	0.25 - 3.0	10	58°H 52°V	?
SICK® 3visitor-T™	temps de vol	176x144	0.5 - 7.2	30	69°H 56°V	?

Nous allons ci-dessous présenter plusieurs méthodes pour extraire des information depuis des données vidéos pour la reconnaissance de gestes. Nous allons nous intéresser à l'extraction de volumes spatio-temporels, de silhouettes et de postures.

3.3.2 Méthodes basées sur des points d'intérêts

Les premières solutions pour reconnaître des actions dans des vidéos sont dérivées de méthodes pour identifier des objets dans des images. Ces méthodes consistent à trouver dans des images des pixels comportant des singularités au niveau local. Un objet est alors représenté par un ensemble de ces points d'intérêt. La reconnaissance d'objet consiste à extraire de nouveaux points d'intérêt dans de nouvelles images puis les comparer avec ceux associés à un objet déjà connu. Les recherches pour caractériser des actions se sont en premier lieu intéressées à l'application et/ou l'adaptation de ces méthodes aux vidéos. Leur but est de trouver des caractéristiques aux mouvements humains afin de pouvoir les discerner et reconnaître. Pour les vidéos, ce sont des volumes spatio-temporels qui sont recherchés. Ces volumes spatio-temporels sont des extensions des points d'intérêt issues des images en y ajoutant une dimension temporelle. La classification d'actions se fait alors par appariement de ces volumes spatio-temporels. Plusieurs méthodes ont été développées pour trouver ces volumes dans les vidéos. Une fois détectés, des descripteurs permettent d'uniformiser la description de ces volumes spatio-temporel afin de pouvoir les comparer. Ci dessous nous allons détailler des détecteurs qui permettent de trouver des points d'intérêt, puis des descripteurs qui sont utilisés pour les décrire. Nous nous sommes également intéressés à l'applications du *deep-learning* dans ce domaine et à l'utilisation de trajectoires pour décrire une action.

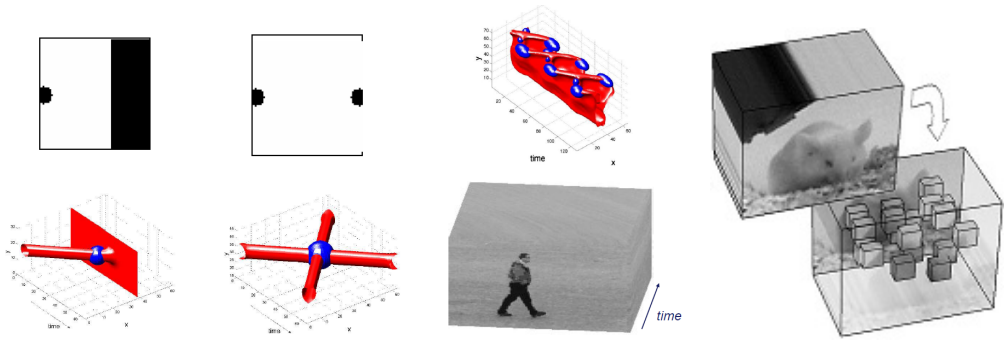
3.3.2.1 Détecteurs

Une extension du détecteur Harris [50] a été proposée par Laptev et Lindberg en 2003 [76]. Le détecteur de Harris permet de détecter des coins dans une image, c'est à dire les points de l'image pour lesquels les valeurs des pixels varient dans toutes les directions. Ce détecteur est invariant aux rotations et aux translations mais pas aux changements d'échelles. Laptev et Lindberg ont généralisé ce détecteur aux vidéos avec un nouveau détecteur, communément appelé Harris 3D, en cherchant des coins dans un volume spatio-temporel 3D. Ils permettent de détecter des changements de direction d'un objet, des divisions, des fusions ou des collisions par exemple, Figures 3.6a et 3.6b. Ces détecteurs sont utilisés pour caractériser des mouvements humains. Ils furent utilisés par Schuldt et al. [112] avec des SVM pour reconnaître des actions dans des vidéos de la base de données KTH [1].

Willems et al. [139] utilise le déterminant de matrices Hessiennes 3D pour définir la présence de points saillants dans un volume spatio-temporel.

Le détecteur cuboïde a été introduit par Dollár en 2005 [35] basé sur les filtres de Gabor, Figure 3.6c. Ce détecteur permet de mettre en évidence des variations locales et périodiques d'intensité, mais pas de décrire des mouvements de translation. Le cuboïde est un ensemble de pixels dans une fenêtre temporelle centré sur un pixel ayant eu une forte réponse aux filtres de Gabor. Ce détecteur a été utilisé par Niebles [94] avec une méthode d'apprentissage non supervisée en deux étapes. Une première étape de partitionnement des cuboïdes avec l'algorithme des *K-Means* permet la création d'un *codebook*. Le seconde étape consiste à créer un modèle de geste à partir des clusters du *codebook*

1. http ://www.nada.kth.se/cvap/actions/



(a) (b) (c)

FIGURE 3.6 – Exemples de points d'intérêts issus du détecteur Harris 3D (©Laptev et Lindeberg [76]) (a et b) et du détecteur de cuboïdes (c) (©Dollár et al.[35]).

en utilisant la méthode d'apprentissage non supervisée pLSA *probalistic Latent Semantic Analysis*. En 2009, Bregonzio et al. [17] ont proposé une version améliorée du détecteur cuboïde qui était trop sensible au bruit, ignorait les mouvements de translation et détectait des points d'intérêt dans des arrière-plans texturés. La détection se fait alors en deux étapes. Une première étape consiste à calculer la différence de deux images consécutives d'une vidéo afin de ne s'intéresser qu'aux zones en mouvement. La seconde étape applique des filtres de Gabor 2D avec différentes orientations sur ces zones. Une autre extension des cuboïdes a été faite par Rapantzikos et al. [105] qui a également utilisé des informations sur la couleur, l'intensité et le mouvement pour déterminer des points d'intérêt.

Dans Oikonomopoulos et al. [98] les auteurs présentent une méthode pour trouver les points saillants en calculant des maximums locaux d'énergie représentant des pics d'activité. La taille des régions spatio-temporelles autour des points d'intérêts détectés est déterminée en maximisant l'entropie. En 2009, Wang et al. [132] ont montré que le *dense sampling* donnait de meilleurs résultats pour la reconnaissance d'actions sur plusieurs dataset que les détecteurs cuboïdes [35], Harris3D [76] et ceux basés sur les matrices Hessiennes 3D [139]. Le *dense sampling* extrait des volumes 3D de vidéo a des positions régulières à plusieurs échelles temporelles et spatiales.

3.3.2.2 Descripteurs

Une fois les volumes spatio-temporels détectés, il faut les décrire de manière uniforme pour pouvoir les comparer indépendamment de changement d'échelle ou d'orientation. En plus de son détecteur, Dollár [35] proposa également un descripteur pour ses cuboïdes. Un vecteur contient la valeur du gradient et du flux optique de chaque pixel du volume spatio-temporel, puis une PCA, *Principal Component Analysis*, est appliquée pour en réduire la dimension. Niebles et al. [94] ont préféré proposer leur propre descripteur pour décrire des cuboïdes. Ils n'utilisent que les valeurs du gradient, après filtrage et calculés sur plusieurs échelles.

En 2008, Laptev et al. [77] ont proposé un descripteur alliant histogrammes de flux optique et histogrammes de gradients HOG/HOF. Le volume spatio-temporel est divisé en une grille de cellules 3D. Pour chaque cellule, un histogramme de gradients et de flux optique est calculé, puis tous ces histogrammes sont concaténés en un vecteur qui sert de descripteur au volume spatio-temporel.

Le descripteur SIFT pour les points d'intérêt dans les images, introduit par Lowe [89],

a été généralisé aux volumes 3D par Scovanner et al. [114], en créant le descripteur SIFT 3D. De la même manière, le descripteur SURF introduit par Bay et al. [8] a été adapté aux vidéos par Willems et al. [139], renommé alors descripteur eSURF. Le volume 3D est divisé en plusieurs cellules. Chaque cellule est représentée par un vecteur contenant des sommes pondérées de réponses aux ondelettes de Haar selon les trois axes, x, y et t.

Dalal et al. [32] introduit le descripteur MBH *Motion Boundary Histograms* qui calcule les dérivées horizontales et verticales du flux vidéo. Ce descripteur se veut plus robuste aux mouvements de caméra.

3.3.2.3 Trajectoires

Afin d'avoir une meilleure vision de l'évolution temporelle d'un mouvement, des études se sont concentrées sur des trajectoires de points d'intérêt. Une méthode a été détaillée par Wang et al. [130]. Dans un premier temps, des points d'intérêt sont sélectionnés à la manière du *dense sampling* en échantillonnant une image sur plusieurs échelles et en ne gardant que des points n'appartenant pas à des zones homogènes. Un suivi de ces points est fait en utilisant le flux optique de la vidéo. Plusieurs descripteurs, HOG, HOF et MBH, sont calculés pour décrire des volumes délimités par ces trajectoires. Une illustration de l'algorithme peut être vue Figure 3.7. L'utilisation de ces trajectoires a ensuite été améliorée par Wang et Schmid [131] pour prendre en compte le mouvement de la caméra.

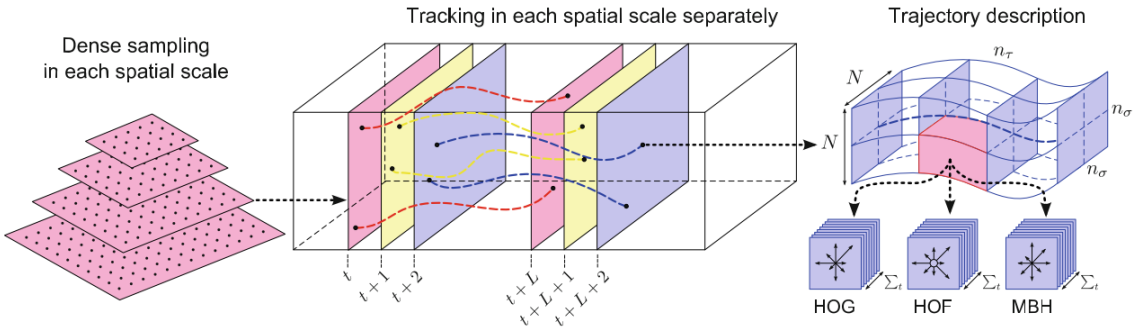


FIGURE 3.7 – Illustration des travaux de (©Wang et al [130]) : *dense sampling* de la vidéo, suivi des points d'intérêt et description des volumes entre les trajectoires par des descripteurs HOG, HOF et MBH

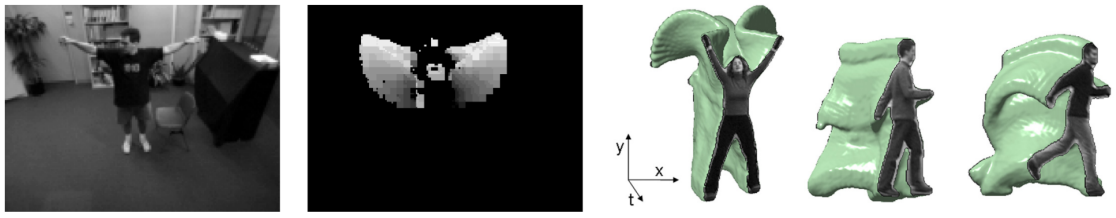
Jain et al. [57] proposent une méthode pour décomposer le mouvement dans l'image en deux classes : le mouvement résiduel et le mouvement dominant. Ils en extraient des trajectoires afin de reconnaître des actions humaines.

3.3.3 Méthodes basées sur des silhouettes

Pour reconnaître une action, une autre approche consiste à étudier une succession de mouvements en analysant la silhouette de l'acteur.

Une première étape pour localiser la personne filmée doit être utilisée. Ce peut être par une simple soustraction de l'arrière-plan, si la caméra est fixe et que l'auteur bouge, ou dans les autres cas par une localisation de la personne plus complexe. Ces méthodes sont souvent faciles à mettre en place, rapides à calculer et donnent de bons résultats. Néanmoins, elles sont soit limitées à l'utilisation d'une caméra fixe et d'un arrière-plan statique, soit dépendante de la fiabilité de l'algorithme de détection et de segmentation de la personne. De plus, elles peuvent être sensibles aux conditions d'illumination, apparition d'ombres par exemple.

Une des premières utilisations des silhouettes pour la reconnaissance de gestes a été faite par Bobick et Davis [15]. Ils utilisent une caméra fixe et ils créent une image binaire accumulant les zones de mouvement dans la séquence vidéo appelée MEI, *Motion-Energy Image*. Ils créent ensuite une MHI, *Motion-History Image*, en niveaux de gris, où la valeur de chaque pixel est d'autant plus grande qu'un mouvement sur ce pixel a été détecté récemment, voir Figure 3.8a. Les actions sont ensuite reconnues en effectuant un *template matching* en utilisant les moments de Hu.



(a) (b)

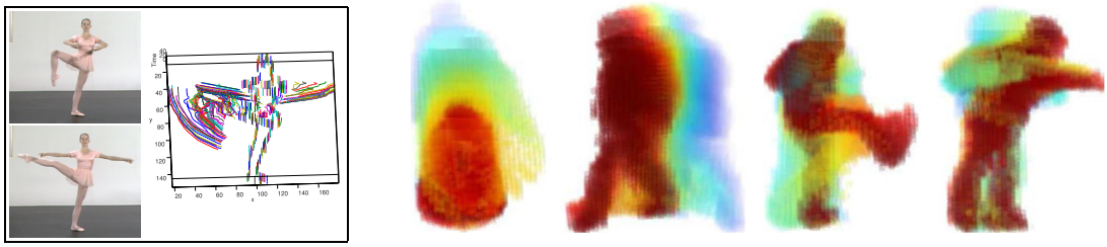
FIGURE 3.8 – (a) Illustration du MHI pour les actions "agiter les bras" et "s'accroupir" (©Bobick et Davis [15]), (b) Accumulation de silhouettes selon la méthode de Blank et al. (©Blank et al [14])

En 2005, Blank et al. [14] ont proposé une solution de reconnaissance d'actions en utilisant des silhouettes. Ils extraient pour chaque image de la vidéo la silhouette de la personne filmée en soustrayant le fond, voir Figure 3.8b. Chaque action est alors une accumulation temporelle de silhouettes . Pour chaque silhouette, les auteurs déterminent plusieurs caractéristiques de la forme en utilisant les équations de Poisson. Ces caractéristiques seront utilisées pour la classification des actions en utilisant l'algorithme des plus proches voisins. Achard et al. [1] comparent chaque image de la vidéo à l'image initiale pour créer un volume spatio-temporel contenant les pixels en mouvement dans la séquence. Ces "micros-volumes" sont ensuite décrits grâce à un ensemble de moments de deuxième et troisième ordre. La reconnaissance d'action se fait alors à l'aide de HMM.

FIGURE 3.9 – Construction d'une surface 3D à partir de silhouettes issues de plusieurs images consécutives (©Yilmaz et Shah [142])

Yilmaz et Shah [142] proposent une méthode basée sur la création d'un volume 3D à partir des contours successifs de silhouette, voir Figure 3.9. Des points singuliers (pic, fosse, vallée, arête...) sont ensuite calculés sur ces surfaces pour pouvoir reconnaître quelle action a été effectuée. Cette méthode est assez coûteuse en calcul et sensible au bruit car le volume 3D est directement calculé à partir de contours dans l'image. Weinland et al. [135] utilisent cette méthode pour étendre les MHI de Bobick et Davis [15] au domaine 3D en créant des MHV, *Motion History Volumes*, Figure 3.10b. L'extension au domaine 3D des *Motion Energy Image* permet une indépendance du point de vue pour la comparaison des silhouettes. Les MHV sont décrits en coordonnées cylindriques autour de l'axe z

puis représentés par des vecteurs de Fourier. La comparaison entre deux actions se fait en calculant la distance de Mahalanobis entre les deux vecteurs de Fourier qui leurs sont associés. Jiang et Martin [59] ont utilisé les contours de silhouettes. Ils calculent un *shape flow*, c'est à dire un ensemble de lignes 3D représentant le suivi de points du contour de la silhouette, pour décrire un mouvement, voir Figure 3.10a.



(a) (b)

FIGURE 3.10 – (a) *Shape flow* d'un mouvement de danseuse (©Jiang et Martin [59]), (b) *Motion History Volumes*, pour les actions s'assoir, marcher, donner un coup de pied et donner un coup de poing (©Weinland et al. [135])

Ke et al. [65] se sont intéressés à une méthode qui ne nécessite pas une soustraction du fond. Ils segmentent les vidéos en des volumes 3D homogènes avec un algorithme de partitionnement, ou *clustering*, non supervisé. Ils utilisent ensuite des informations sur les formes et les flux optiques des vidéos pour reconnaître des actions.

3.3.4 Analyse de la succession de postures

Lorsqu'on s'intéresse aux mouvements du corps humain, une solution instinctive pour le décrire est de suivre le mouvement du squelette de la personne filmée. En effet l'utilisation seule de la silhouette ne permet pas de décrire un mouvement complexe ou de faible amplitude, comme boire par exemple. Johansson [60] explique qu'un mouvement peut être décrit en observant seulement le mouvement des articulations principales du corps humain. Pour cela il a attaché des points lumineux aux articulations d'un volontaire et a conclu qu'une fois en mouvement, la connaissance des positions successives de ces points lumineux permettait de reconnaître le mouvement effectué.

Fujiyoshi et Lipton [41] ont utilisé la silhouette d'une personne filmée pour en déduire un squelette simplifié. Le squelette, représenté par une étoile à 5 branches maximum, correspondant aux bras, jambes et à la tête. La reconnaissance des actions se fait ensuite par reconnaissance de cycles de cette étoile. Cette méthode qui fut novatrice est néanmoins, comme les méthodes basées sur l'extraction de silhouettes, sensible au bruit et nécessite une caméra fixe avec un arrière-plan identique au cours du temps. De plus elle ne reconnait presque exclusivement que les mouvements périodiques allant de gauche vers la droite (ou droite vers la gauche). Bourdev et Malik [16] ont introduit les "poselets", des parties de postures qui ont une faible variabilité entre elles, donc facile à classer et à détecter. La détection de ces "poselets" permet ensuite de retrouver la posture d'une personne dans une image. Ils utilisent 2000 images annotées pour entrainer leur classifieur. Dantone et al. [33] utilisent deux couches successives de forêts d'arbres aléatoires, ou RDF "*random decision forest*" en anglais, pour trouver la localisation des articulations d'une personne dans une image. La première couche permet de labelliser les différentes parties du corps, tandis que la seconde détermine ensuite la position des articulations. La base de données pour l'apprentissage des forêts d'arbres aléatoire comporte plus de

6 000 images. Renna et al. [106] ont utilisé un filtrage particulaire pour mettre à jour un modèle 3D représentant le haut du corps de la personne filmée. Après avoir soustrait le fond d'écran, ils déterminent des zones d'énergie dans l'image, c'est à dire les zones en mouvement, ainsi que les zones où la peau de l'utilisateur est visible dans l'image. Ces données sont utilisées pour mettre à jour le modèle 3D du haut du corps.

FIGURE 3.11 – Resultats des travaux de (©Tompson et al. [125])

L'utilisation du *deep-learning*, voir partie 3.6.1.4, a permis de faire des estimations de postures à partir d'images RGB. Toshev et Szegedy [127] utilisent des réseaux de neurones profonds (*Deep Neural Network* ou plus simplement abrégé DNN) pour déterminer la position des articulations d'une personne dans une image. Une première étape consiste à appliquer un DNN à l'image entière pour déterminer une première position de chaque articulation. Une seconde étape consiste à appliquer localement autour de chaque articulation une cascade de DNN afin d'affiner leur résultat. Tompson et al. [126] présentent une méthode hybride de CNN combiné à un modèle de distributions de la position de chaque articulation *B* connaissant la position de l'articulation *A*. Ils arrivent à déterminer la posture de personnes dans des images de manière robuste. Cette méthode fut améliorée en 2015, par Tompson et al. [125], pour rendre plus précise la localisation des articulations.

Avec l'arrivée des caméras de profondeur, et donc la possibilité d'un accès plus facile à la géométrie des objets filmés, la squelettisation du corps humain est devenue un nouvel axe de recherche étudié par de nombreuses équipes. Une revue de plusieurs de ces études a été faite par Chen et al. [25]. Nous allons en détailler quelques-unes ci-dessous.

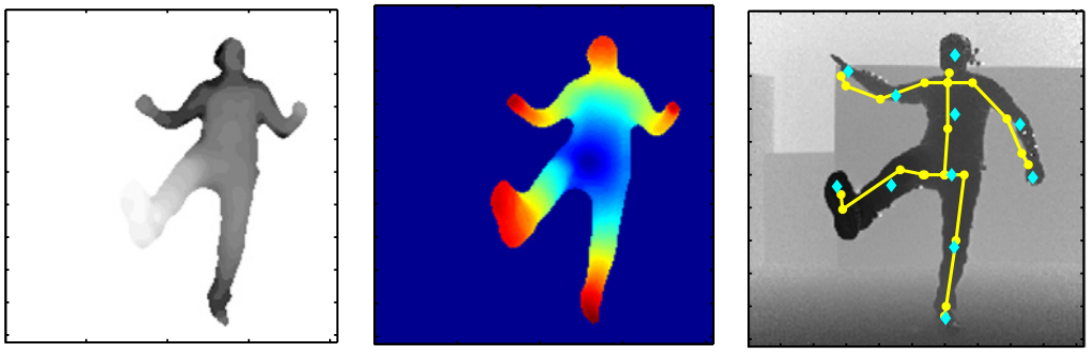
FIGURE 3.12 – Illustration des travaux de (©Shotton et al. [117]) : à gauche : une personne filmée avec une caméra de profondeur, au centre : les pixels après classification, à droite : position des articulations

En 2010, Plagemann et al. [100] ont proposé une méthode de squelettisation basée sur des images issues de caméras de profondeur utilisant la technologie du temps de vol. Ils déterminent les positions d'articulations en cherchant des maxima locaux de la dis-

tance géodésique entre les points du corps et le centroïde de l'ensemble de ces points. La classification de ces points se fait ensuite en utilisant une méthode de *boosting*.

Une des squelettisation les plus connue du grand public est celle fournie par le SDK de la Kinect [2] : un capteur RGB-D commercialisé par Microsoft . Les travaux ont été menés par l'équipe de Shotton et al. [117]. Ils squelettisent un utilisateur de la Kinect en se basant sur une forêt d'arbres aléatoires entrainée avec un grand nombre de données, réelles et synthétiques, labellisées (presque 1 million d'images) provenant de personnes ayant des morphologies variées. Ils ont utilisé des primitives de différence de profondeur pour entrainer leur algorithme de classification afin que chaque pixel appartenant à la personne filmée soit classé comme appartenant à une des 31 parties du corps humain. La position des articulations est ensuite calculée comme centre de gravité des points appartenant à une même classe. Cette méthode donne des bons résultats et permet une détection du squelette en temps réel. Néanmoins, elle a exclusivement été entraînée pour la squelettisation de personne filmée de face, et du fait du grand nombre de données nécessaires à sa réalisation, elle est difficilement utilisable pour créer un système de squelettisation avec une vue différente. Holt et al. [54] ont proposé une méthode similaire, revendiquant ne pas avoir besoin d'une aussi grande base de donnée que Shotton et al. [117]. Leur méthode repose sur la détection de "poselets", comme définit dans Bourdev et Malik [16], puis leur classification avec une forêt d'arbres aléatoires.

Schwarz et al. [113] ont proposé une méthode utilisant l'image de profondeur pour créer le squelette de la personne filmée sans avoir recours à une base de donnée et un apprentissage. Après avoir extrait le corps de la personne filmée grâce à une soustraction de l'arrière-plan, Figure 3.13a, les distances géodésiques, entre chaque point du corps et le centre de gravité de l'ensemble des points, sont calculées, Figure 3.13b. Les auteurs font l'hypothèse que les distances géodésiques entre les articulations d'un corps humain et le centre de ce corps sont constantes, quelle que soit la posture de la personne filmée. Ils en déduisent les positions des articulations en prenant en compte des données morphologiques, Figure 3.13c.



(a) (b) (c)

FIGURE 3.13 – Squelettisation avec la méthode de Schwarz et al. [113] (a) : Corps humain après soustraction de l'arrière plan, (b) : Distances géodésiques de chaque point du corps au centre de gravité de celui-ci et (c) : Positions des articulations et squelette final

En 2010, Siddiqui et Medioni [119] ont proposé une méthode utilisant une méthode de Monte-Carlo par chaîne de Markov pour comparer des images 3D synthétiques avec une image 3D actuelle et en déduire la posture la plus probable. Migniot et Ababsa [90] comparent également un modèle 3D du haut d'un corps humain avec une personne filmée

2. https ://developer.microsoft.com/fr-fr/windows/kinect

avec une vue de haut pour déterminer la posture de plus probable ce celle -ci. Ganapathi et al. [42] ont proposé un algorithme de filtrage pour suivre les positions d'une personne filmée avec une caméra de profondeur. Ils utilisent la méthode de Plagemann et al. [100] pour détecter dans un premier temps les différentes parties du corps.

3.4 La reconnaissance de gestes avec un équipement intrusif

Les capteurs de vision ne sont pas les seuls capteurs à être utilisés pour la reconnaissance de gestes. Tandis que les caméras sont sensibles aux occultations, des équipements intrusifs, c'est à dire portés par les utilisateurs, permettent d'avoir des informations directes sur les mouvements à reconnaître. Nous allons voir ici deux types de capteurs : les capteurs inertiels et les marqueurs.

3.4.1 Les capteurs inertiels

Lors de l'acquisition de données avec des capteurs inertiels, des informations sur le mouvement sont directement accessibles, via des accéléromètres et des gyroscopes. Lors de l'utilisation de gyroscopes, les angles aux articulations peuvent directement être utilisés pour décrire l'action. Il existe plusieurs manières de les représenter qui peuvent influer sur la robustesse du système de reconnaissance. Nous allons ci-dessous présenter les angles d'Euler et les quaternions, puis des méthodes utilisant des capteurs inertiels pour la reconnaissance de gestes.

3.4.1.1 Les angles d'Euler

Les angles d'Euler permettent de décrire, avec trois valeurs, la rotation d'un système par rapport à sa position précédente. Selon le théorème d'Euler on peut passer d'une orientation à une autre grâce à une suite de trois rotations élémentaires, voir Figure 3.14 :

- Le précession : ψ
- La nutation : θ
- La rotation propre : φ

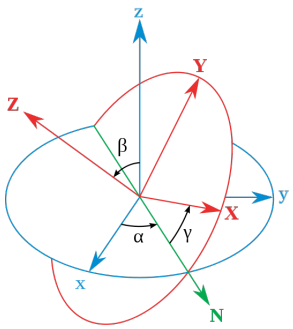


FIGURE 3.14 – Angles d'Euler, passage du repère bleu au repère rouge

Bien que facile à comprendre et à calculer, le principal inconvénient des angles d'Euler est le blocage de cadran, appelé *gimbal lock* en anglais. Le blocage de cadran est la perte d'un degré de liberté lorsque les axes de deux des trois cadrans utilisés pour effectuer les rotations ont la même direction. Pour pallier ce problème, on peut utiliser les quaternions.

3.4.1.2 Les quaternions

Un quaternion a quatre composantes : q_1, q_2, q_3 et q_4 telles que Q = $1.q_1 + i.q_2 + j.q_3 + k.q_4$ avec $i^2 = -1$, $j^2 = -1$, $k^2 = -1$.

Une rotation peut être représentée par un quaternion unité, c'est à dire tel que |Q| = 1. Lorsque l'on veut décrire une rotation selon l'axe $[u_x, u_y, u_z]$ et d'angle θ on utilise le quaternion correspondant : $q = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2})u_x.i + \sin(\frac{\theta}{2})u_y.j + \sin(\frac{\theta}{2})u_z.k$.

La rotation d'un vecteur **v** se fait en calculant $\mathbf{Q.v2.Q^{-1}}$ avec $v2$ un quaternion avec la partie scalaire nulle et la partie vectorielle égale à **v**.

Bien que moins intuitif que les angles d'Euler, les quaternions sont plus faciles d'utilisation et ne présentent pas le problème du blocage de cadran. De plus, lorsque l'on veut faire deux rotations successives, représentées par q_1 et q_2, le quaternion équivalent se calcule facilement par $q_1.q_2$.

3.4.1.3 Utilisation pour la reconnaissance de gestes

Bulling et al. [21] ont proposé un tutoriel complet sur la reconnaissance d'actions en utilisant des capteurs inertiels. Ils présentent une "*Acitivity Recognition Chain*", une trame générale pour la reconnaissance de gestes avec ces capteurs, Figure 3.15.

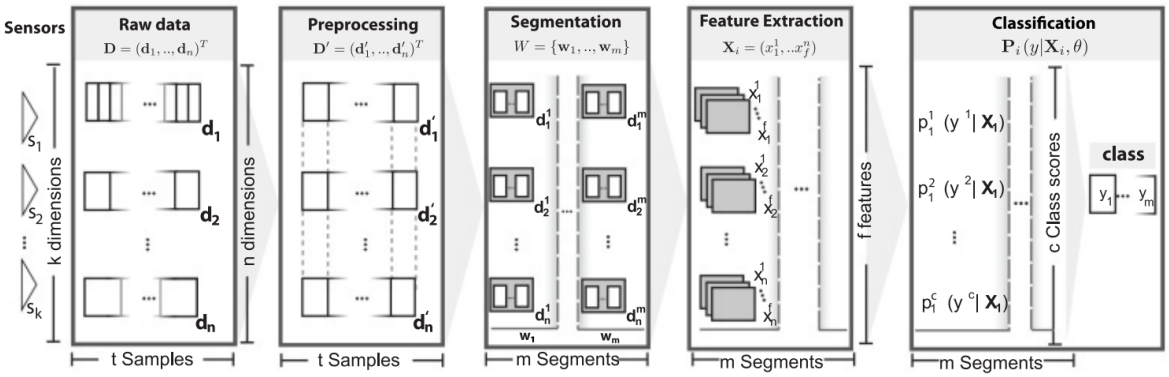


FIGURE 3.15 – "*Acitivity Recognition Chain*" (©Bulling et al. [21]). Description des étape de la trame présentée : acquisition des données, préprocessing, segmentation, extraction de descripteurs et classification

Des méthodes sont présentées pour chaque étape de cette trame qui sont l'acquisition des données, le préprocessing, la segmentation, l'extraction de descripteurs et la classification. Benbasat et al. [9] ont également présenté un système de reconnaissance de gestes avec des capteurs inertiels. Ils utilisent des mesures inertielles avec 6 degrés de liberté. Ils se servent ensuite d'HMMs pour reconnaître, axe par axe, des gestes atomiques qui peuvent ensuite être combinés en des gestes plus complexes.

Nous pouvons également présenter les méthodes ci-dessous qui se sont basées sur des données provenant d'accéléromètres pour reconnaître des actions. En 2007, Dong et al. [36] ont utilisé des accéléromètres apposés sur des segments du corps comme le torse, les bras et les jambes, pour reconnaître des actions. Ils divisent les mouvements en quatre classes : "posture statique", "mouvement mineur", "mouvement périodique" et "mouvement". Ils convertissent les accélérations en angles de rotation qu'ils utilisent pour reconnaître des actions via des HMMs. Pylvänäinen [103] a proposé une méthode pour uniformiser des données provenant d'accéléromètres afin de reconnaître des gestes avec des HMMs continus. Junker et al. [61] ont mené une étude sur la reconnaissance de gestes avec des accéléromètres. Ils se sont dans un premier temps intéressés à la sélection,

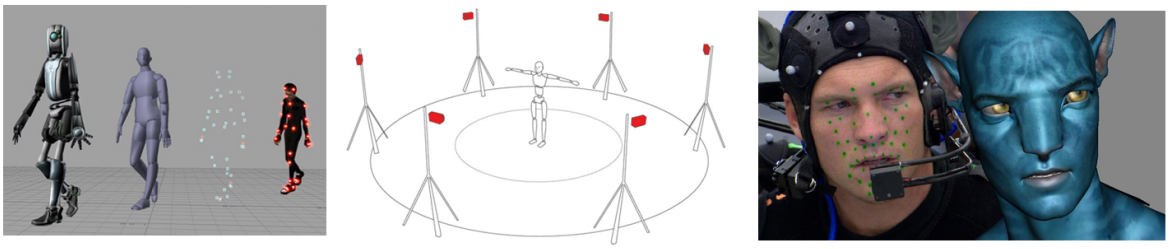
dans un flux de données constant, de données pouvant être des gestes à reconnaître. Ils effectuent ensuite une classification avec des HMMs. Cooney et al. [30] ont essayé de reconnaitre les gestes que faisait un petit robot humanoïde lorsque celui ci était manipulé par des humains : "faire marcher", "balancer", "mettre debout", "faire l'avion" font partie des gestes à reconnaître. Pour cela ils utilisent des capteurs embarqués dans le robot, un accéléromètre et un gyromètre. Pour labéliser les gestes, ils ont choisi d'utiliser des SVM.

3.4.2 Les marqueurs optiques

Les marqueurs optiques sont des points, généralement réfléchissants, apposés sur une personne, le plus souvent sur des zones caractéristiques du mouvement, comme les articulations. Des images binaires, issues de caméras infrarouges, sont générées et, grâce à un suivi des marqueurs, les mouvements peuvent être enregistrés.

Cette méthode est beaucoup utilisée dans le cinéma pour créer des images de synthèses en 3 dimensions. Pour cela, l'acteur est équipé avec un grand nombre de marqueurs et est filmé par plusieurs caméras, voir Figure 3.16. Par appariement des positions dans les différents angles de vue on peut retrouver la position 3D de chaque marqueur. Cette méthode est difficile à mettre en place pour la reconnaissance de gestes dans des lieux qui ne sont pas adaptés. De plus, pour éviter les occultations et avoir des informations fiables sur les mouvements de la personne équipée, un grand nombre de caméras est nécessaire.

Kirk et al. [66] détaillent un algorithme permettant de suivre les mouvements d'une personne équipée de marqueurs en reconstruisant son squelette. L'utilisateur est doté d'au moins un ou deux marqueurs par segment du corps humain.



(a) (b) (c)

FIGURE 3.16 – Utilisation de marqueurs pour la reconnaissance de mouvements. (a) : Détails des étapes de reconstruction d'images de synthèses : extraction des marqueurs, reconstruction du squelette puis mise en mouvements d'un personnage fictif. (b) : Dispositif pour la génération d'images de synthèse en 3D avec des marqueurs. (c) : Application à l'animation de visage avec pour exemple le film Avatar.

Qautre sociétés se partagent le marché des capteurs optiques : Vicon [3], MotionAnalysis [4], Natural Point [5] et OptiTrack [6].

3. www.vicon.com
4. www.motionanalysis.com
5. www.naturalpoint.com
6. www.optitrack.com/

3.5 La reconnaissance des gestes avec un système hybride vision et inertiel

L'utilisation jointe de capteurs de vision et de capteurs inertiels a également été utilisée pour rendre plus robuste un système de reconnaissance de gestes. Ils ont des caractéristiques qui les rendent complémentaires. Les capteurs de vision permettent d'avoir une connaissance sur l'ensemble de la scène, mais sont sensibles aux occultations. Les capteurs inertiels permettent de connaître des informations fiables sur les mouvements de la personne qui les porte mais n'informent pas sur l'environnement et sont intrusifs.

L'utilisation de données provenant de capteurs de vision et de capteurs inertiels peut servir en amont de la classification des gestes, pour améliorer le suivi des parties du corps de la personne effectuant une action. Pons-Moll et al. [101] ont proposé une méthode se basant sur les caméras 3D et des capteurs inertiels pour améliorer la fluidité et la précision de l'estimation de postures basées sur seulement la vision. En 2013, Helten et al. [51] ont utilisé un système de capteurs combinant une caméra de profondeur et six capteurs inertiels pour déterminer la posture d'une personne filmée. Deux postures dans la base de données de référence sont choisies, une se basant sur les données de la caméra, l'autre sur les données des capteurs inertiels. Le choix final se fait en utilisant des mesures de fiabilité sur les deux postures. Elles pourront ensuite être utilisées pour reconnaître des actions.

Les données provenant des capteurs de vision et de capteurs inertiels peuvent être utilisées directement pour la classification, comme descripteurs du geste qui a été exécutés. Liu et al. [87] ont combiné des informations provenant de la caméra de profondeur et de capteurs inertiels pour reconnaître des gestes de la main. Ils concatènent ces informations dans un même vecteur à chaque pas de temps et se servent de ces données fusionnées pour apprendre des HMMs et ensuite reconnaître des gestes comme "dessiner un cercle", ou "dessiner un X". Ó Conaire et al. [96] ont également utilisé ces deux types de capteurs pour reconnaître des actions au tennis et en extraient différentes informations. Depuis les données RGB des caméras, ils obtiennent la silhouette du joueur et ils utilisent directement les données brutes des accéléromètres. Ils pondèrent les données des deux capteurs avec un "poids de confiance" appris sur une base de données d'apprentissage avant de les fusionner. Il se servent des données alors fusionnées pour reconnaître les gestes des joueurs.

3.6 Classification des gestes

Nous allons présenter dans cette partie plusieurs méthodes de classification de gestes. Nous les avons classées en deux catégories : les algorithmes ne prenant pas en compte la dimension temporelle dans la partie 3.6.1, et les algorithmes prenant en compte la dimension temporelle du geste dans la partie 3.6.2. Dans sa revue, Poppe [102] fait un état de l'art sur la classification d'actions avec des données vidéos. Les méthodes qui y sont présentées peuvent également être appliquées à la reconnaissance d'actions avec des capteurs inertiels. C'est aussi le cas des algorithmes détaillés ci-dessous.

Le Tableau 3.3 recense l'ensemble des méthodes de reconnaissance des gestes de cette partie en fonction des descripteurs et du classifieur utilisés.

TABLEAU 3.3 – Classification des gestes en fonctions des descripteurs utilisés

	Sac de Mots	**SVM**	**k-NN**	**DTW**	**HMM**	**CNN**
Points d'intérêts 3D	Dollár et al. [35] Laptev et al. [77] Niebles et al. [94] Willems et al. [139]	Schuldt et al. [112] *Bregonzio et al.* [17] Wang et al. [132]	*Bregonzio et al.* [17] Rapantzikos et al. [105] Oikonomopoulos et al. [98]		Liu et al [88] Lee et al. [80] Tang et al. [124]	
Trajectoires	Wang et al. [130]	Jain et al. [131] Wang et al. [57]				
Silhouettes		Ke et al. [65] *Ó Conaire et al.* [96]	Blank et al. [14] *Ó Conaire et al.* [96]		Yamato et al. [141] Achard et al. [1]	
Postures		Bourdev et Malik [16]		Gavrila et Davis [43] Sempena et al. [115] Reyes et al. [107]	Zhu et Pun [143] Xia et al. [140] *Liu et al.* [87]	
Descripteurs issus du *deep-learning*						Wang et al. [133] Ji et al. [58] Karpathy et al. [63] Simonyan et al. [120] Du et al. [37] Molchanov et al. [92]
Angles		*Ó Conaire et al.* [96]	Dong et al. [36] Pylavänäinen [103] *Ó Conaire et al.* [96]	Liu et al. [86]	*Liu et al.* [87]	

3.6.1 Classification directe

3.6.1.1 Sac de Mots

Les Sacs de Mots, ou *Bag of Words*, est une méthode de classification se basant sur l'occurrence de "mots-clefs", ou caractéristiques, dans une action, par exemple. Cette méthode découle de l'analyse de textes où un document peut être décrit par un histogramme d'occurrences des mots-clefs le composant, sans prendre en compte leur ordre, d'où le terme "sac".

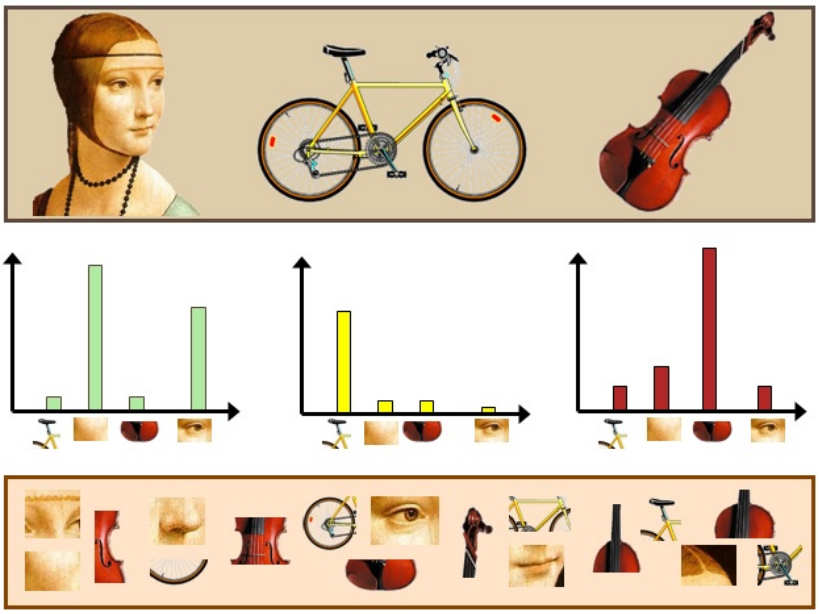


FIGURE 3.17 – Représentation des Sacs de Mots. Sur la première ligne on peut voir les trois objets de la base de donnée, sur la dernière ligne est représenté le dictionnaire de l'ensemble des caractéristiques (mots-clefs) de ces trois objets. Au milieu, nous pouvons voir les histogrammes d'occurrences des caractéristiques pour chaque objet (image issue de la présentation de Li Fei-Fei à ICCV 2005).

En vision par ordinateur, cette méthode a également été utilisée pour reconnaître des objets dans des images. Les mots-clefs sont alors extraits par une méthode non supervisée, comme des K-Means, à partir d'un ensemble de descripteurs issus de la base de données. Ils constituent le vocabulaire d'un "dictionnaire" dont chaque mot correspond au centroïde d'un cluster issu du K-Means. Un histogramme décrivant, pour chaque classe, l'occurrence de chaque mots du dictionnaire est alors calculé, Figure 3.17.

La classification de nouveaux objets est effectuée avec des classifieurs bayésiens ou avec des SVM par exemple. Cette méthode a été utilisée pour la reconnaissance d'actions par Dollár et al. [35], Laptev et al. [77], Niebles et al. [94] et Willems et al. [139] sur des points d'intérêt 3D.

3.6.1.2 Séparateur à Vaste Marge (SVM)

Les séparateurs à vaste marge, ou *Support Vector Machine* en anglais, proposé par Vapnik [128] est un algorithme d'apprentissage supervisé, c'est à dire que les données pour l'apprentissage sont labellisées. Il permet de créer une frontière dans un espace de caractéristiques entre deux ensembles d'échantillons labellisés. Pour cela, on détermine un hyperplan optimal séparant ces deux classes dans l'espace des caractéristiques. Il doit

classifier correctement les échantillons en se trouvant "le plus loin possible" de ceux ci. Lorsque les données ne sont pas linéairement séparables, on les projette dans un autre espace de dimension généralement supérieure en utilisant un noyau de fonction. En choisissant le bon noyau, les caractéristiques deviennent linéairement séparables.

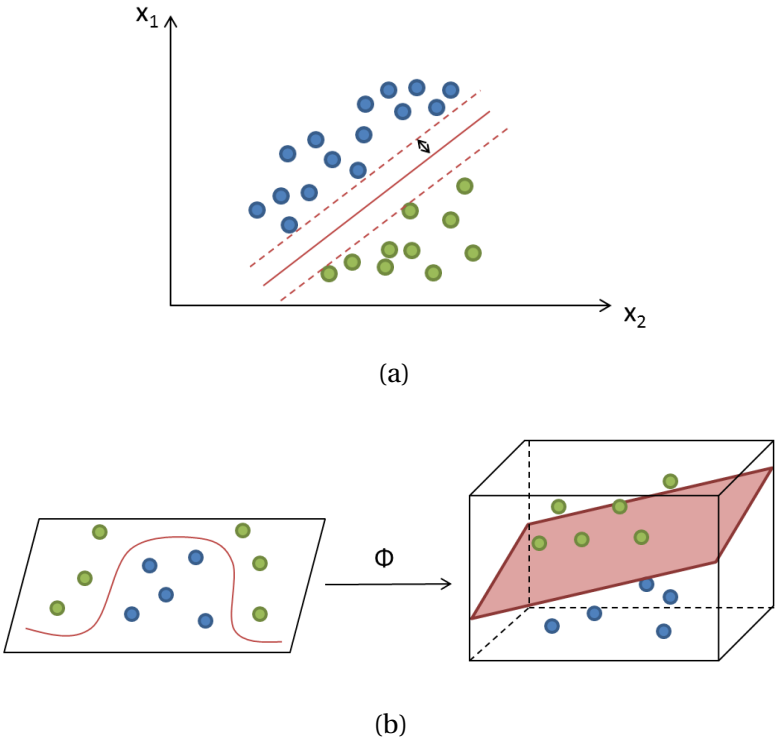


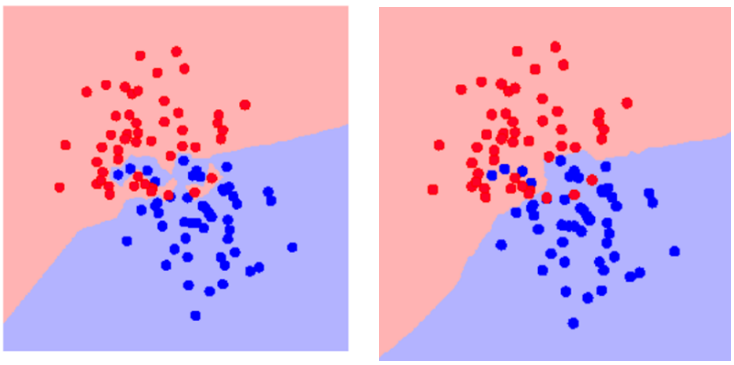
FIGURE 3.18 – Illustration des SVM (a) : Cas linéairement séparable. (b) : Cas non-linéairement séparable. On passe dans un espace de dimension supérieure pour déterminer un hyperplan séparant les deux classes

Cette méthode a été utilisée pour la reconnaissance d'action par Schuldt et al. [112], Bregonzio et al. [17], Wang et al. [132] sur des points d'intérêt 3D. Jain et al. [57] les ont appliqués aux trajectoires et Ke et al. [65] aux silhouettes.

3.6.1.3 k-Plus Proches Voisins *(k-NN)*

La méthode des k-Plus Proches Voisins , ou *k-NN (k Nearest Neighbors)*, est une méthode de classification supervisée multi-classes. Dans un espace de caractéristiques, un nouvel échantillon sera labellisé en fonction de la classe la plus représentée parmi celles de ses k plus proches voisins, Figure 3.19. La méthode des k-Plus Proches Voisins est simple à implémenter et permet facilement d'ajouter de nouveaux exemples dans la base d'apprentissage. Néanmoins, le temps de calcul pour déterminer le label d'un nouvel échantillon est proportionnel au nombre d'éléments dans l'ensemble d'apprentissage et à leur dimension, ce qui peut vite devenir très gourmand. De plus, le choix du nombre de voisins pris en compte, k, influe sur les résultats : un k trop petit donnera un système sensible au bruit, tandis qu'un k trop grand inclura trop de voisins d'autres classes.

Cette méthode a été utilisée pour la reconnaissance d'action par Blank et al. [14] en utilisant la distance de Hausdorff pour calculer la distance entre deux suites de silhouettes. Bregonzio et al. [17] compare les k-NN avec les SVM pour reconnaître des actions e utilisant des points d'intérêt 3D, et obtient des résultats similaires avec les deux méthodes. Rapantzikos et al. [105] combinent la méthodes des Sacs de Mots avec des k-NN pour



(a) k = 1 (b) k = 7

FIGURE 3.19 – Illustration des Plus Proches Voisins. Séparation de l'espace des caractéristiques en fonction du nombre de voisins pris en compte

reconnaître des actions. Oikonomopoulos et al. [98] ont utilisé les k-NN pour tester leur détecteurs, des points d'intérêt 3D.

3.6.1.4 Le *deep-learning* pour la reconnaissance de gestes

Présentation La classification d'actions dans des vidéos se fait par l'extension des réseaux de neurones à convolutions 2D (*Convolutional Neural Networks*, CNN), surtout appliqués à la reconnaissance d'objets dans les images, au domaine 2D $+t$, ou 3D. Les CNN 2D ont été popularisés avec les travaux de LeCun et al. [78] en 1990 sur la reconnaissance de caractères écrits. Les CNN consistent en une succession de couches qui comportent des cartes de caractéristiques et des cartes de sous-échantillonnage. Le nombre de couches, le nombre de cartes et leurs dimensions sont des paramètres qui varient d'une problématique à une autre.

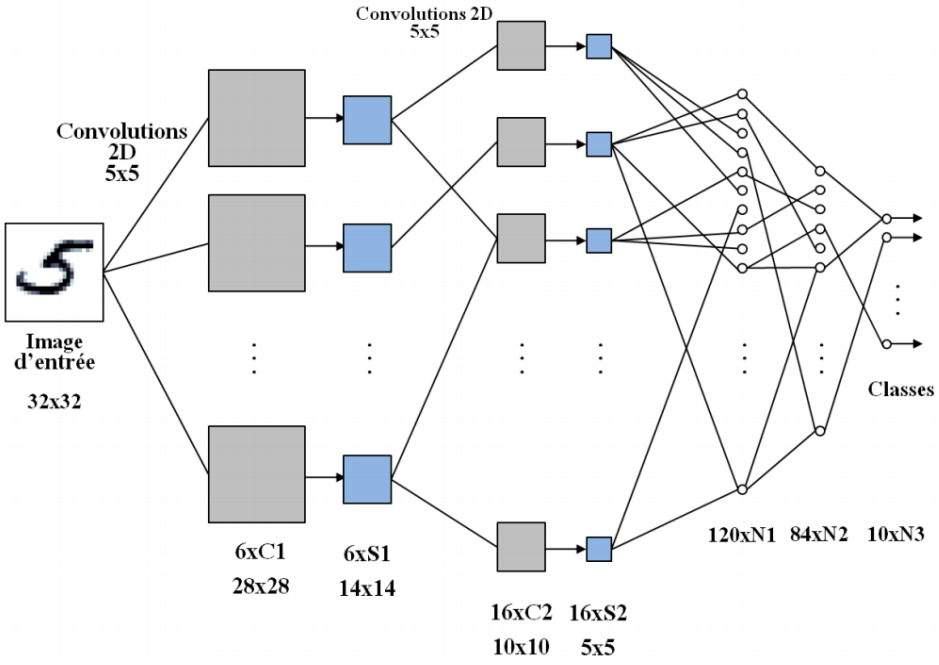


FIGURE 3.20 – Exemple d'une architecture d'un réseau de neurones à convolution (©LeCun et al. [79])

La Figure 3.20 illustre une architecture d'un réseau de neurone à convolution 2D. Ce

réseau de neurones est composé de 7 couches cachées. Les couches représentées par des carrés gris correspondent à des cartes de caractéristiques et sont notées C_i. Les couches représentées par des carrés bleus illustrent des cartes de sous échantillonnage, elles sont notées S_i. Enfin les trois dernières couches sont des neurones, N_i, et sont représentés par des ronds blancs. Les couches C_i appliquent sur leurs entrées des convolutions 2D, dont les noyaux doivent être appris, puis alimentent les couches supérieures. Les couches S_i appliquent un moyennage spatial sur les entrées puis les multiplient par un poids. La succession de ces couches permet de mettre en évidence des informations saillantes partir de l'image d'entrée et à les encoder en un vecteur (descripteur). Les trois dernières couches, les neurones N_i, permettent de classer les données encodées. Ces neurones sont connectées entre elles par des connections pondérées, ces poids sont également des paramètres du modèle. Les neurones appliquent deux traitements à leurs entrées : une combinaison linéaire de celles-ci avec les paramètres du réseau (poids), et une fonction non linéaire appelée *fonction d'activation*. Les deux fonctions les plus courantes sont la tangente hyperbolique et la fonction sigmoïde. En sortie de ce réseau de neurone, nous aurons un score pour chaque classe qui permettra de labelliser l'image en entrée.

L'objectif de cette méthode est de construire une représentation de l'image brute d'entrée de plus en plus haut niveau de couche en couche. On parle alors d'apprentissage "profond", de *deep-learning* en anglais ou DNN pour *Deep Neural Network*.

Nous avons ici présenté les CNN, qui sont un type de réseaux de neurones. D'autres structures existent, comme les RNN *Recurrent Neural Network* par exemple, que nous ne présenterons pas dans ce manuscrit.

Bien que prometteur, le *deep-learning* impose la contrainte d'avoir une grande base de données labellisée pour effectuer son apprentissage. Des méthodes ont néanmoins été proposées pour augmenter les bases de données. Krizhevsky et al. [70] ont par exemple effectué des translations, des réflexions horizontales sur des sous parties d'images de la base de données. Les auteurs ont également modifié les intensité de chacun des canaux de couleur rouge, vert et bleu pour étoffer leur base de données.

Méthodes utilisant le *deep-learning* pour la reconnaissance de gestes dans les vidéos
Le choix du type de points d'intérêts ou d'un autre genre de descripteur pour la reconnaissance de geste a un grand impact sur les performances du système de reconnaissance. Néanmoins, choisir les bons descripteurs pour représenter un geste ou un mouvement peut être non intuitif. Des recherches se sont alors penchées sur le *deep-learning* pour déterminer des descripteurs de manière moins "artisanale". L'utilisation du *deep-learning* pour la reconnaissance d'objets dans les images ayant déjà fait ses preuves [79], [52] et [10], des recherches ont tenté d'étendre ces résultats à la reconnaissance d'actions dans les vidéos. Les descripteurs sont créés en utilisant les premières couches de cartes de caractéristiques et de sous-échantillonnage, tandis que la classification se fait sur les dernières couches du réseau de neurones.

Ji et al. [58] ont proposé une méthode basée sur des réseaux de neurones convolutifs 3D (*Convolutional Neural Networks* ou *CNN 3D*) en étendant les CNN 2D, utilisés pour les images, aux vidéos en prenant en compte la dimension temporelle. La méthode a été utilisée pour reconnaître des actions filmées par des caméras de vidéos surveillances. En 2014, Karpathy et al. [63] ont également utilisé des réseaux de neurones convolutifs pour reconnaître des actions dans des vidéos. Ils testent différentes méthodes de fusion de données pour prendre en compte plusieurs trames pour entrainer leurs CNN. Les meilleurs résultats sont obtenus avec une "slow fusion", c'est à dire les premières couches de convolutions prennent en compte des informations contenues dans 4 trames consécutives, puis

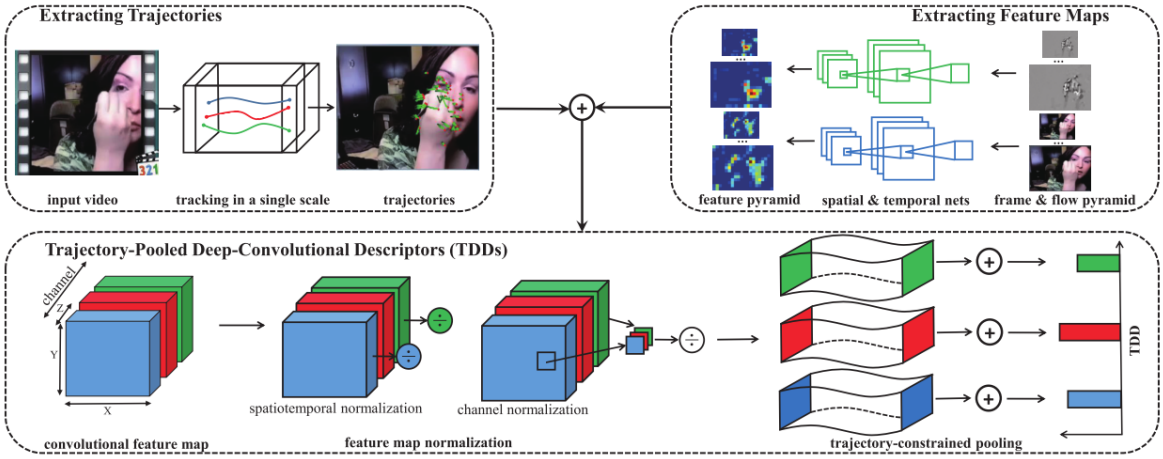


FIGURE 3.21 – Illustration des travaux de (©Wang et al [133]) : Les auteurs utilisent les trajectoires selon [131], en haut à gauche et des CNN selon [120] pour reconnaître des actions.

ces couches sont ensuite fusionnées entre elles de manière à ce que la couche résultante contienne des informations provenant de 10 images successives. Néanmoins, les résultats obtenus ne sont pas meilleurs que ceux utilisant des descripteurs "artisanaux" comme Wang et al.[130] par exemple. Simonyan et al. [120] fusionnent deux CNN : un CNN spatial entrainé sur des images issues de vidéos et un CNN temporel entrainé sur des flux optiques issus de ces mêmes vidéos. Cela leur permet d'avoir des informations sur le mouvement ainsi que sur l'apparence. La fusion de données des sorties des deux CNN se fait soit par SVM, soit par moyennage. Ils obtiennent de meilleurs résultats que Karpathy et al. [63] et des résultats similaire à ceux obtenus par Wang et al.[130].

Tout comme les points d'intérêt, la méthode du *deep-learning* a été utilisée avec les trajectoires. Dans Wang et al. [133] les auteurs combinent les méthodes de Simonyan et al. [120] avec les trajectoires améliorées de Wang et Schmid [131] pour reconnaître des actions dans des vidéos, voir Figure 3.21.

Les données provenant des caméras de profondeur ont dans un premier temps été utilisées pour reconnaître des objets dans des images. Socher et al. [121] ont notamment utilisé les données provenant du capteur RGB et du capteur de profondeur de la Kinect pour reconnaître des objets en utilisant un système composés de CNN et de RNN. Les auteurs Du et al. [37] ont utilisé des RNN hiérarchiques pour reconnaître des actions dans des vidéos. Pour cela ils utilisent les squelettes des personnes filmées, divisent les parties du corps en 5 groupes qui alimentent chacun un RNN et qui sont finalement fusionnés pour prendre une décision finale. Les auteurs utilisent notamment une base de données créée a partir de squelettes calculés avec des données de profondeur provenant de la Kinect. Kang et al. [62] ont utilisé des CNN pour reconnaître les lettre de l'alphabet du langage des signes en utilisant des données de profondeur. Néanmoins, dans ce cas, il s'agit plus de la reconnaissance de postures que de gestes dynamiques. Molchanov et al. [92] ont utilisé des CNNs et des données provenant du capteur RGB et depth d'une Kinect pour reconnaître des gestes afin de limiter l'utilisation de bouton dans l'habitacle d'une voiture. Néanmoins, la combinaison des données provenant des caméras de profondeur pour faire de la reconnaissance de gestes avec des méthode de *deep-learning* a été encore un domaine peu étudié mais qui semble être prometteur.

3.6.2 Classification avec une approche temporelle

3.6.2.1 *Dynamic Time Wraping*

L'algorithme DTW (*Dynamic Time Wraping*), ou déformation temporelle dynamique en français, est un algorithme déterministe utilisé pour évaluer la similarité entre deux signaux pouvant ne pas être exécutés à la même vitesse. Il a dans un premier temps été utilisé pour la reconnaissance de la parole, notamment dans Myers et al. [93].

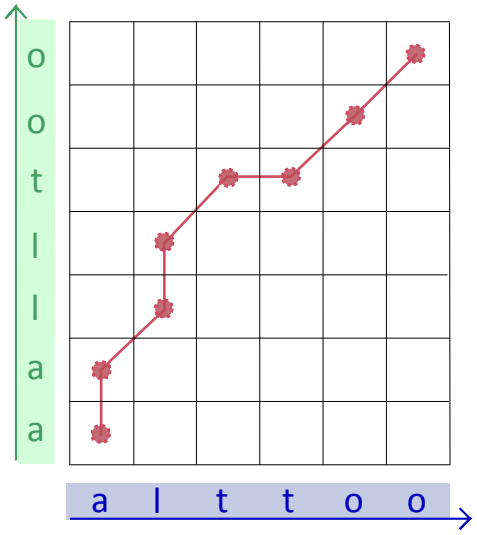


FIGURE 3.22 – Exemple d'alignement de deux mots identiques mais ayant été prononcé a des vitesses différentes. En rose, le chemin optimal alignant les deux mots.

Le but de l'algorithme est de minimiser la distance entre les deux signaux en prenant en compte trois hypothèses :

— Le début et la fin des deux signaux sont connus et concordent

— L'ordre des échantillons des signaux est conservé

— Tous les échantillons sont pris en compte pour calculer la distance entre les deux signaux

L'algorithme teste toutes les configurations possibles pour de trouver le chemin optimal garantissant un alignement des deux signaux, Figure 3.22. Si on considère deux signaux

$$\mathrm{R}(n), n = 1..\mathrm{N}$$

et

$$\mathrm{T}(m), m = 1..\mathrm{M}$$

il faut associer chaque point du signal R avec chaque point du signal T selon un chemin de plus faible coût

$$w : [1,\mathrm{N}] \rightarrow [1,\mathrm{M}]$$

tel qu'il minimise

$$\mathrm{D}ist(w) = \sum_{n=1}^{\mathrm{N}} d(\mathrm{R}(n), \mathrm{T}(w(n)))$$

où d est une fonction de distance.

En reconnaissance des gestes, cette technique consiste à comparer un geste à tester avec des *templates*, ou gestes de références. La classe du geste à tester correspond à la

classe du geste de référence qui a la distance la plus faible avec le geste à tester. Chaque classe est donc représentée par un *template*, ce qui ne permet pas de prendre en compte une grande variabilité dans l'exécution des gestes. Une solution pourrait être de créer un *template* par geste de la base d'apprentissage puis de procéder par un système de vote pour classer un nouveau geste.

Gillian et al. [44] proposent une méthode, appelée ND-DTW, pour calculer le chemin optimal entre deux signaux à N-dimensions. Une première utilisation pour la reconnaissance des gestes a été faite par Gavrila et Davis [43]. Après avoir déterminé la posture 3D d'une personne portant des marqueurs, ils utilisent les angles 3D des articulations comme caractéristiques pour reconnaître des gestes grâce aux DTW. Liu et al. [86] utilisent les DTW pour reconnaître des actions en utilisant des données issues d'accéléromètres. Pour permettre au système de s'adapter à l'utilisateur, ils permettent de modifier le geste "modèle" par une mise à jour de celui ci lorsqu'il est mal reconnu. Sempena et al. [115] reconnaissent des gestes comme "taper des mains", "boxer" et "courir" en utilisant des DTW. Chaque geste modèle est décrit par une suite de quaternions représentant les angle de chacune des 15 articulations issues du squelette calculé grâce au SDK de la Kinect. Reyes et al [107] utilisent également des données issues de la Kinect pour reconnaître des gestes avec des DTW. Ils utilisent directement les positions 3D des articulations. Pour chaque DTW, les positions des articulations sont pondérées en fonction de leur impact dans le geste effectué. Cette méthode permet d'augmenter jusqu'à 13% le taux de reconnaissance par rapport à un DTW classique, c'est à dire sans pondération.

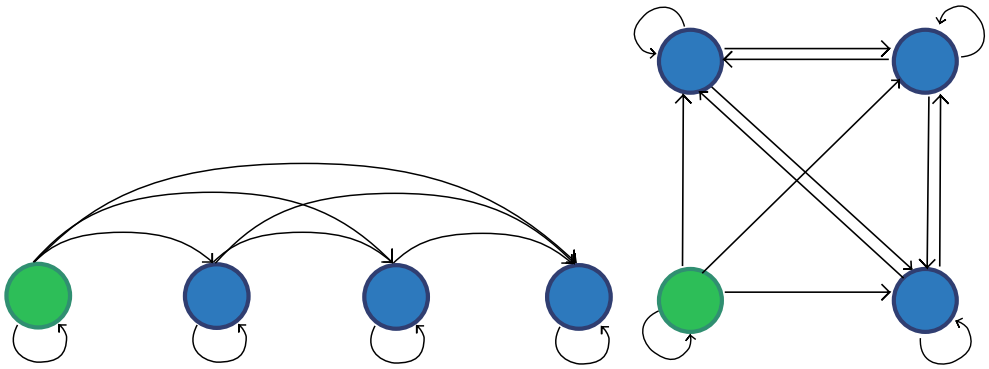
3.6.2.2 Modèles de Markov Cachés

Les modèles de Markov Cachés, ou *Hidden Markov Models* (HMM) sont couramment utilisés pour la reconnaissance de gestes car ils permettent, comme les DTW, de prendre en compte la dimension temporelle dans l'exécution d'un geste. Ils ont dans un premier temps été introduits pour la reconnaissance de la parole ou de l'écriture, Hu et al. [55]. Un célèbre tutoriel a été écrite par Rabiner [104] en 1989. D'autres méthodes proches des HMMs sont également utilisées pour la reconnaissance de gestes : les HCRF [123] (Hidden Conditional Random Fields) et les DBN [122] (Dynamic Bayesian Network) par exemple, mais ne seront pas détaillées dans ce manuscrit.

Présentation générale Les HMMs sont des systèmes probabilistes à états finis qui reposent sur le principe de dépendance entre deux observations successives. Ils découlent des chaînes de Markov qui permettent de modéliser l'évolution dynamique d'un système aléatoire. La propriété fondamentale des chaînes de Markov est que l'évolution future du système ne dépend que de l'état actuel dans lequel il se trouve. Il s'agit de la propriété dite *markovienne*. Les chaînes de Markov ont de nombreuses applications comme la génétique des populations ou les mathématiques financières.

Dans le cas des HMMs, les états du système ne sont pas directement visibles. En revanche, à chaque état sont associées des observations visibles. On infère donc l'état actuel dans lequel le système se trouve en fonction d'observations, qui suivent une loi de probabilité associée à l'état courant. Les observations peuvent être discrètes. Dans ce cas, pour chaque état caché, on aura la probabilité d'émettre chaque observation ; ou elles peuvent être continues, dans ce cas on associe une fonction de densité à chaque état, Figure 3.23. Dans la suite de ce manuscrit, nous nous concentrerons sur la cas des HMMs discrets.

Un HMM se définit par sa structure et ses paramètres. L'ensemble se note $\lambda = (\mathscr{A}, \mathscr{B}, \Pi)$. Les éléments $\mathscr{A}$, $\mathscr{B}$ et Π sont définis ci-dessous. La structure d'un HMM se compose du



(a) (b)

FIGURE 3.23 – Exemples de structures ed HMMs pour des HMMs à quatre états cachés, en vert l'état initial, les flèches représentent les transitions possibles : (a) : structure gauche-droite, (b) : structure ergodique

nombre d'états cachés et de l'ensemble des transitions possibles.Deux types de structures existent, la structure ergodique : tous les états sont reliés ; et la structure gauche-droite : seules les transitions permettant d'aller dans un état supérieur ou de rester dans le même état sont possibles. Pour la reconnaissance de gestes ou d'actions, la structure gauche-droite est préférée pour mieux modéliser la dimension temporelle d'une action.

Un HMM est donc caractérisé par :

— Le nombre N d'états dans le modèle : $S = \{S_1, S_2, ...S_N\}$. L'état dans lequel se trouve le système à l'instant t se note q_t

— Le nombre M de symboles pouvant être émis à chaque état : $V = \{V_1, V_2, ...V_M\}$

— La probabilité de transition d'états $\mathscr{A} = \{a_{ij}\}$ telle que :

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i \leq N \tag{3.5}$$

— Les probabilités d'observation d'un symbole k à l'état j, $\mathscr{B} = \{b_j(k)\}$, aussi appelée loi de probabilité d'émission, avec

$$b_j(k) = P[v_k \; at \; t | q_t = S_j], \quad 1 \leq i \leq N, \quad 1 \leq j \leq M \tag{3.6}$$

— Les probabilités initiales de distribution des états $\Pi = \{\pi_i\}$ avec

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N \tag{3.7}$$

Une notation compacte d'un HMM est $\lambda = (\mathscr{A}, \mathscr{B}, \Pi)$. Une illustration des HMMs peut être vue Figure 3.24.

Utilisation des HMM L'utilisation des HMMs nécessite la résolution de trois problèmes :

— *problème 1, évaluation des HMMs (reconnaissance)* : Ayant une séquence d'observations $O = O_1 O_2 ... O_T$ et un modèle $\lambda = (\mathscr{A}, \mathscr{B}, \Pi)$, comment calculer la probabilité que que le HMM ait généré cette suite d'observations ?,

— *problème 2, trouver la suite d'états cachés la plus probable :* Ayant une séquence d'observations $O = O_1 O_2 ... O_T$ et un modèle $\lambda = (\mathscr{A}, \mathscr{B}, \Pi)$, quelle est la suite d'états $Q = q_1 q_2 ... q_T$ qui justifie le mieux la suite d'observations ?

— *problème 3, apprentissage des HMMs :* Comment ajuster le modèle $\lambda = (\mathscr{A}, \mathscr{B}, \Pi)$ qui maximise $P(O|\lambda)$?

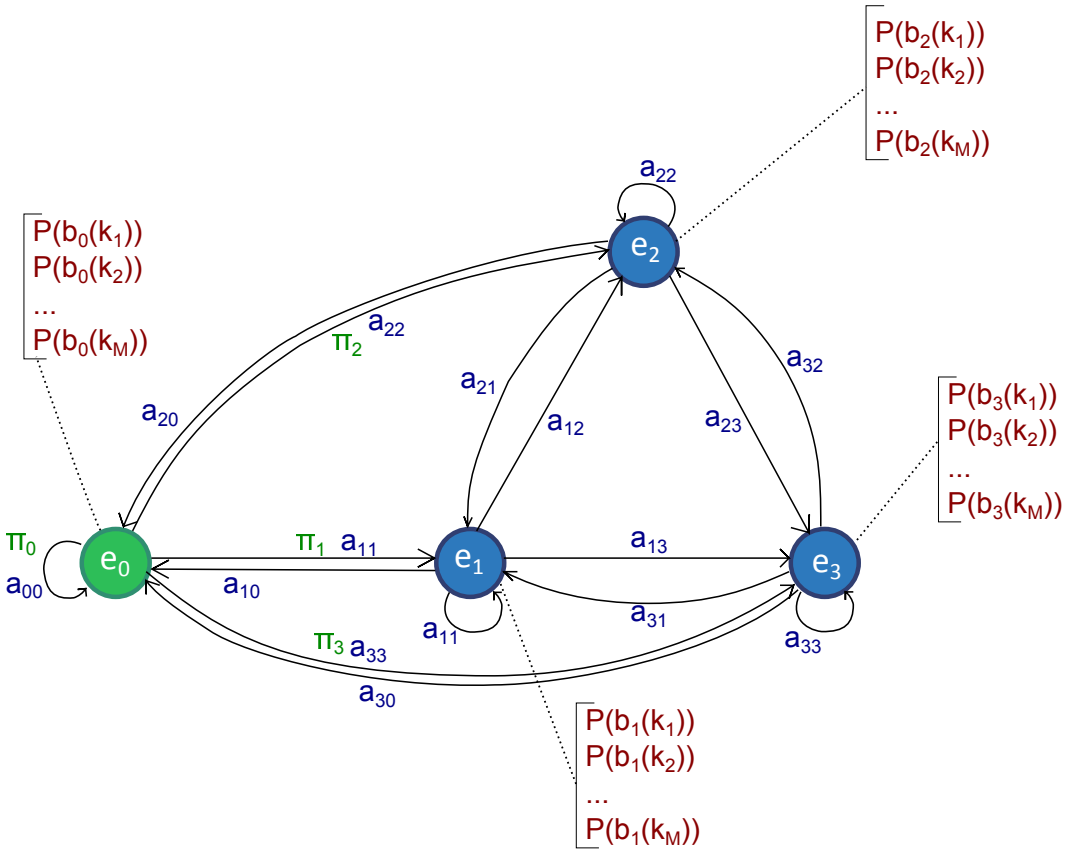


FIGURE 3.24 – Illsutration d'un HMM à quatre états cachés

Résolution du problème 1, reconnaissance : La résolution de ce problème peut se faire en utilisant l'algorithme Forward, décrit dans le tutoriel de Rabiner [104]. Calculer la probabilité $P(O|\lambda)$ revient à calculer la probabilité d'avoir obtenu la suite s'observations O selon tous les chemins possibles entre états. Néanmoins, ce calcul nécessite N^T opérations, ce qui peut vite devenir très gourmand. On introduit donc la variable *forward* $\alpha_t(i)$ telle que :

$$\alpha_t(i) = P(O_1 O_2 .. O_t, q_t = S_i|\lambda) \tag{3.8}$$

Elle est calculée récursivement, telle que

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1}) \tag{3.9}$$

et on obtient

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \tag{3.10}$$

Le nombre d'opérations n'est alors plus que de N^2T. La résolution de ce problème peut être perçue comme une méthode pour évaluer plusieurs HMMs, en calculant la probabilité pour chacun d'avoir généré une suite d'observations. C'est donc la résolution de ce problème qui est le plus pertinent pour la reconnaissance des gestes. Si on a un ensemble de gestes, on commence par entraîner un HMM par geste. Lors de l'acquisition de nouvelles observations, pour savoir quel geste a le plus probablement été effectué, il suffit de résoudre le problème 1, c'est à dire appliquer l'algorithme *forward*, à tous ces HMMs pour en déduire quel est celui qui aurait le plus probablement généré cette suite d'observations.

De manière analogue à la variable *forward*, nous pouvons introduire la variable *backward* $\beta_t(i)$ telle que

$$\beta_t(i) = P(O_{t+1}O_{t+2}..O_T, q_t = S_i|\lambda) \tag{3.11}$$

Elle représente la probabilité d'observer une suite d'observation à partir du temps $t+1$ jusqu'à la fin des observations. Cette variable se calcule aussi de manière itérative et est utilisée pour la résolution des problèmes 2 et 3.

Résolution du problème 2, trouver la suite d'états la plus probable : Pour résoudre le problème 2, c'est l'algorithme de Viterbi qui est utilisé, également décrit dans le tutoriel de Rabiner [104]. Considérant une suite d'observations, l'algorithme de Viterbi a pour but de trouver la suite d'états la plus probable l'ayant produit. Il faut donc évaluer :

$$\delta_t(i) = \max_{q_1,q_2,...,q_{t-1}} P(q_1, q_2, ..., q_t = i, O_1, O_2, ..., O_t|\lambda) \tag{3.12}$$

Par induction nous avons :

$$\delta_{t+1}(i) = [\max_i \delta_t(i).a_{ij}].b_j(O_{t+1}) \tag{3.13}$$

Pour chaque observation, on garde en mémoire les paramètres qui maximisent $\delta_{t+1}(j)$ dans un tableau $\psi_t(j)$, tel que :

$$\psi_t(j) = \arg\max_{1\leq i\leq N} [\delta_{t-1}(i).a_{ij}] \tag{3.14}$$

Le chemin optimal $q_1^*, q_2^*, ..., q_T^*$ est déterminé en parcourant par *backtracking* ce tableau.

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, ..., 1. \tag{3.15}$$

Résolution du problème 3, apprentissage : La résolution du problème 3 revient à effectuer l'étape d'apprentissage du HMM. L'algorithme de Baum-Welch, décrit dans [104] peut être utilisé dans ce but. Il permet de déterminer λ qui maximise localement $P(O|\lambda)$. Pour résoudre ce problème la grandeur $\xi_t(i,j)$ qui représente la probabilité d'être à l'état S_i au temps t et à l'état S_j au temps t+1. Elle peut être décrite grâce aux variables *forward* $\alpha_t(i)$ et *backward* $\beta_t(i)$ introduites ci-dessus. Nous avons également la probabilité$\gamma_t(i)$ d'être dans l'état S_i au temps t qui peut être donc être évaluée par :

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3.16}$$

On peut déduire de ces deux grandeurs les informations suivantes :

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{nombre attendu de transitions depuis l'état } S_i \tag{3.17}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{nombre attendu de transitions depuis l'état } S_i \text{ vers l'état } S_j \tag{3.18}$$

Grace à ces grandeurs, nous pouvons mettre à jour les paramètres du modèle $\lambda = (\mathscr{A}, \mathscr{B}, \Pi)$ par les formules suivantes :

$$\overline{\pi}_i = \text{fréquence attendue d'état } S_i \text{ au temps (t=1)} = \gamma_1(i) \tag{3.19}$$

$$\overline{a}_{ij} = \frac{\text{nombre attendu de transitions depuis l'état } S_i \text{ vers l'état } S_j}{\text{nombre attendu de transitions depuis l'état } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (3.20)$$

$$\overline{b}_j(k) = \frac{\text{fréquence attendue d'états } S_j \text{ en observant } v_k}{\text{fréquence attendue d'états } S_j} = \frac{\sum_{\substack{t=1 \\ O_t=v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (3.21)$$

La mise à jour des paramètres est faite de manière itérative. On calcule après chaque mise à jour la probabilité $P(O|\overline{\lambda})$ pour évaluer son amélioration. Si cette amélioration atteint un seuil prédéfini, le processus itératif est arrêté.

Cas particulier des HMMs discrets : une première étape de discrétisation Lors de l'utilisation d'HMMs discrets, une première étape consistant à discrétiser les données est nécessaire. En effet, contrairement aux HMMs continus où pour chaque état caché nous avons une fonction de densité associée, pour les HMMs discrets nous avons la probabilité pour chaque observation d'être émise par l'état caché. Il nous faut donc des observations discrètes. Pour la reconnaissance de gestes, nous disposons, la plupart du temps, d'une suite d'observations continue. Il faut donc dans un premier temps partitionner ces données avant de pouvoir les utiliser pour apprendre les HMMs ou reconnaître le geste qui a été exécuté.

Plusieurs algorithmes non supervisés permettent le partitionnement automatique de ces données. Un des plus utilisés est l'algorithme appelé K-Means, ou K-moyennes, voir figure 3.25.

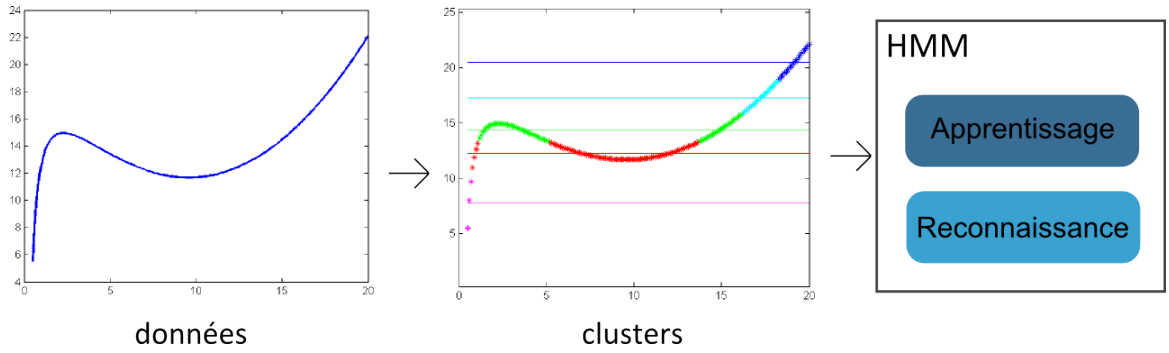


FIGURE 3.25 – Illsutration d'un K-Means sur des données à une dimension

L'algorithme des K-Means permet de diviser un ensemble de données en k partitions, ou *clusters*, en minimisant, pour chaque *cluster*, la distance moyenne des points le composant à son barycentre. La détermination des k centres de gravité se fait pas un calcul itératif dont les étapes sont détaillées ci dessous.

— **Etape 0** : Choix aléatoire de k centres parmi l'ensemble des données de départ

— **Etape 1** : Pour chaque donnée, on lui associe un, et un seul, centre (ou partition). Le choix du centre est fait en cherchant la distance minimale de la donnée à chacun des centres.

— **Etape 2** : On calcule le centre de gravité de chacune des partitions.

Les étapes 1 et 2 sont répétées jusqu'à convergence ou lorqu'un nombre maximum d'itérations a été atteint.

Le choix des *clusters*, c'est à dire l'entrainement des K-Means, se fait sur l'ensemble de la base de données d'apprentissage. Chaque *cluster* est représenté par son barycentre.

Pour la reconnaissance de gestes, après discrétisation, chaque geste est donc une suite de valeurs discrètes (une valeur = un barycentre). Pour chaque état caché, il y a autant d'observations possibles qu'il y a de *clusters*. Une probabilité d'observation représente donc la probabilité d'observer ce cluster dans cet état caché.

Les HMMs pour la reconnaissance de gestes Les HMMs ont été utilisés à de nombreuses reprises pour la reconnaissance de gestes. Nous allons en citer quelques études ci-dessous.

En 1992, Yamato et al. [141] ont utilisé des HMMs pour reconnaître des actions dans une suite d'images. Ils extraient des descripteurs qu'ils discrétisent en utilisant un algorithme de *clustering*, puis utilisent les centres des *clusters* pour entrainer leurs HMMs.

Les HMMs ont également été utilisés par Achard et al. [1] pour reconnaître des actions à partir de silhouettes. Liu et al. [88] ont comparé les HMMs de structure gauche-droite et les HMM ergodiques pour la reconnaissance de gestes de la main dans des vidéos. Le suivi des mains se fait en appliquant une segmentation de la couleur de la peau dans des images. Ils obtiennent en moyenne 10% de meilleurs reconnaissances en utilisant des HMMs gauche-droite qu'en utilisant des HMMs ergodiques. Zhu et Pun [143] ont utilisé des HMMs pour reconnaître des gestes de la main en temps réel avec des images de profondeur. Ils utilisent OpenNi pour suivre les mains et se servent de leurs positions comme descripteurs des gestes.

Xia et al. [140] ont classés des gestes avec des HMMs en utilisant également une caméra de profondeur. Ils se sont intéressés à des gestes exécutés par le corps entier et utilisent l'algorithme de Shotton et al. [117] pour squelettiser la personne filmée. Ils utilisent un histogramme représentant le position de chaque articulation dans un repère sphérique centré autour du milieu des hanches de la personne filmée pour décrire les postures composant les gestes. L'ensemble des histogrammes est ensuite partitionné en K clusteurs grâce à l'algorithme des K-Means, puis l'apprentissage et la reconnaissance sont effectués avec des HMMs. Le geste le moins bien reconnu est le geste "*jeter*" car dans cette position la personne ne fait plus face à la caméra et donc la squelettisation est surement moins précise.

Calinon et Billard [23] ont proposé d'utiliser la reconnaissance de gestes, avec des HMMs, pour permettre aux robots d'apprendre des gestes en observant des exemples. Ils proposent une méthodologie pour reproduire des gestes appris. Les auteurs utilisent des capteurs inertiels fixé à l'utilisateur/le démonstrateur pour capturer le geste, la reconnaissance se fait avec l'algorithme Forward-Backward. L'algorithme de Viterbi est ensuite utilisé pour générer à nouveau des gestes.

Dans Lee et al. [80] les auteurs gèrent les gestes inattendus grâce à un modèle de seuil adaptatif. Chaque geste appris est représenté par un HMM gauche-droite. Un modèle de seuil est en même temps construit. Il s'agit d'un HMM ergodique regroupant tous les états des HMMs des gestes. Le principe du seuillage adaptatif repose sur la comparaison des sorties des HMMs gauche-droite avec le HMM ergodique. En effet, lorsqu'un geste appris est effectué, le HMM gauche-droite le représentant a une probabilité de générer ce geste supérieure à la probabilité en sortie du HMM ergodique. Par contre, si un geste inattendu est effectué, la probabilité en sortie de l'HMM ergodique sera supérieure. La comparaison des deux probabilités permet donc d'écarter, ou non, un geste.

Tang et al. [124] proposent un HMM à durée variable. Ce HMM permet à chaque état caché de pouvoir générer plusieurs observations. Chaque état représente alors une sous-partie de l'action complète.

3.7 Conclusion sur la reconnaissance de gestes

Dans le chapitre précédent nous avons mis en évidence les problématiques de la collaboration homme-robot en milieu industriel. Une des principales limitations est la difficulté de communiquer entre les deux partenaires et de synchroniser leurs tâches respectives. Nous avons pensé à la reconnaissance de gestes pour répondre à cette problématique.

Dans ce chapitre, nous avons présenté plusieurs méthodes utilisant différents capteurs : les caméras RGB, les caméras de profondeur et les capteurs inertiels. Bien que les caméras soient sensibles aux occultations, elles permettent d'avoir d'avoir une meilleure connaissance de la scène et donc de l'environnement du robot. L'utilisation de caméras de profondeur permet de s'émanciper des changements de luminosité et d'avoir des informations plus directes sur les mouvements de l'opérateur. Les capteurs inertiels peuvent être une contrainte pour l'opérateur qui doit les porter en plus d'effectuer ses tâches. Néanmoins, ils peuvent donner des informations fiables sur les mouvement des membres du corps. Ils peuvent servir dans une première étape d'étude de faisabilité de reconnaissance de gestes dans un cas d'étude particulier, avant d'essayer d'obtenir des informations semblables via des capteurs de vision.

Pour la classification des gestes, prendre en compte la dimension temporelle permet de mieux modéliser des gestes complexes. Pour rendre compte de la grande variabilité d'exécution d'un même geste d'un opérateur à l'autre, l'utilisation de HMMs semble plus adaptée.

Chapitre 4

Reconnaissance des gestes techniques avec un équipement intrusif

Sommaire

4.1 Protocole d'acquisition de la base de données 69
4.2 Méthodologie de reconnaissance des gestes 70
4.2.1 Présentation du *Gesture Follower* 70
4.2.2 Reconnaissance des gestes isolés 71
4.2.3 Reconnaissance en temps réel 72
4.3 Résultats 72
4.3.1 Reconnaissance de gestes isolés 72
4.3.2 Reconnaissance en temps réel 73
4.4 Conclusion de l'étude 76

Dans ce chapitre nous allons décrire une méthode pour la reconnaissance des gestes techniques avec des capteurs intrusifs. Nous allons nous concentrer sur le cas d'étude de co-présence, présenté partie 2.2.1.

4.1 Protocole d'acquisition de la base de données

Comme expliqué dans les parties 2.2.1.2 et 2.2.1.3 , nous avons à notre disposition deux types de capteurs, les MotionPods et l'Animazoo. Nous avons également choisi quatre gestes à reconnaître :

geste 1 : enlever l'adhésif de la feuille d'étanchéité (**G1**)

geste 2 : poser la feuille d'étanchéité sur la porte (**G2**)

geste 3 : pré-coller la feuille d'étanchéité sur la porte (**G3**)

geste 4 : mettre le lécheur (**G4**)

Nous avons enregistré un opérateur effectuant ces quatre gestes 5 fois avec chacun des capteurs. Nous utilisons les angles d'Euler pour décrire les gestes. Nous pouvons voir Figure 4.1 le temps moyen d'exécution de ces quatre gestes.

Mis à part le geste G1, la durée des gestes est inférieure ou égale à 10 secondes et varie peu d'une exécution à l'autre. Le geste G1 est le geste le plus long, 11,9 secondes en moyenne, avec une plus grande variabilité temporelle, l'écart type étant de 3.5 secondes. Cette variabilité dans le temps d'exécution du geste G1 peut s'expliquer par l'utilisation

de différentes méthodes pour enlever l'adhésif de la feuille d'étanchéité. En fonction des zones initiales du décollement, l'adhésif s'enlève plus ou moins facilement.

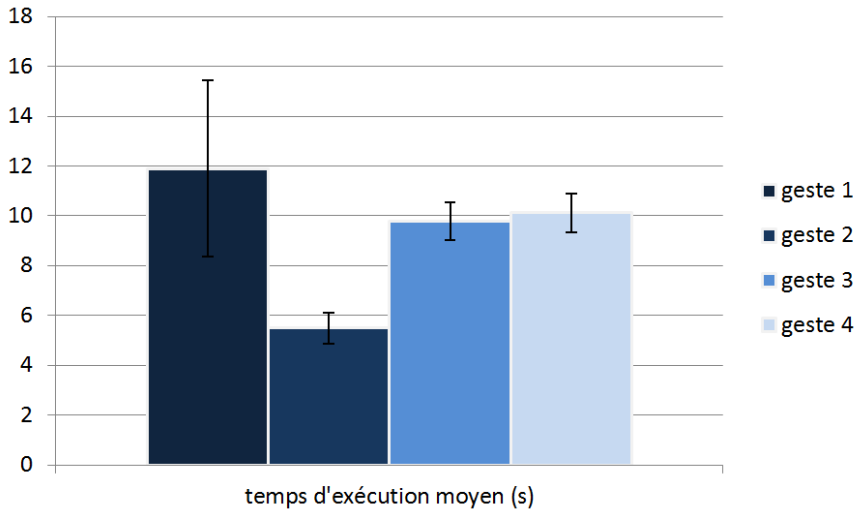


FIGURE 4.1 – Temps moyen d'exécution des gestes dans le cas d'étude de la co-présence

4.2 Méthodologie de reconnaissance des gestes

4.2.1 Présentation du *Gesture Follower*

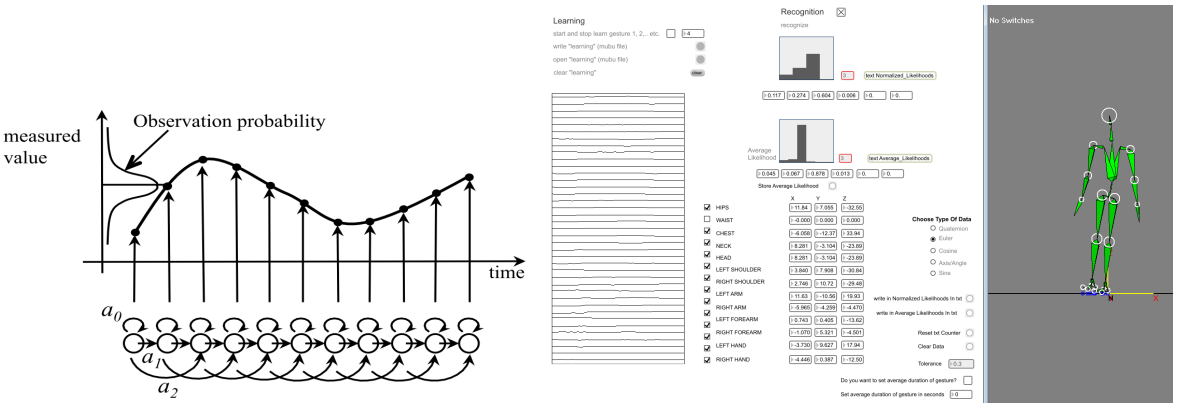
Nous avons utilisé le *Gesture Follower*[1], programme développé sous Max/MSP[2] pour la reconnaissance de gestes en temps réel. Son fonctionnement est décrit par Bevilacqua et al., [11] et [12]. Le *Gesture Follower* permet d'apprendre des HMMs avec très peu d'exemples. En effet, un seul exemple, ou modèle, est utilisé pour apprendre un HMM. Chaque échantillon du geste modèle est utilisé pour créer un état caché auquel est associé une probabilité d'observation sous la forme d'une distribution normale centrée autour de la valeur de l'échantillon à l'origine de l'état caché. Seules les transitions permettant de rester dans le même état ou d'aller dans un des deux supérieurs sont autorisées.

La Figure 4.2a illustre la construction d'un HMM par le *Gesture Follower*. On peut voir une interface Max/MSP utilisant le *Gesture Follower* en Figure 4.2b. Le *Gesture Follower* est simple d'utilisation. Nous avons décidé de l'utiliser pour reconnaître les gestes du cas d'étude de co-présence. Il permet d'avoir rapidement une idée sur la faisabilité de la reconnaissance de gestes techniques. De plus, puisque nous ne disposons pas d'un grand nombre d'exemples de gestes, cette méthode nous semble appropriée.

Le *Gesture Follower* fournit deux informations lors de la reconnaissance d'un nouveau geste :

- la probabilité pour chaque classe, lorsque le geste est terminé, qu'il appartienne à la classe,
- lors de l'arrivée de chaque nouvel échantillon du geste à reconnaître, la probabilité, pour chaque classe, que le début du geste appartienne à la classe.

1. http ://ismm.ircam.fr/gesture-follower/
2. https ://cycling74.com/



(a) (b)

FIGURE 4.2 – Le *Gesture Follower*. (a) : Construction d'un HMM par le *Gesture Follower*, (©Bevilacqua et al.[12]), (b) : Interface graphique du *Gesture Follower* avec une visualisation du squelette de l'Animazoo

Nous allons utiliser la première information pour faire une reconnaissance des gestes isolés. C'est à dire, après avoir enregistré et segmenté les gestes, on effectue une reconnaissance geste par geste. Cette méthode sera décrite dans la partie 4.2.2. Nous allons aussi faire une reconnaissance en temps réel en utilisant la seconde information, ceci sera détaillé dans la partie 4.2.3. Nous allons y traiter les données en temps réel afin d'informer le robot le plus tôt possible sur le geste qui a été, ou qui est en train d'être effectué.

4.2.2 Reconnaissance des gestes isolés

Pour la reconnaissance des gestes isolés, la méthode du jackknife, aussi appelée "*leave one out*", a été utilisée pour évaluer notre processus de reconnaissance. Comme nous n'avons besoin que d'un exemple de chaque geste pour entraîner le *Gesture Follower*, nous testons toutes les combinaisons possibles de "1 enregistrement des quatre gestes pour l'apprentissage, N-1 enregistrements pour la reconnaissance".

Nous avons utilisé la précision et le rappel qui sont définis comme ci-dessous :

$$\text{precision} = \frac{\#(\text{vrai positif})}{\#(\text{vrai positif}) + \#(\text{faux positif})} \tag{4.1}$$

$$\text{rappel} = \frac{\#(\text{vrai positif})}{\#(\text{vrai positif}) + \#(\text{faux négatif})} \tag{4.2}$$

La précision et le rappel sont calculés pour chacun des quatre gestes. Pour un geste de classe i,

- #(vrai positif) représente le nombre de gestes de classe i correctement reconnus,
- #(faux positif) représente le nombre de gestes qui ne sont pas de classe i et qui ont reconnus comme étant de classe i
- #(faux négatif) représente le nombre de gestes de classe i qui n'ont pas été reconnus comme étant de classe i

En d'autres termes, la précision représente le taux de gestes qui sont réellement de classe i parmi tous ceux qui sont reconnus comme étant de classe i. Le rappel représente le taux de gestes de classe i qui ont été reconnus comme étant de classe i.

4.2.3 Reconnaissance en temps réel

Pour la reconnaissance en temps réel, nous utilisons les enregistrements complets où les quatre gestes sont effectués les uns à la suite des autres. Nous voulons reconnaître les gestes le plus tôt possible pour pouvoir transférer l'information au robot.

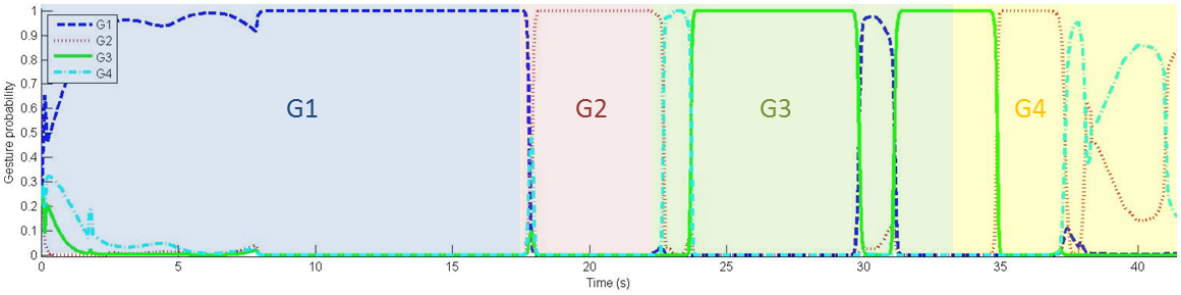


FIGURE 4.3 – Exemples de données brutes issues du *Gesture Follower* pour la reconnaissance en temps réel. Probabilité, pour chaque classe, que le geste exécuté lui appartienne.

La Figure 4.3 illustre des données que nous obtenons avec le *Gesture Follower*. Les couleurs en arrière plan représentent la vérité de terrain, c'est à dire le geste effectivement exécuté par l'opérateur à chaque instant (bleu pour le geste G1, rouge pour le geste G2, vert pour le geste G3 et jaune pour le geste G4). Ce code couleur est repris pour les Figures de la partie 4.3.2. Chaque courbe représente la probabilité que le geste appartienne à chaque classe apprise. La somme des probabilités est égale à 1 à chaque pas de temps.

Nous pouvons observer sur cet exemple que les gestes sont en général correctement reconnus. Il y a des fluctuations lors du passage d'un geste à un autre et quelques fausses reconnaissances, principalement lorsque le geste G4 est exécuté.

Nous souhaitons transmettre au robot des informations fiables pour éviter d'induire en erreur son fonctionnement lors de la réalisation de ses tâches. Nous avons décidé de ne prendre en compte que les probabilités supérieures à un certain seuil. Nous avons également utilisé une fenêtre temporelle glissante. Nous prenons en compte le geste qui a le plus de fois une probabilité maximale dans cette fenêtre.

4.3 Résultats

4.3.1 Reconnaissance de gestes isolés

Nous avons utilisé des données provenant des MotionPods et des données provenant de l'Animazoo.

Nous pouvons voir sur le Tableau 4.1 les résultats avec les MotionPods. 86% des gestes sont correctement reconnus. Les gestes G2 et G4 sont parfois confondus, cela peut s'expliquer par le fait que pour ces deux gestes l'opérateur a les bras tendus vers l'avant avec peu de rotation au niveau des poignets. Bien que 86% de bonnes reconnaissances est un bon résultat, cela n'est pas suffisant en milieu industriel.

Le Tableau 4.2 donne les résultats obtenus avec la veste Animazoo. Avec ces capteurs, 96% des gestes sont correctement reconnus, 10% de plus que ce que nous avions obtenus avec les Motion Pods. Le geste G4 est également confondu avec le geste G2, cela peut être expliqué pour les raisons identiques que pour les MotionPods. 96% de reconnaissances correctes est un très bon résultat, néanmoins il doit être confirmé par une reconnaissance en temps réel pour que cette méthode puisse être utilisée pour la collaboration avec un robot.

TABLEAU 4.1 – Résultats avec les MotionPods

		Sortie (Probabilité maximale)				
		G1	G2	G3	G4	Rappel
Entrée	G1	**18**	1	1	-	90%
	G2	1	**16**	1	2	80%
	G3	-	-	**20**	-	100%
	G4	1	3	1	**15**	75%
	Précision	90%	80%	87%	88%	**86%**

TABLEAU 4.2 – Résultats avec l'Animazoo

		Sortie (Probabilité maximale)				
		G1	G2	G3	G4	Rappel
Entrée	G1	**20**	-	-	-	100%
	G2	-	**20**	-	-	100%
	G3	-	-	**20**	-	100%
	G4	-	3	-	**17**	85%
	Précision	100%	87%	100%	100%	**96%**

Ces résultats confirme que la reconnaissance des gestes techniques est possible. Nous pouvons observer que plus nous disposons d'information (12 capteurs inertiels pour l'Animazoo contre 2 pour les MotionPods), plus le système semble robuste.

4.3.2 Reconnaissance en temps réel

Nous souhaitons obtenir un système qui détecte rapidement quel geste est effectué, mais nous souhaitons également limiter les *faux positifs* pour ne pas induire le robot en erreur. En d'autres termes, nous préférons privilégier la précision par rapport au rappel. Nous avons fait varier plusieurs paramètres afin de traiter les données de reconnaissance de gestes en temps réel. Comme expliqué dans la partie 4.2.3, nous utilisons une fenêtre glissante et nous cherchons le geste qui a le plus grand nombre de probabilités maximales, en ne prenant en compte que les probabilités qui sont au dessus d'un certain seuil. Nous pouvons donc faire varier la largeur de la fenêtre glissante et le seuil au dessus duquel les probabilités sont prises en compte.

La Figure 4.4 illustre l'utilisation de seuils sur l'exemple de la Figure 4.3. Nous pouvons y voir les gestes ayant la probabilité maximale à chaque instant (courbes bleues) et après seuillage, en ne prenant que les probabilités au dessus d'un certain seuil (courbes rouges).

Lorsque la probabilité maximale est au-dessous du seuil choisi, la sortie vaut "0", c'est à dire qu'aucun geste n'est reconnu. Grâce à l'application d'un seuil, les petites erreurs lors des changements de gestes semblent être évitées, comme lors du passage du geste G1 au geste G2. Néanmoins, lors d'erreurs de reconnaissance plus importantes, comme lors de l'exécution des gestes G3 et G4 sur cet exemple, l'application d'un seuil ne permet pas de les corriger.

Dans un second temps, comme expliqué ci-dessus, nous utilisons une fenêtre glis-

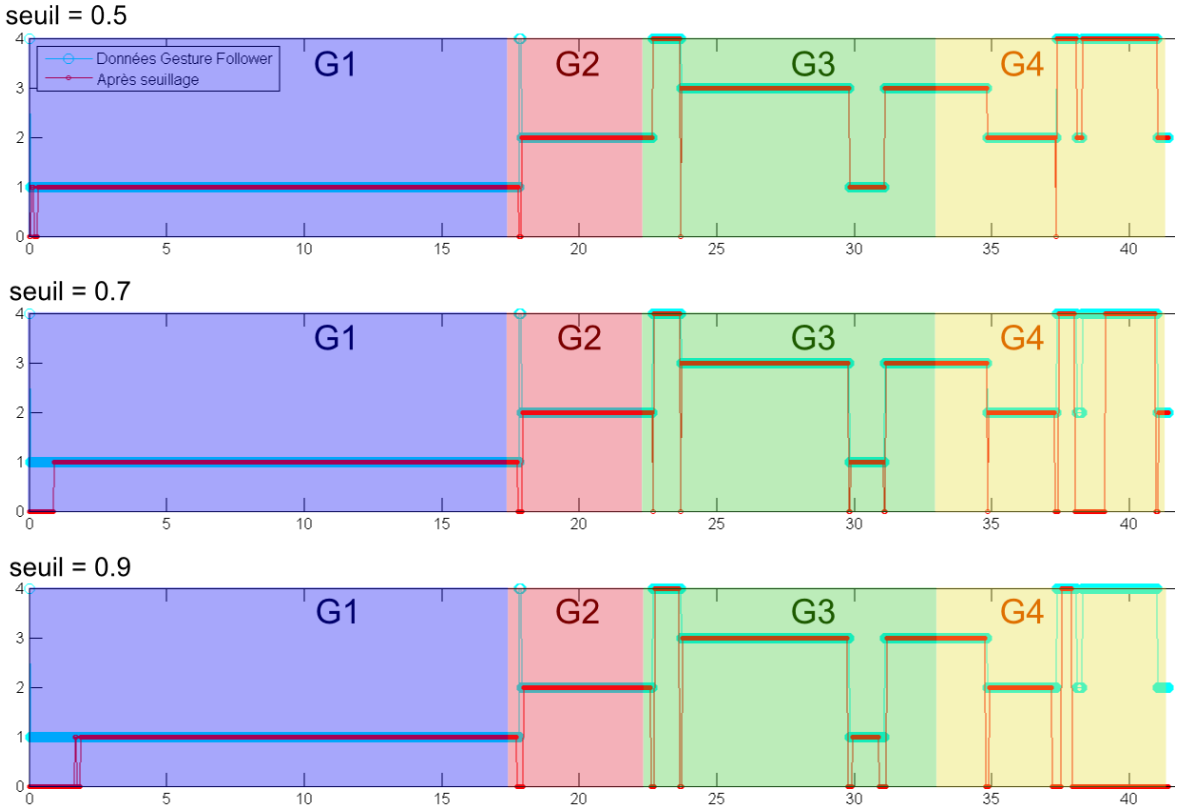


FIGURE 4.4 – Application de différents seuils pour la reconnaissance de gestes. En bleu : sortie du *Gesture Follower*. En rouge, geste ayant la plus grande probabilité, en ne prenant en compte que les probabilités supérieures à un certain seuil. Haut : seuil = 0.5, milieu : seuil= 0.7, bas : seuil = 0.9. En ordonnée : le numéro du geste reconnu.

sante de manière à ne prendre en compte que le geste qui a le plus souvent été reconnu lors des derniers instants. Nous devons choisir la largeur de cette fenêtre glissante. Une fenêtre trop large éliminerait les gestes qui sont correctement reconnus que lors de courts instants, tandis qu'une fenêtre moins large pourrait ne pas être assez stricte pour éviter les fausses reconnaissances. Aussi, plus une fenêtre est large, plus elle induit un retard dans la reconnaissance.

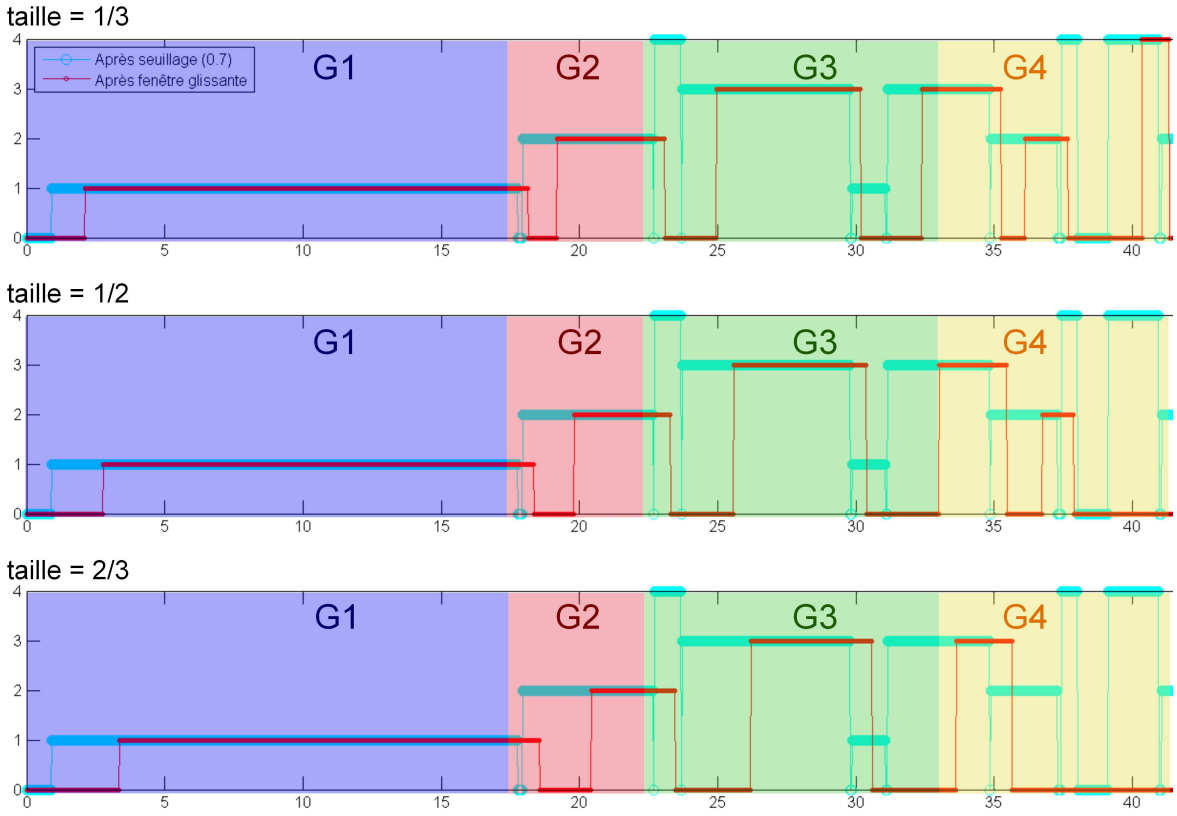


FIGURE 4.5 – Utilisation de fenêtres glissantes de tailles variables pour la reconnaissance de gestes en temps réel. En haut : 1/3 de la durée moyenne du geste le plus court, milieu : 1/2, bas : 2/3. En ordonnée : le numéro du geste reconnu.

Pour être sûr de permettre la reconnaissance de tous les gestes, nous avons choisi des tailles de fenêtre inférieures à la durée moyenne du geste le plus court, le geste G2 dans notre cas d'étude. Nous avons testé différentes largeurs, égales à 1/3, 1/2 et 2/3 de cette durée moyenne. La Figure 4.5 illustre l'utilisation d'une fenêtre glissante sur le même exemple, avec des données précédemment seuillées à 0.7. Nous pouvons y voir en bleu le geste ayant la probabilité maximale en ne prenant en compte que celles au dessus du seuil. En rouge, nous avons à chaque instant le geste ayant le plus grand nombre de probabilités maximales dans la fenêtre glissante. Dans l'exemple présenté en Figure 4.5, la majorité des fausses reconnaissances sont évitées dès que la fenêtre est égale à 1/3 de la durée moyenne du geste le plus court. Cependant, les erreurs de reconnaissance pour le geste G4 sont trop importantes. Elles ne sont évitées qu'avec la fenêtre la plus large, mais dans ce cas là, le geste G4 n'est pas non plus reconnu. Dans le cas où l'on souhaite un système qui privilégie la précision, une fenêtre large semblerait plus adaptée.

Nous avons utilisé la méthode du jackknife décrite dans la partie 4.3.1 pour évaluer la précision et le rappel pour chacun des gestes sur l'ensemble de notre base de données. Nous les avons évalués pour différentes tailles de fenêtres et différents seuils. Nous pouvons voir les résultats sur les Tableaux 4.3 et 4.4. Comme nous pouvions nous y attendre,

plus le seuil est strict (proche de 1), plus le rappel est bas. De plus, lorsque nous augmentons la taille de la fenêtre, la précision augmente tandis que le rappel baisse. En regardant les résultats plus en détail, on peut observer que la précision du geste G2 est faible dans toutes les configurations, de même que les rappels des gestes G2 et G4. Notre analyse précédente expliquait que ces deux gestes peuvent être confondus car ils semblent similaires : peu de rotations au niveau des poignets, les bras tendus vers l'avant. Les capteurs inertiels ne sont donc pas les capteurs les mieux adaptés pour les discerner. La meilleure précision moyenne sur les quatre gestes que nous obtenons est une précision de 66%, ce qui reste faible pour une application industrielle.

TABLEAU 4.3 – Précision et rappel pour chaque geste, seuls les gestes avec une probabilité d'au moins 0.7 sont prises en compte

	Précision					Rappel				
	G1	G2	G3	G4	moyenne	G1	G2	G3	G4	moyenne
fenêtre 1/3	0.62	0.27	0.83	0.64	**0.59**	1	0.56	0.95	0.80	**0.83**
fenêtre 1/2	0.71	0.33	0.86	0.65	**0.64**	1	0.5	0.9	0.65	**0.76**
fenêtre 2/3	0.83	0.27	0.78	0.75	**0.66**	0.95	0.33	0.78	0.45	**0.63**

TABLEAU 4.4 – Précision et rappel pour chaque geste, seuls les gestes avec une probabilité d'au moins 0.9 sont prises en compte

	Précision					Rappel				
	G1	G2	G3	G4	moyenne	G1	G2	G3	G4	moyenne
fenêtre 1/3	0.62	0.31	0.86	0.60	**0.60**	1	0.50	0.95	0.63	**0.77**
fenêtre 1/2	0.77	0.33	0.84	0.64	**0.64**	1	0.41	0.84	0.47	**0.68**
fenêtre 2/3	0.85	0.19	0.87	0.70	**0.65**	0.85	0.19	0.70	0.35	**0.52**

4.4 Conclusion de l'étude

Cette première étude nous a permis d'évaluer la reconnaissance de gestes professionnels en utilisant des capteurs intrusifs. Nous avons utilisé soit deux MotionPods, soit une veste Animazoo. Nous avons fait des tests de reconnaissance de gestes isolés et des tests en temps réel sur des séquences continues de gestes. Pour les gestes isolés, nous obtenons avec les deux types de capteurs de bons résultats, avec quand même +10% de meilleurs reconnaissances pour l'Animazoo. Nous avons donc utilisé des données provenant de l'Animazoo pour faire de la reconnaissance en temps réel, c'est à dire reconnaître les gestes pendant leur exécution afin de rendre la collaboration avec le robot plus fluide. Nous avons fait un filtrage simple des données pour ne prendre en compte que les gestes qui sont "bien" reconnus pendant une certaine durée. Nous avons deux gestes qui sont bien reconnus, avec des précisions et rappels allant jusqu'à 87% et 100% respectivement. Deux autres gestes, qui sont souvent confondus, obtiennent des résultats bien inférieurs.

Cette première étude a permis de démontrer que la reconnaissance de gestes techniques est possible mais que la méthode employée est perfectible. En effet, l'utilisation du *Gesture Follower*, bien qu'adapté à notre cas d'étude, ne permet pas de faire une reconnaissance précise du fait de son fonctionnement. Construire un HMM à partir d'un

seul modèle le rend peu robuste aux variabilités lors d'exécution d'un même geste. De plus, lorsque deux gestes peuvent paraître similaires aux yeux des descripteurs issus des capteurs utilisés, prendre en compte d'autres sources d'information peut être une solution pour les distinguer. Enfin, lors de cette étude nous ne nous sommes intéressés qu'à un cas "mono-opérateur", c'est à dire que les gestes pour l'apprentissage et la reconnaissance proviennent du même opérateur. Pour avoir un système utilisable en milieu industriel, il sera nécessaire d'avoir un système multi-opérateurs, c'est à dire avec des gestes provenant de différents opérateurs pour l'apprentissage et la reconnaissance.

De plus, l'utilisation de capteurs intrusifs n'est pas à privilégier pour une application industrielle. Le port d'un équipement supplémentaire est source de gêne pour les opérateurs dans la réalisation de leurs tâches. Il est également possible que ces capteurs soient soumis à des perturbations électromagnétiques dans certains environnements industriels.

Dans la suite de ce manuscrit nous allons donc nous intéresser à l'utilisation d'une caméra de profondeur avec une vue de haut pour reconnaître des gestes techniques. Dans le Chapitre 5, nous allons décrire un algorithme permettant le suivi des mains de l'opérateur filmé. Puis, dans le Chapitre 6, nous allons étudier une méthode de reconnaissance de gestes.

Chapitre 5

Suivi des mains de l'opérateur

Sommaire

5.1 Description d'une image de profondeur 79
5.2 Le suivi en plusieurs étapes 80
5.2.1 Graphe 2D du torse de l'opérateur 81
5.2.1.1 Localisation de la tête 81
5.2.1.2 Correction des pixels non-mesurés autour de la tête 81
5.2.1.3 Extraction du haut du corps de l'opérateur 82
5.2.1.4 Construction du graphe 2D du haut du corps de l'opérateur 82
5.2.1.5 Le cas particulier des mains jointes 83
5.2.2 Trouver les mains 84
5.2.2.1 Application de l'algorithme de Dijkstra 84
5.2.2.2 Localisation des mains 85
5.3 Implémentation et résultats 87
5.3.1 Choix d'implémentation 87
5.3.1.1 Le langage C++ 87
5.3.1.2 OpenCV 87
5.3.1.3 Kinect SDK 88
5.3.1.4 RTMaps 88
5.3.2 Rapidité de l'algorithme 88
5.3.3 Précision de l'algorithme 89
5.3.3.1 Précision lors de la localisation des mains 89
5.3.3.2 Suivi des mains 89
5.4 Conclusion 93

Dans ce chapitre nous allons décrire un algorithme permettant de suivre les mains d'une personne filmée par une caméra de profondeur avec une vue de dessus. Nous allons, dans un premier temps, détailler les étapes de notre algorithme. Puis, dans un second temps, nous allons présenter nos choix d'implémentation et les résultats que nous obtenons en appliquant notre solution de suivi des mains de l'opérateur au cas d'étude de collaboration.

5.1 Description d'une image de profondeur

Nous avons décidé d'utiliser une caméra Kinect V1. C'est une caméra a un prix raisonnable, qui a une plage d'acquisition de 40 cm à 4 m, une résolution de sa carte de

profondeur de 640x480 pixels et une fréquence d'acquisition de 30 FPS . Comme expliqué partie 3.3.1.3, ces caméras génèrent des cartes de profondeur où chaque pixel donne une information sur la distance entre la caméra et l'objet filmé. Plus un pixel est foncé, plus il est associé à un objet proche de la caméra. Nous pouvons voir en Figure 5.1 un exemple de carte de profondeur que nous obtenons dans le cas d'étude de collaboration décrit partie 2.2.2. L'image de droite permet de mettre en évidence les objets qui constituent la scène : l'opérateur, les pièces à assembler, les pinces du robot et la table de travail.

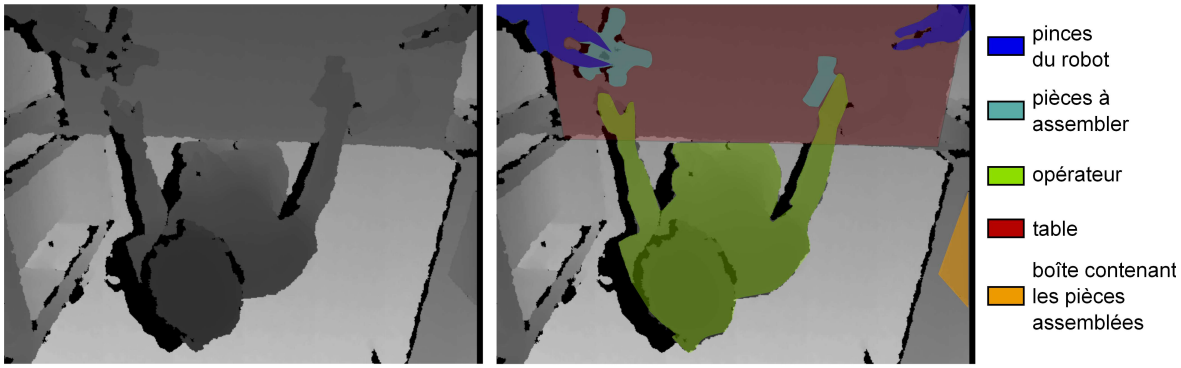


FIGURE 5.1 – Exemple d'une carte de profondeur sur le cas d'étude de collaboration. A gauche : carte de profondeur, à droite : mise en évidence des objets d'intérêt.

Comme expliqué dans la partie 3.3, nous devons extraire des informations pertinentes pour reconnaître les gestes effectués par l'opérateur. Nous avons décidé de nous concentrer sur les mouvements de l'opérateur. Nous souhaitons extraire des descripteurs rapides à calculer, pour faire de la reconnaissance en temps réel, et indépendants de la morphologie de l'opérateur, pour pouvoir mettre en place une méthode de reconnaissance multi-opérateurs. L'approche développée par Schwarz et al. [113] pour le suivi de personnes vue de face nous semblait intéressante car elle répond à ces critères. Cette méthode a de plus l'avantage de ne pas nécessiter l'annotation d'une grande base de données afin de pouvoir squelettiser la personne filmée, contrairement à celle proposée par Shotton et al. [117] par exemple. Nous avons donc décidé de l'adapter à une vue de dessus, en faisant l'hypothèse que les parties du haut du corps qui sont le plus éloignées du haut de la tête selon une distance géodésique sont les mains. Nous allons décrire ci-dessous les différentes étapes de l'algorithme que nous avons mis en place. Nous allons, dans un premier temps, expliquer comment nous extrayons les pixels du haut du corps de l'opérateur et comment nous construisons un graphe 2D à partir de ces points. Dans un second temps, nous expliquerons comment nous y appliquons l'algorithme de Dijkstra [34] pour trouver les mains de l'opérateur et les suivre, le tout en temps réel.

5.2 Le suivi en plusieurs étapes

La Figure 5.2 illustre les différentes étapes de notre algorithme, elles seront détaillées dans les paragraphes ci-dessous. On peut y voir la carte de profondeur initiale, puis l'extraction du haut du corps avec la position du haut de la tête. Ensuite, la distance géodésique de chaque point du torse au haut de la tête, calculée avec l'algorithme de Dijkstra. Enfin la position des deux mains et de la tête, avec les chemins les plus courts les reliant.

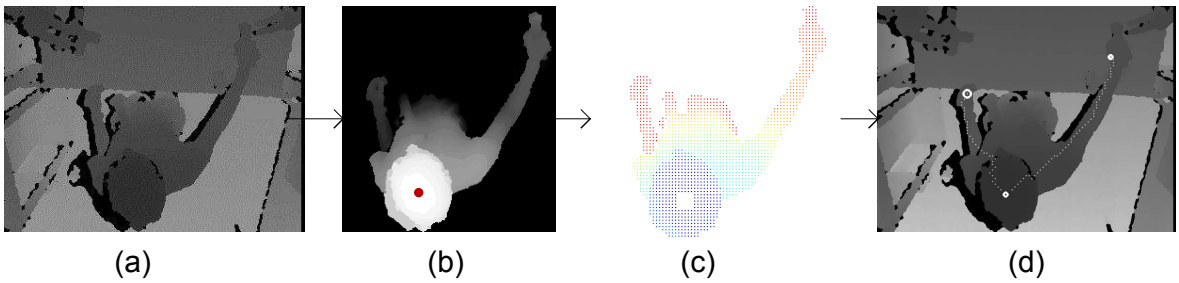


FIGURE 5.2 – Étapes de l'algorithme de suivi des mains. (a) : Image initiale, (b) : Extraction du torse, (c) : Distance géodésique de chaque point du haut du corps de l'opérateur au haut de la tête, (d) Positions des mains, de la tête, et chemins les plus courts les reliant

5.2.1 Graphe 2D du torse de l'opérateur

5.2.1.1 Localisation de la tête

Une première étape de notre algorithme est d'extraire le haut du corps de l'opérateur. Nous faisons l'hypothèse (tout le temps vérifiée) que lorsqu'un opérateur est en train de travailler *sous la caméra*, le point le plus près de la caméra dans l'image est le haut de la tête de l'opérateur. Nous avons choisi de ré-étalonner les valeurs de pixels pour les décrire sur 256 niveaux de gris, de manière à ce que les pixels les plus près de la caméra soient égaux à 255 et que les plus éloignés soient égaux à 0. Un simple seuillage permet de trouver le haut de la tête de l'opérateur. La Figure 5.3 illustre ces étapes.

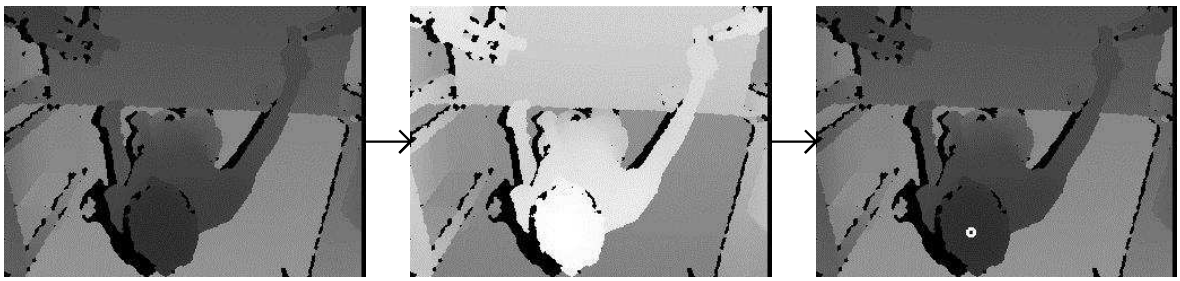


FIGURE 5.3 – Détection de la tête de l'opérateur. gauche : image initiale, centre : pixels sur 256 niveaux de gris, droite : localisation de la tête

5.2.1.2 Correction des pixels non-mesurés autour de la tête

La caméra Kinect projette une lumière structurée, voir partie 3.3.1.3, pour calculer la distance entre la caméra et les objets dans la scène. Les zones du motif infra-rouge qui ne sont pas visibles par l'émetteur et le récepteur conduisent à des pixels auxquels ne sont associés aucune donnée de distance, ils ont pour valeur '0'. Comme la caméra est fixe, pour extraire le haut du corps de l'opérateur, nous souhaitons appliquer un simple seuil au dessus de la table de travail. Ensuite, l'amas de pixels au-dessus de ce seuil auquel appartient la tête serait associé à l'opérateur. Néanmoins, les artéfacts dans l'image liés aux erreurs de mesure de la caméra dissocient parfois la tête du reste du corps, ce qui peut mener à l'apparition de deux amas distincts après seuillage. De plus, dans la suite de l'algorithme, nous voulons calculer la distance géodésique entre chaque point du haut du corps et le haut de la tête, nous souhaiterions donc, dans un premier temps, "souder" la tête au reste du haut du corps en corrigeant ces pixels non mesurés.

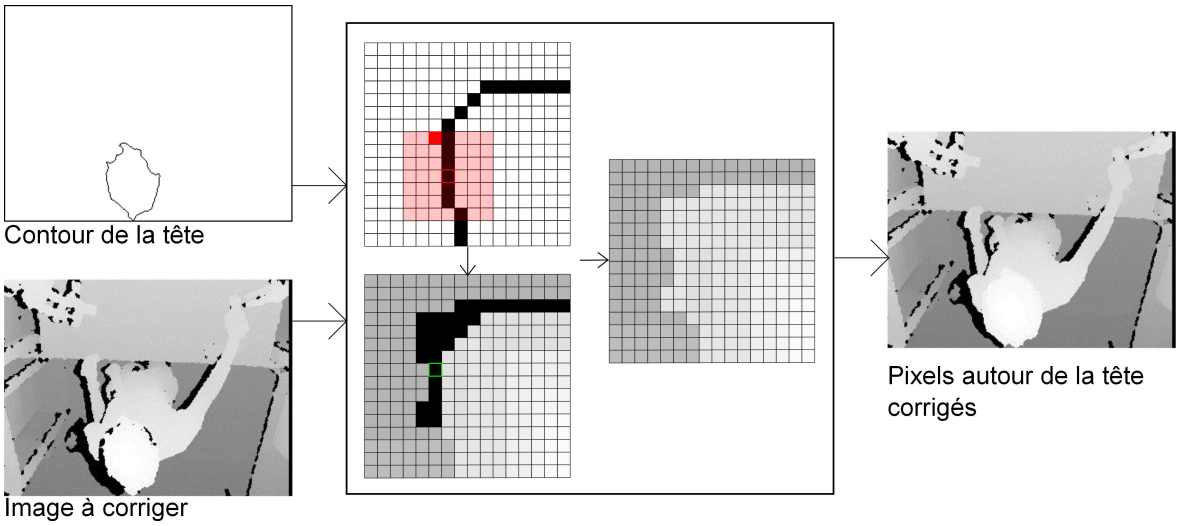


FIGURE 5.4 – Correction des artéfacts autour de la tête

Pour cela, nous extrayons le contour de la tête de la personne filmée grâce à un seuillage. Nous utilisons une fenêtre 2D centrée sur chacun des pixels de ce contour. Si, dans cette fenêtre, un pixel est égal à '0', sa valeur est remplacée par la valeur moyenne des pixels non nuls qui sont à l'intérieur du contour de la tête. La demi-largeur de la fenêtre utilisée est égale à un quart du rayon de la tête. Le rayon de la tête correspond à la distance moyenne entre les pixels du contour de la tête et le centre de la tête. Ces étapes sont illustrées Figure 5.4.

5.2.1.3 Extraction du haut du corps de l'opérateur

Une fois les pixels non mesurés autour de la tête corrigés, nous souhaitons extraire le haut du corps de l'opérateur. Un seuillage au dessus de la table permettra de mettre en évidence le haut du corps de l'opérateur, mais aussi tout ce qui est plus proche de la caméra que la table, notamment les pinces du robot quand celles-ci sont visibles. Comme expliqué précédemment, nous conservons l'amas de pixels qui contient le point associé au centre de la tête.

5.2.1.4 Construction du graphe 2D du haut du corps de l'opérateur

Après avoir extrait le haut du corps de l'opérateur, nous calculons ensuite un graphe 2D reliant les pixels lui appartenant. Ce graphe permettra de calculer la distance entre chaque point du haut du corps et le centre de la tête.

Chaque point du torse peut être relié à ses 8 voisins. Pour accélérer le temps de calcul, nous ne prenons pas en compte tous les pixels du haut du corps de l'opérateur, seulement 1 pixel sur 4.

Une liaison est créée si la différence de profondeur entre deux pixels voisins est inférieure à une valeur maximale. En effet, nous ne souhaitons relier entre eux que les pixels qui appartiennent à des parties anatomiquement connectées du corps. Par exemple, si deux pixels voisins appartiennent respectivement au torse et à l'avant bras de l'opérateur, nous ne souhaitons pas créer de liaison entre eux. Ce seuil est de plus en plus strict lorsqu'on s'éloigne, en distance euclidienne, du centre de la tête. En effet, la différence de profondeur entre la tête est les épaules est supérieure à la différence entre une main devant le torse, or nous souhaitons relier les deux premières parties mais pas les deux se-

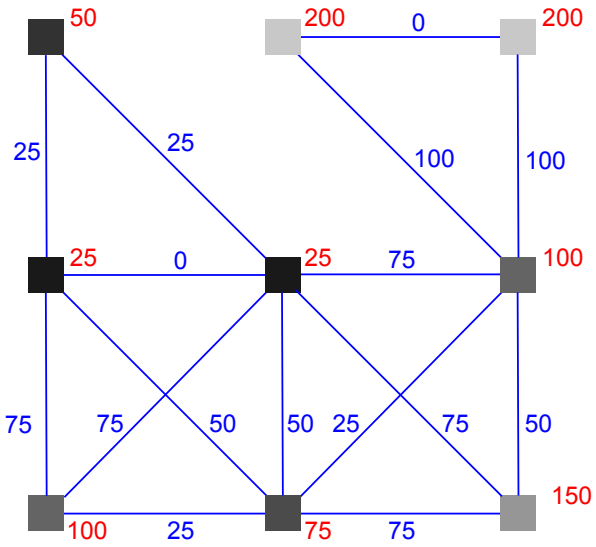


FIGURE 5.5 – Illustration de la construction d'u graphe 2D

condes. Finalement ce graphe 2D est pondéré. On associe à chaque liaison un poids égal à la valeur absolue de la différence de profondeur entre les deux pixels reliés.

La Figure 5.5 illustre la construction de ce graphe. Les pixels sont représentés par des carrés gris, le niveau de gris de chaque carré est inscrit en rouge à coté de celui-ci. La valeur du poids associé à chaque liaison est écrite bleu. Nous n'avons reliés que les pixels qui ont une différence de niveau de gris inférieure ou égale à 100.

L'image du milieu de la Figure 5.7 illustre le graphe 2D du haut du corps de l'opérateur. La couleur de chaque liaison représente la valeur de poids qui lui est associée. Plus la couleur tend vers le rouge, plus le poids est important. On observe que ces liaisons ont un poids particulièrement élevé surtout lors du passage de la tête aux épaules, et également au niveau du coude du bras gauche sur l'image. C'est en effet dans ces zones que les différence de profondeur des pixels voisins sont les plus importantes.

Ce graphe nous servira de base pour appliquer l'algorithme de Dijkstra et calculer la distance géodésique entre chaque point du haut du corps et le centre de la tête.

5.2.1.5 Le cas particulier des mains jointes

Lorsque l'opérateur a les mains jointes, celles-ci peuvent être connectées sur le graphe 2D. Or, par la suite, nous allons chercher les deux parties du haut du corps qui sont le plus éloignées du haut de la tête afin de trouver la position de chaque main. Nous souhaiterions donc que les deux mains ne soient pas reliées, afin qu'elles représentent les deux zones les plus loin de la tête, et non une seule et même zone. Pour éviter cela, nous empêchons la connexion de pixels voisins, perpendiculaires à l'axe reliant les deux épaules de l'opérateur passant par le haut de la tête et éloignés du centre de la tête d'une distance euclidienne au moins deux fois égale au rayon de la tête.

Connaissant la position et donc la hauteur du haut de la tête, nous appliquons un seuil au niveau des épaules. Les positions des deux épaules sont les deux points, de part et d'autre de la tête, dans l'image résultante du seuillage qui sont le plus éloignés de la position de la tête, voir image du milieu Figure 5.6.

Nous pouvons voir dans l'image de droite de la Figure 5.6 la construction du graphe 2D en ne reliant pas les deux mains pourtant jointes. Des pixels du torse sont également déconnectés avec cette méthode, mais cela n'aura pas d'impact ensuite sur la localisation des deux mains dans l'image.

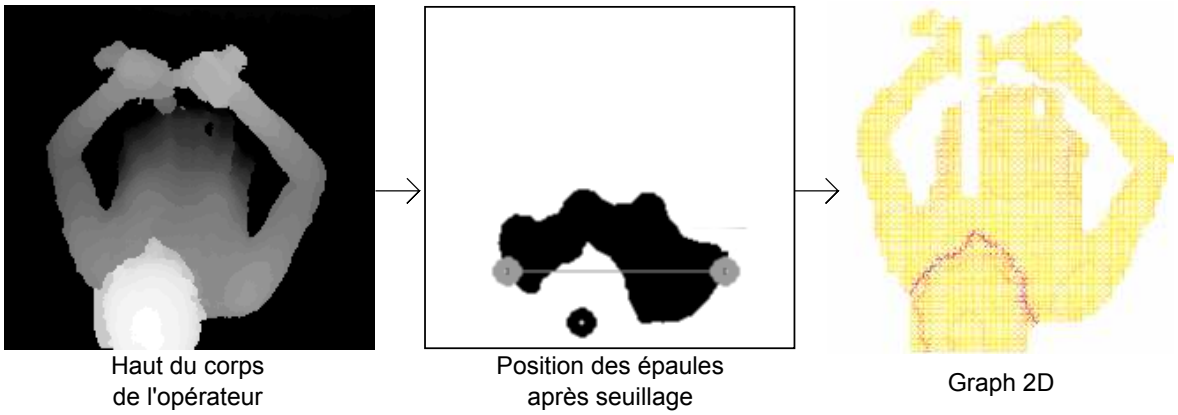


FIGURE 5.6 – Cas particulier des mains jointes dans la construction du graphe

5.2.2 Trouver les mains

5.2.2.1 Application de l'algorithme de Dijkstra

L'algorithme de Dijkstra est un algorithme utilisé pour trouver le *chemin le plus court* entre deux sommets d'un graphe valué. Cet algorithme est décrit dans la partie *Algorithm 1*. Nous utilisons le graphe 2D du haut du corps de l'opérateur que nous avons calculé selon la méthode expliquée dans la partie précédente. En entrée de l'algorithme nous avons donc le graphe 2D précédemment calculé $\text{Graph}_{operateur} = (S, P)$ avec S l'ensemble des sommets du graphe, et P l'ensemble des poids reliant les sommets connectés. En sortie nous avons $D_{operateur}$, la distance entre chaque point de l'opérateur et le haut de la tête et *NoeudPrecedent*, tableau recensant pour chaque point (ou nœud) quel est le point (ou nœud) connexe auquel il est relié pour former le chemin le plus court allant jusqu'au haut de la tête. De manière itérative, on pourra alors retracer le chemin le plus court pour chaque point de l'opérateur.

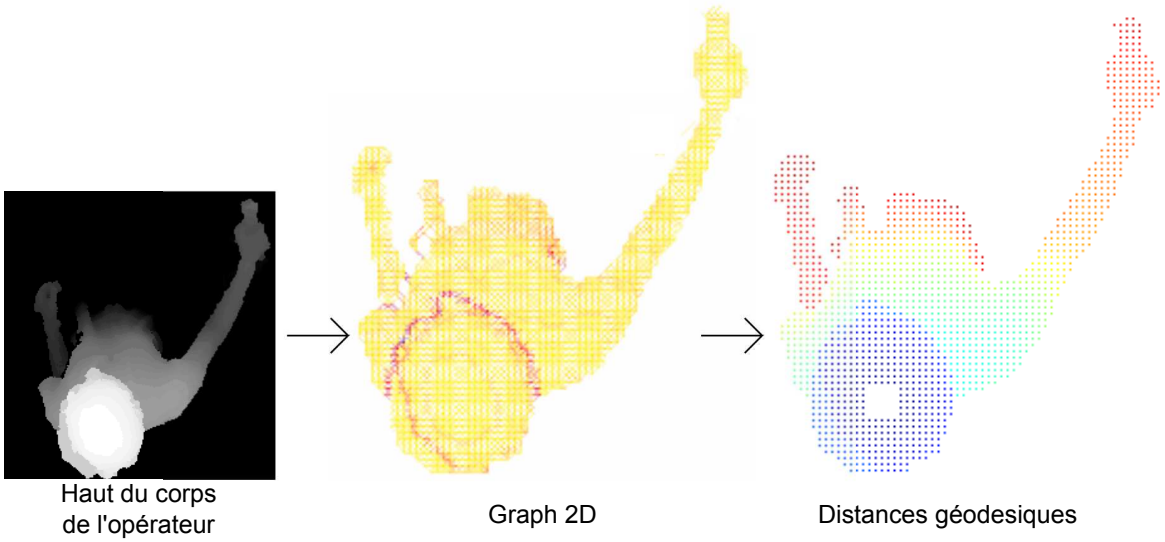


FIGURE 5.7 – Calcul des distances géodésiques du haut du corps de l'opérateur

La distance géodésique correspond à la somme des poids associés aux sommets reliant le haut de tête au point du torse selon le *chemin le plus court*. Cette distance n'a pas d'unité, le coefficient associé à chaque point n'a pas de réalité physique. On va plutôt se servir de ce coefficient comme une information qualitative pour déterminer les parties

du corps les plus éloignées du haut de la tête de l'opérateur, en suivant la topologie de l'opérateur.

Data: $\text{Graph}_{operateur} = (\text{S}, \text{P})$
Result: $\text{D}_{operateur}$, *NoeudPrecedent*

On assigne un coefficient nul au sommet associé à la tête : $\text{D}_{operateur}(\text{S}_{tête}) = 0$;
On assigne un coefficient infini à tous les autres sommets :
$\text{D}_{operateur}(\text{S}_i) = +\infty, \quad i = 1..\text{N}_{sommmets}, \quad i \neq tête$;
while *Tous les sommets n'ont pas été traités* **do**
    On cherche le sommet ayant le coefficient le plus faible, $\text{S}_{c_{min}}$;
    On cherche l'ensemble des sommets adjacents à $\text{S}_{c_{min}}$: $\text{S}_{fils_1}, ..., \text{S}_{fils_\text{K}}$;
    for *Tous les sommets* $\text{S}_{fils_1}, ..., \text{S}_{fils_\text{K}}$ **do**
        if $\text{D}_{operateur}(\text{S}_{fils_k}) < \text{D}_{operateur}(\text{S}_{c_{min}}) + \text{P}(\text{S}_{c_{min}} - \text{S}_{fils_k})$ **then**
            $\text{D}_{operateur}(\text{S}_{fils_k}) = \text{D}_{operateur}(\text{S}_{c_{min}}) + \text{P}(\text{S}_{c_{min}} - \text{S}_{fils_k})$;
            $NoeudPrecedent(\text{S}_{fils_k}) = \text{S}_{c_{min}}$;
        end
    end
end

Algorithm 1: Algorithme de Dijkstra

L'image de droite sur la Figure 5.7 illustre le résultat que nous obtenons après avoir appliqué l'algorithme de Dijkstra à notre graphe 2D. Les couleurs froides représentent les points les plus près du haut de la tête, tandis que les couleurs chaudes représentent les parties du corps les plus éloignées du haut de la tête.

5.2.2.2 Localisation des mains

Une fois que nous disposons de la distance de chaque point au haut de la tête, nous ne conservons que les pixels qui sont les 15% les plus éloignés. Plusieurs scénarios peuvent alors avoir lieu :

- un seul amas de pixels est issu du seuillage : on fait l'hypothèse qu'une seule main est visible
- deux amas de pixels sont issus du seuillage : les deux mains sont localisées
- plus de deux amas de pixels sont issus du seuillage : on conserve les deux amas contenant le plus de pixels

Nous calculons les centres de gravité des amas de pixels associés aux mains. Nous devons ensuite effectuer un suivi de chaque main pour pouvoir reconnaître les gestes effectués par l'opérateur. Une première étape consiste à associer les deux amas aux labels "main gauche" et "main droite".

Si nous ne disposons pas de positions précédentes, on associe l'amas le plus à gauche à la main gauche et l'amas le plus à droite à la main droite.

Si nous disposons des positions précédentes, nous calculons les distances des deux nouvelles positions, (x_i, y_i) et (x_j, y_j), aux deux précédentes, (x_A, y_A) et (x_B, y_B). Pour labelliser une nouvelle position, (x_i, y_i) par exemple, avec un des deux labels, le $label_\text{A}$ qui était précédemment à la position (x_A, y_A), plusieurs critères doivent être remplis :

— la distance euclidienne entre (x_i, y_i) et (x_A, y_A) est inférieure à la distance entre (x_i, y_i) et (x_B, y_B).

— la distance euclidienne entre (x_i, y_i) et (x_A, y_A) est inférieure à la distance entre (x_j, y_j) et (x_A, y_A)

— la distance euclidienne entre (x_i, y_i) et (x_A, y_A) est inférieure à un seuil. En effet, notre programme fonctionne à une fréquence de 10 images traitées par seconde, une main ne se déplace que de quelques centimètres en une dixième de secondes. Nous avons estimé, en connaissant la taille de la table séparant l'opérateur et le robot dans le cas d'étude, qu'un déplacement de 10 cm juste au dessus de cette surface équivalait à un déplacement de 40 pixels. Nous avons donc mis ce seuil à 40.

Si un des deux amas ne remplit pas ces critères, il n'est pas pris en compte pour la localisation des mains. Afin d'avoir un mouvement fluide des deux mains et de ne pas être sensible aux mauvais calculs ponctuels, nous utilisons une valeur moyennées des positions des labels sur les 5 dernière images. Ces positions moyennées peuvent être utilisées comme valeur de référence pour labélliser de nouvelles positions si les positions des mains dans l'image précédente n'ont pas été calculées ou alors ne remplissaient pas les critères cités ci-dessus.

Un inconvénient de l'utilisation des distances géodésiques est qu'en cherchant les parties les plus éloignées du haut de la tête nous ne faisons pas la différence entre la main et la pièce tenue par l'opérateur. Pour limiter les erreurs, nous utilisons le *chemin le plus court* reliant la tête à la position de chaque main initialement calculée. Nous utilisons le tableau *NoeudPrecedent* pour obtenir ce chemin.

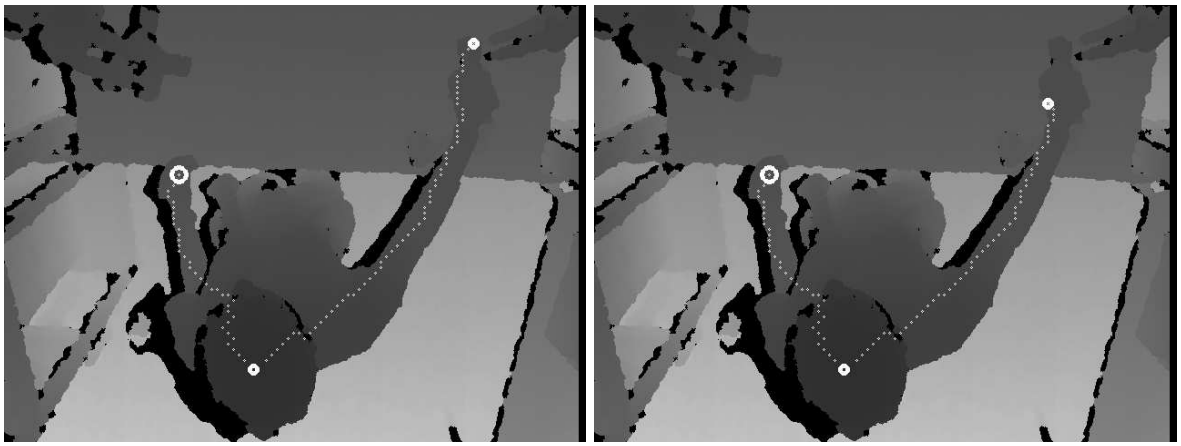


FIGURE 5.8 – Positions des mains, de la tête et les chemins les plus courts. A gauche : avant correction de la main droite, à droite : après correction.

Nous pouvons calculer la distance entre chaque nœud composant ce *chemin le plus court* et la tête. Pour cela, on somme tous les poids reliant les nœuds intermédiaires entre le nœud considéré et la tête, suivant le tracé du *chemin le plus court.*

La distance géodésique entre la main et le haut de la tête est supposée restée constante, quelque soit la position de l'opérateur. Lors d'augmentation brusque de cette distance, nous pouvons revenir en arrière dans le chemin pour finalement sélectionner le nœud qui est à une distance habituelle de la tête. De cette manière on corrige les positions des mains qui peuvent être faussement localisées sur les pièces tenues par l'opérateur. Nous pouvons voir la correction de la position de la mains droite sur la Figure 5.8. Sur l'image de gauche, la main droite est positionnée sur l'objet porté par l'opérateur tandis que la position est corrigée sur l'image droite.

5.3 Implémentation et résultats

Dans la première partie de ce paragraphe nous allons présenter et expliquer nos choix d'implémentation pour cet algorithme. Puis, dans une seconde partie, nous allons présenter des résultats sur le suivi des mains.

5.3.1 Choix d'implémentation

Afin d'implémenter l'algorithme de suivi des mains, nous avons utilisé les solutions suivantes :

— développement en C++ sous Microsoft Visual Studio

— utilisation des librairies OpenCV et KinectSDK

— utilisation du logiciel RTMaps d'Intempora

Nous allons détailler leurs fonctionnements ci-dessous.

5.3.1.1 Le langage C++

Le langage C++ est un langage libre de droit, orienté objet, développé dans les années 80 par Bjarne Stroustrup. Le but de Bjarne Stroustrup était d'améliorer le langage C en y ajoutant des classes et des encapsulations de données. Le langage C++ est aujourd'hui un des langages de programmation le plus utilisé en matière de développement informatique. Sa popularité fait que de nombreux outils sont compatibles avec le langage C++. C'est pour cela que nous avons chois d'utiliser ce langage pour implémenter notre algorithme.

5.3.1.2 OpenCV

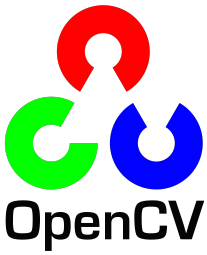


FIGURE 5.9 – Logo d'OpenCV

OpenCV (*Open Computer Vision*) est une librairie libre d'accès développée pour le traitement d'image en temps réel. Depuis 2008, la société Willow Garage assure son support technique. Elle met à dispositions des fonctions classiques de traitement d'image (lissage, filtrage, seuillage...) et de traitement vidéo. Elle met aussi à disposition des implémentation d'algorithmes classiques de traitement d'images et de vidéos plus complexes (transformée de Hough, soustraction d'arrière plan, détection de points d'intérêts...).

Dans notre implémentation, nous avons principalement utilisé des fonctions basiques de traitement d'image comme les fonctions de seuillage, de filtrage, de sous-échantillonnage et de visualisation par exemple.

5.3.1.3 Kinect SDK

Le kit de développement de la Kinect (*SDK : Software Development Kit* en anglais) a été mis gratuitement à disposition par Microsoft avec le lancement de la Kinect pour Windows en 2011. Il permet d'avoir accès au pilote de la Kinect et de pouvoir développer des programmes en C++, C# ou Visual Basic sous Microsoft Visual Studio. Pour cela on a accès directement aux cartes de profondeur issues de la Kinect qui sont codées sur 16 bits, les 13 bits de poids forts donnent la distance en mm tandis que les 3 bits de poids faibles donnent le numéro de la personne détectée (ce numéro est nul si aucune personne n'est détecté au niveau du pixel). Pour déterminer si une personne est filmée par la Kinect, le SDK se base sur l'algorithme de Shotton et al. [117] qui ne permet que de trouver des personne faisant face à la caméra. Le SDK fournit également les positions des articulations des personnes filmées de face par le Kinect, toujours selon Shotton et al. [117]. Le SDK permet de détecter jusqu'à 4 personnes simultanément. Le SDK donne également accès à l'image RGB de la Kinect ainsi qu'à des données audio.

Dans le cadre de la thèse, nous avons utilisé le SDK de la Kinect pour avoir accès à la carte de profondeur afin de l'utiliser pour notre algorithme de suivi des mains.

5.3.1.4 RTMaps

Nous avons utilisé le logiciel RTMaps d'Intempora [1] pour faire des premiers enregistrements d'opérateurs sur notre cas d'étude. Le logiciel RTMaps permet d'enregistrer des données provenant de plusieurs capteurs simultanément. Pour la Kinect, les pilotes du SDK sont utilisés pour avoir accès à la carte de profondeur. Nous avons enregistré chaque carte de profondeur au format *raw*. Nous nous sommes ensuite servi de ces enregistrements pour développer et tester notre algorithme de suivi des mains.

5.3.2 Rapidité de l'algorithme

Pour évaluer les performances de notre algorithme nous avons utilisé une PC équipé de 8 Go de RAM et d'un processeur Intel®i7-3540M. Avec nos choix d'implémentation, ne prendre en compte que 1 pixel sur 4, c'est à dire diviser par 2 le nombre de colonnes et de lignes, nous arrivons à traiter entre 10 et 12 images par secondes pour chaque enregistrement. Ces variations s'expliquent par les différentes morphologies des opérateurs. Plus un opérateur est massif, plus le graphe 2D comportera un grand nombre de points et le temps de calcul de l'algorithme de Dijkstra, de complexité polynomiale, sera long. A contrario, plus l'opérateur est mince, plus le temps de calcul sera rapide.

Si nous ne sous-échantillonnons pas l'image et que nous gardons donc sa taille initiale, nous n'arrivons à traiter au maximum qu'une seule image par seconde. Cette vitesse de traitement est trop faible pour pouvoir faire de la reconnaissance de gestes en temps réel.

Si nous divisons par 4 le nombre de colonnes et de lignes de l'image, c'est à dire si nous ne prenons en compte qu'un pixel sur 16, nous traitons entre 42 et 44 images par secondes. La Kinect fonctionnant à 30 FPS, nous pourrions traiter toutes les cartes de profondeur qu'elle acquiert. Néanmoins, en diminuant autant la résolution de l'image, certaine partie du corps, notamment les mains, sont difficilement localisables. Cela peut entrainer des erreurs dans le suivi des mains et donc a fortiori dans la reconnaissance des gestes.

1. https ://intempora.com/

Nous avons donc choisi de sous-échantillonner avec un facteur 4 la carte de profondeur pour effectuer le suivi des mains. Traiter 10 images par seconde permet d'avoir assez d'informations sur le geste effectué et la taille de l'image après le sous-échantillonnage est suffisante pour détecter les positions des mains.

5.3.3 Précision de l'algorithme

Dans cette partie, nous allons, dans un premier temps, évaluer la précision de notre algorithme lors de la localisation des mains. Puis, dans un second temps, nous allons illustrer des résultats de suivi des mains sur deux exemples alors que l'opérateur enchaîne plusieurs gestes.

5.3.3.1 Précision lors de la localisation des mains

Nous avons évalué la précision de notre algorithme pour la localisation des mains dans l'image. Pour cela, nous avons manuellement annoté 360 images de profondeur prises aléatoirement dans notre base de données. Ces images proviennent de l'enregistrement de 6 opérateurs, nous avons annoté 60 images par opérateur. Pour chaque image nous avons noté la position du centre de chacune des mains dans les images. Nous calculons ensuite la différence, en distance euclidienne en utilisant le pixel comme unité, entre la position annotée et la position calculée par notre algorithme. Nous obtenons les résultats présentés Tableau 5.1. Les images de profondeur que nous utilisons ont une résolution de $640x480$ pixels. La taille moyenne des mains des opérateurs dans les images est de 50 pixels de largeur et 60 pixels de hauteur.

TABLEAU 5.1 – Précision de l'algorithme de localisation des mains

	Main gauche	**Main droite**
Erreur moyenne sur l'axe x (en pixels)	26	24
Erreur moyenne sur l'axe x (en largeur de main)	0.52	0.48
Erreur moyenne sur l'axe y (en pixels)	26	36
Erreur moyenne sur l'axe y (en hauteur de main)	0.43	0.6
Erreur moyenne globale (en pixels)	42	49

Nous pouvons observer que les erreurs moyennes sur les axes x et y (respectivement les colonnes et les lignes de l'image) sont du même ordre de grandeur que la demi-taille de la main. Avec notre méthode, nous cherchons les pixels les plus éloignés du haut de la tête selon la distance géodésique, ils se trouvent généralement sur une extrémité de la main, et non en son centre. Cela explique pourquoi nous observons ces erreurs. Ce test nous a permis de vérifier la précision dans la localisation des mains de notre algorithme.

5.3.3.2 Suivi des mains

Nous présentons ci-dessous deux exemples de suivi des mains. Dans le premier exemple l'opérateur attrape une pièce de moteur dans la pince droite puis dans la pince gauche avant de les assembler. Dans le second exemple, l'opérateur assemble deux pièces, visse,

puis pose la pièce terminée dans une boîte sur sa droite. Afin d'avoir une vérité de terrain, les images correspondantes ont été annotées manuellement comme dans la partie précédente.

La Figure 5.10 illustre le premier exemple introduit ci-dessus. Nous pouvons y voir la position des mains sur plusieurs images consécutives. Ces images filment un opérateur qui prend une pièce dans la pince droite (images 1 à 40), puis une pièce dans la pince gauche (images 40 à 80) et enfin assemble ces deux pièces (images 80 à 100).

Sur les deux graphiques du haut de l'image, en haut pour la main gauche et en bas pour la main droite, on peut voir la position des deux mains calculées par l'algorithme, selon les axes x et y (colonnes et lignes des images), et les positions annotées manuellement sur ces mêmes images. En bas de la figure nous avons fait apparaître trois images avec les positions des mains et de la tête ainsi que les chemins les plus courts reliant chaque main à la tête. Les positions des mains calculées par l'algorithme et les positions qui ont été annotées manuellement restent proches sur l'ensemble des 100 images. On observe bien que la main droite se dirige vers le coin en haut à droite de l'image en début de séquence (x tend vers 640 et y tend vers 0), puis la main gauche se dirige vers le coin en haut à gauche (x et y se rapprochent de 0) avant que les deux mains se rejoignent au centre de l'image pour assembler deux pièces de moteur.

Les deux graphiques du bas indiquent la valeur de la carte de profondeur pour les positions calculées par l'algorithme et pour les vérités de terrain. On observe que les profondeurs calculées et celles provenant de la vérité de terrain sont similaires, mais s'éloignent lorsque les positions des mains calculées et des vérités de terrain divergent. Notamment à la fin des graphiques pour cet exemple.

La Figure 5.11 illustre le second exemple de suivi des mains sur 160 images consécutives. Elle est structurée de la même manière que la Figure 5.10. Lors de cette séquence, l'opérateur assemble deux pièces de moteur (images 1 à 20), puis visse les deux pièces ensemble (images 21 à 110) et enfin pose la pièce terminée dans la boîte à côté de lui (images 111 à 160). La particularité de la dernière action est que seule une main de l'opérateur est visible. En effet, la main qui met la pièce dans la boîte est hors du champs d'acquisition de la caméra, voir la vignette en bas à droite de la Figure 5.11.

Lors de l'étape d'assemblage, les positions des mains calculées par l'algorithme suivent majoritairement les positions qui ont été annotées manuellement. La main gauche diverge légèrement au milieu de l'étape d'assemblage pour revenir aux valeurs de la vérité de terrain. Lorsque l'opérateur visse, on peut décomposer ce geste en trois actions : aller prendre la visseuse sur la table, visser les pièces ensemble puis reposer la visseuse. On peut observer le mouvement de la main droite sur le graphique de haut qui se rapproche de la visseuse sur la droite de l'image (x et y augmentent), puis revient vers le centre pour visser (x et y reviennent au centre de l'image), puis repart sur la droite de l'image pour reposer la visseuse (x et y augmentent à nouveau). Lors de l'action de vissage, lorsque les mains sont revenues vers le centre de l'image, on peut voir que les positions des deux mains s'éloignent des positions annotées manuellement. A ce moment-là, les parties qui sont les plus éloignées du haut du corps sont une main tenant les pièces de moteur et une autre main tenant un visseuse. Bien qu'en prenant en compte la longueur du *chemin le plus court* reliant la tête à ces points pour limiter les erreurs de localisation, dans ce cas les positions ont divergé sur les pièces et la visseuse. Les positions redeviennent proches des positions annotées lorsque l'opérateur repose la visseuse.

Dans les dernières images, lorsque l'opérateur pose la pièce terminée dans la boîte, la main droite n'est plus suivie tandis que la main gauche reste statique.

Pour les deux derniers graphiques, présentant les profondeurs des mains, on retrouve

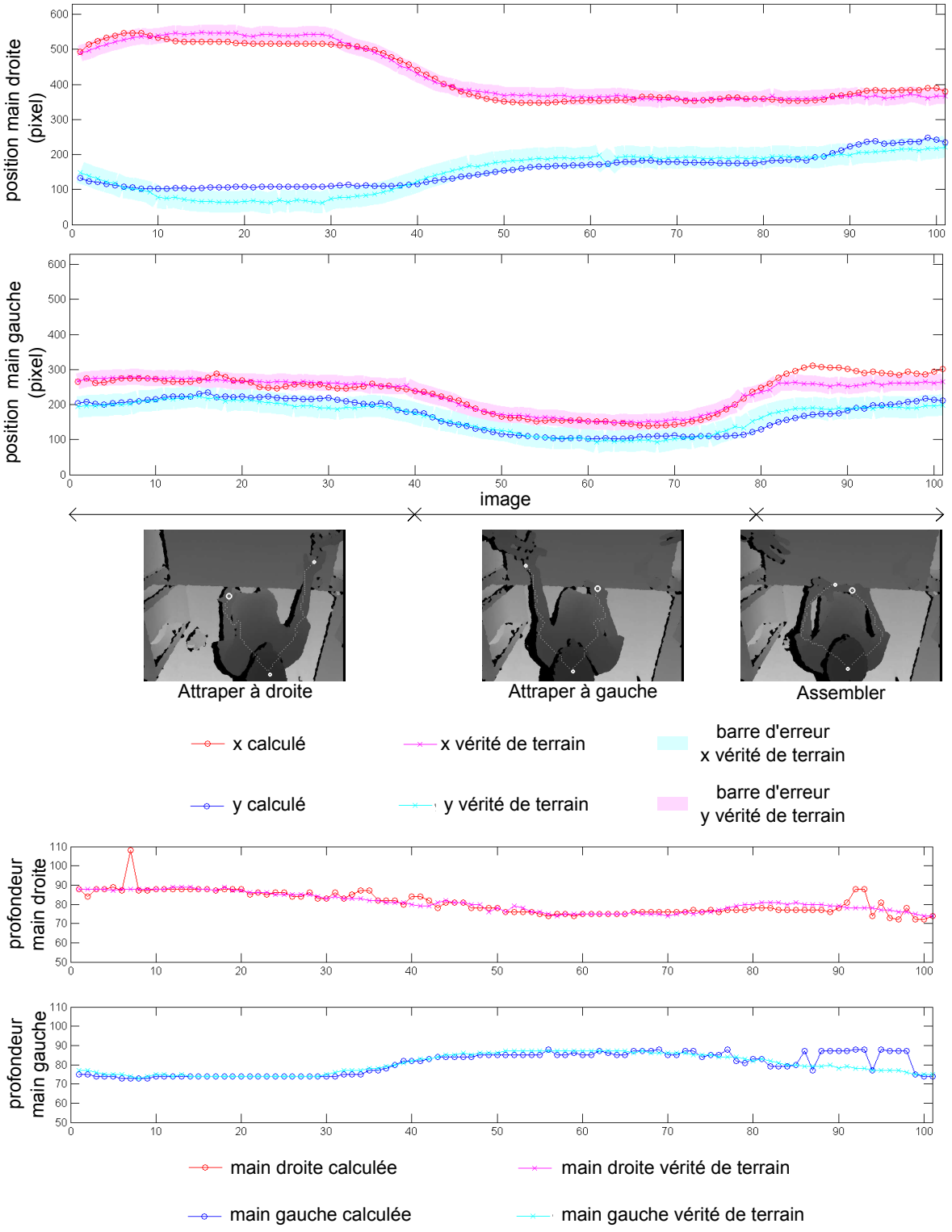


FIGURE 5.10 – Suivi des mains sur 100 images consécutives

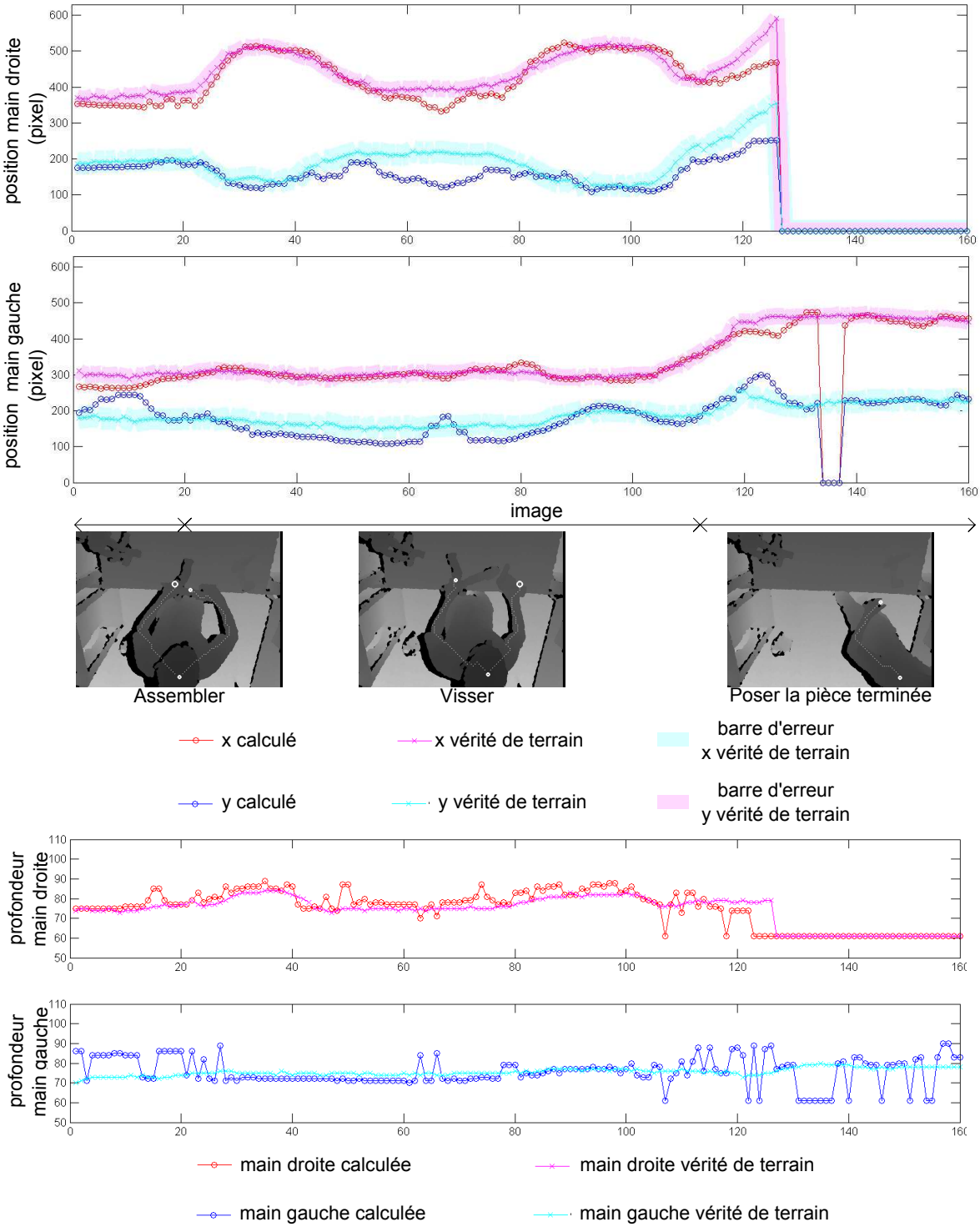


FIGURE 5.11 – Suivi des mains sur 160 images consécutives

les mêmes tendances pour les vérités de terrain et les profondeurs issues des positions calculées. On peut néanmoins observer, comme pour l'exemple précédent, que les profondeurs calculées sont un peu bruitées lorsque la position calculée s'éloigne de la vérité de terrain.

Ces deux exemples ont pu mettre en évidence les résultats de notre algorithme pour le suivi des mains sur une vidéo. Dans la majorité des cas, les mains sont correctement suivies. La principale source d'erreurs est une mauvaise localisation de la main lorsque celle-ci tient une pièce. La position calculée peut alors se trouver sur la pièce. Néanmoins, on peut supposer que cette erreur aura lieu de manière répétitive pour tous les opérateurs, et donc ne sera pas gênante pour la reconnaissance de gestes. En effet, pour avoir une reconnaissance robuste, il faut, pour un geste donné, que nous ayons des données en entrée du système de reconnaissance semblable d'une exécution à une autre. Si la même erreur est effectuée à chaque fois pour un même geste, cela n'ajoutera pas de variabilité dans l'exécution du geste du point de vue du système de reconnaissance.

5.4 Conclusion

Dans cette partie, nous avons présenté notre algorithme de suivi des mains. Pour cela, nous utilisons une caméra de profondeur, avec une vue de dessus. Nous adaptons la méthode de Schwarz et al. [113] pour localiser les mains en calculant la distance géodésique entre chaque point du haut du corps de l'opérateur et le haut de la tête.

Nous avons évalué la précision de cet algorithme en comparant les positions calculées à une vérité de terrain (annotation manuelle) sur 360 images de six opérateurs différents. Nous avons pu observer que l'erreur moyenne est inférieure à la taille moyenne d'une main dans l'image, ce qui nous conforte dans la précision de notre algorithme.

Par ailleurs, nous avons illustré par deux exemples le suivi des mains sur plusieurs images consécutives, 100 pour le premier exemple et 160 pour le second. Sur chacun de ces exemples, plusieurs gestes sont successivement exécutés. Nous avons pu observer que les positions calculées par notre algorithme suivent majoritairement la vérité de terrain. Des erreurs surviennent lorsque l'opérateur tient des pièces dans ses mains. Dans ce cas-là, les parties du corps les plus éloignées du haut de la tête peuvent être confondues avec les pièces de moteur. Une méthode utilisant la distance moyenne du chemin le plus court reliant la tête aux mains est utilisée pour éviter que les positions des mains ne glissent sur ces pièces, mais certaines erreurs persistent. Néanmoins, ces erreurs seront répétables d'une exécution d'un même geste à une autre, et ne devrait donc pas engendrer d'erreurs lors de l'utilisation de ces données pour la reconnaissance des gestes.

Chapitre 6

Reconnaissance des gestes techniques avec des capteurs non intrusifs

Sommaire

6.1 Acquisition d'une base de données 96
6.1.1 Protocole d'acquisition de la base de données 96
6.1.2 Analyse de la base de données 97
6.2 Reconnaissance des gestes techniques 101
6.2.1 Méthodologie de reconnaissance des gestes 101
6.2.2 Choix du descripteur 102
6.2.3 Critères d'évaluation 104
6.2.3.1 Evaluation de gestes isolés 104
6.2.3.2 Évaluation de la reconnaissance des gestes enchaînés en séquence continue 105
6.2.4 Résultats 107
6.2.4.1 Évaluation de la reconnaissance de gestes isolés avec le critère *jackknife* 107
6.2.4.2 Évaluation de la reconnaissance de gestes isolés avec le critère 80%-20% 111
6.2.4.3 Évaluation de la reconnaissance de gestes enchaînés en séquences continues 112
6.2.5 Temps de calcul 114
6.3 Adaptation de la base de données pour un nouvel opérateur 115
6.3.1 Méthodologie 115
6.3.2 Résultats 116
6.3.3 Comparaison avec une base d'apprentissage mono-opérateur . . . 117
6.4 Utilisation du capteur inertiel sur les outils pour affiner la reconnaissance des gestes basée sur la vision 118
6.4.1 Méthodologie 119
6.4.2 Acquisition et traitement des données issues du capteur inertiel . . 119
6.4.3 Résultats 120
6.4.3.1 Reconnaissance de gestes isolés 120
6.4.3.2 Reconnaissance de séquences continues 122
6.5 Gestion des gestes inattendus 124
6.5.1 Présentation du problème 124

6.5.2 Réaction de notre système de reconnaissance face à des gestes parasites 126
6.6 Mise en place d'un démonstrateur 128
6.7 Conclusion et synthèse 129

Dans ce chapitre, nous allons présenter différentes méthodes de reconnaissance de gestes techniques en utilisant un système de capteurs non intrusifs pour l'opérateur. Dans une première partie, nous allons expliquer comment nous avons acquis une base de données de gestes techniques et nous allons décrire cette base de données.

Dans une seconde partie, nous allons présenter notre méthodologie de reconnaissance de gestes avec les données issues de la caméra de profondeur. Nous allons présenter plusieurs descripteurs et deux critères pour évaluer notre système pour la reconnaissance de gestes isolés. Ensuite nous évaluerons également la reconnaissance continue de gestes enchaînés en séquence durant des cycles complets de montage.

Dans une troisième partie, nous étudierons de quelle manière une adaptation de la base de données peut améliorer la reconnaissance d'un système qui est initialement indépendant de l'utilisateur.

Nous nous intéresserons dans une quatrième partie à l'utilisation d'un capteur inertiel sur la visseuse pour affiner notre reconnaissance des gestes techniques. Nous présenterons notre méthodologie ainsi que des résultats sur des gestes isolés et des gestes enchaînés en séquence.

Nous étudierons dans une cinquième partie comment nous pouvons gérer les gestes inattendus, c'est à dire des gestes qui peuvent être effectués par l'opérateur et qui pourraient fausser notre système de reconnaissance.

Enfin, dans une sixième et dernière partie, nous décrirons une implémentation de notre système de reconnaissance pour collaborer avec un robot industriel.

6.1 Acquisition d'une base de données

6.1.1 Protocole d'acquisition de la base de données

Nous avons enregistré 13 opérateurs sur le poste de Kitting décrit partie 2.2.2. Les opérateurs ont effectué entre 4 et 5 cycles, chaque cycle est composé de 5 assemblages de pièces.

Un seul opérateur était présent à la fois lors de ces enregistrements. Dans un premier temps, nous lui expliquions le fonctionnement du robot et nous lui présentions les pièces à assembler. Dans un second temps, l'opérateur effectuait les cycles d'assemblage avec le robot. Aucune directive n'était donnée sur la manière de réaliser les gestes. Comme décrite partie 2.2.2, les gestes à reconnaître que nous avons choisis sont :

geste 1 : attraper une pièce dans la pince gauche du robot (**G1**)

geste 2 : attraper une pièce dans la pince droite du robot (**G2**)

geste 3 : assembler deux pièces (**G3**)

geste 4 : visser (**G4**)

geste 5 : poser la pièce terminée dans la boîte (**G5**)

Le système de reconnaissance des gestes n'était alors pas encore au point, une personne tierce, ou *Magicien d'Oz*, appuyait sur deux boutons pour commander le robot, chacun des boutons commandant l'ouverture d'une des deux pinces. Une fois la pince

ouverte, et donc la pièce donnée à l'opérateur, le robot était programmé pour aller chercher la pièce suivante et attendre une nouvelle commande pour donner cette nouvelle pièce. L'ouverture de chaque pince était actionnée par le *Magicien d'Oz* lorsque l'opérateur tenait la pièce dans la pince du robot, afin de ne pas la faire brutalement tomber sur la table.

La caméra était placée sur un chariot mobile. Tous les enregistrements n'ayant pas eu lieu le même jour, la position de la caméra est légèrement différente d'un enregistrement à un autre.

Le logiciel RTMaps, présenté partie 5.3, a été utilisé pour ces enregistrements.

6.1.2 Analyse de la base de données

Les gestes composant notre base de données sont des gestes techniques qui n'ont donc pas été élaboré pour permettre une collaboration avec un robot. Ils sont exécutés par l'opérateur dans l'optique de réaliser correctement la tâche qui lui est confiée, dans notre cas assembler des pièces de moteur.

La Figure 6.1 illustre la durée moyenne de chaque geste, ainsi que son écart-type. Sur ce cas d'étude, les gestes sont plutôt courts, ils durent tous en moyenne moins de 3 secondes. Pour les gestes 1, 2 et 4 d'une réalisation à une autre, la durée des gestes varie peu, ils ont tous un écart type inférieur à 0,3 secondes. Par contre, les gestes 3 et 4 ont un écart type plus important, respectivement de 0,84 secondes et 0,7 secondes.

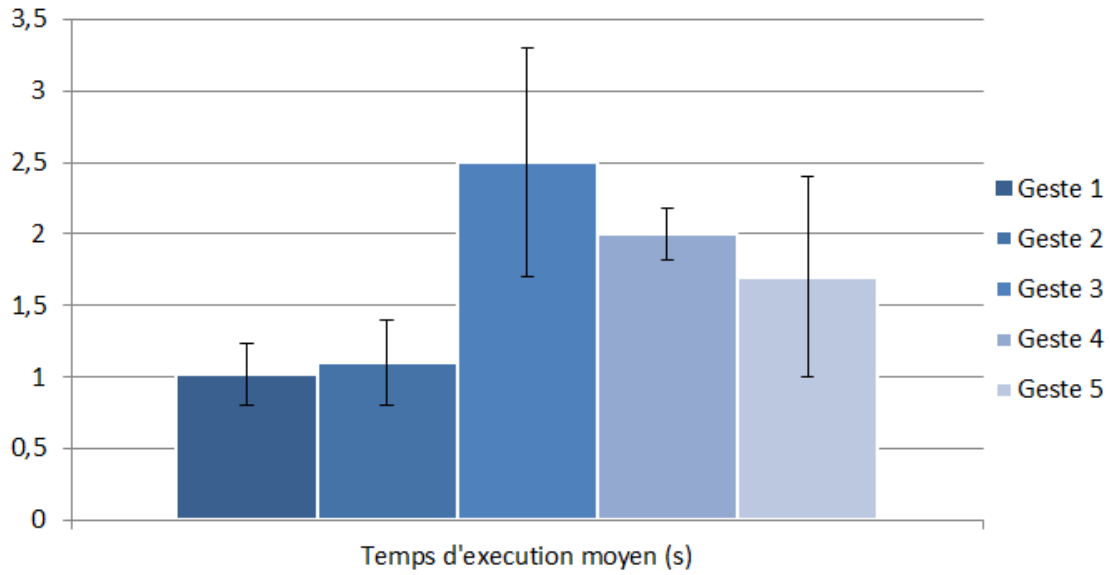


FIGURE 6.1 – Temps d'exécution moyen de chaque geste pour le cas d'étude collaboratif

En effet, ces deux derniers gestes offrent une grande variabilité de réalisation. Pour l'assemblage, le geste 3, l'opérateur n'est pas guidé et peut utiliser plusieurs méthodes pour assembler. Il lui arrive également de s'y reprendre à deux fois pour être sûr que les deux pièces sont bien fixées ensemble. Pour le geste 5, poser la pièce terminée dans la boîte, certains opérateurs se contentent d'allonger le bras vers la boîte pour poser la pièce, tandis que d'autres s'approchent de la boîte afin de "ranger" la pièce, c'est à dire d'ordonner les pièces terminées les unes contre les autres. L'utilisation de l'une ou de l'autre de ces deux méthodes influe sur la durée d'exécution des gestes.

Les Figures 6.2, 6.3, 6.4, 6.5 et 6.8 représentent les projections sur le plan XY des trajectoires 3D des mains : la gauche, à gauche dans les Figures, et la droite à droite dans les

Figures, lors de la réalisation des gestes. Chaque couleur représente un opérateur, pour des questions de lisibilité des Figures nous n'avons fait figurer que 4 opérateurs. De plus, pour chaque opérateur nous avons fait figurer plusieurs exécutions de chaque geste.

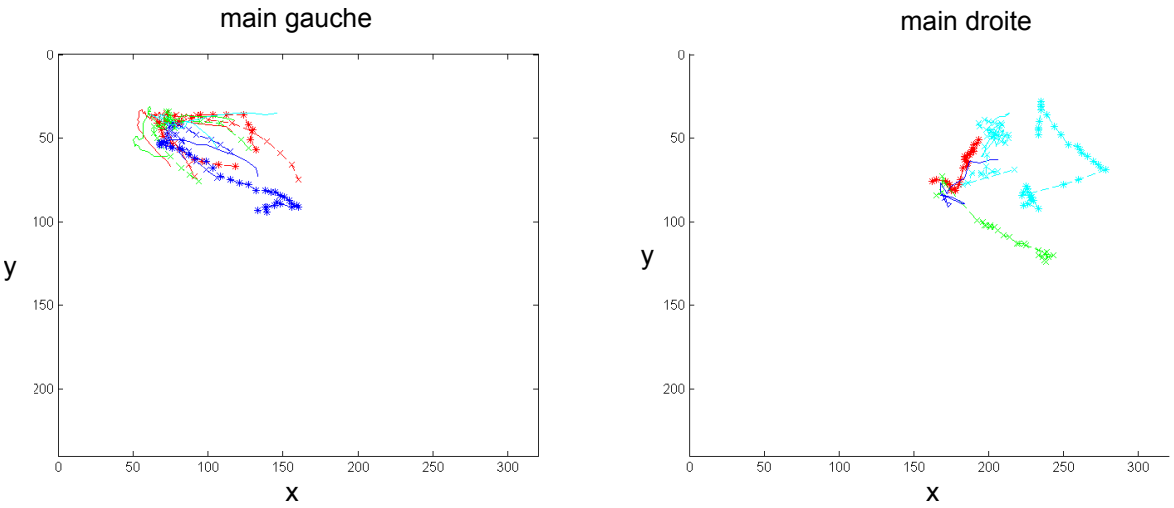


FIGURE 6.2 – Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 1, attraper une pièce à gauche

Nous pouvons voir que lors de la réalisation du geste 1, Figure 6.2, attraper une pièce dans la pince gauche, les mains gauches se rapprochent du coin en haut à gauche de l'image. La main droite, qui est alors en attente, ne fait pas un mouvement particulier. Nous pouvons observer le schéma inverse pour le geste 2, Figure 6.3, attraper une pièce à droite. Cette fois ci c'est la main droite qui fait un mouvement vers le coin en haut à droite de l'image tandis que la main gauche n'a pas de mouvement particulier.

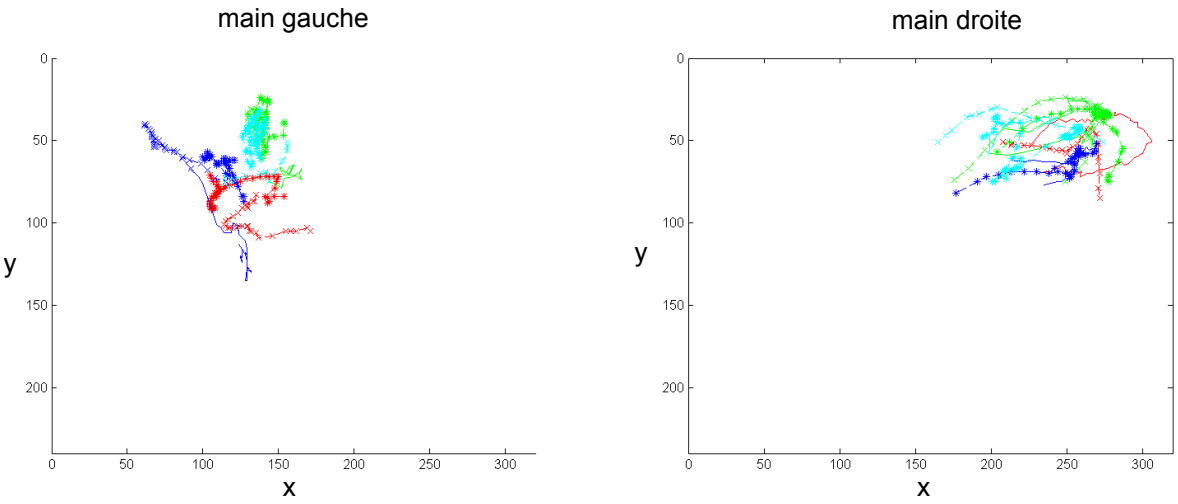


FIGURE 6.3 – Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 2, attraper une pièce à droite

Lors de la réalisation du geste 3, Figure 6.4, assembler deux pièces, l'opérateur a ses deux mains en face de lui. On ne distingue donc pas de mouvement discriminant lors des réalisations de ce geste, que ce soit pour la main gauche ou pour la main droite. Les deux mains restent plutôt statiques au milieu de l'image.

Il est assez difficile de distinguer les gestes 3 (assembler, Figure 6.4) et les gestes 4, (visser, Figure 6.5) lorsqu'on regarde les projections sur XY des trajectoires des mains. La

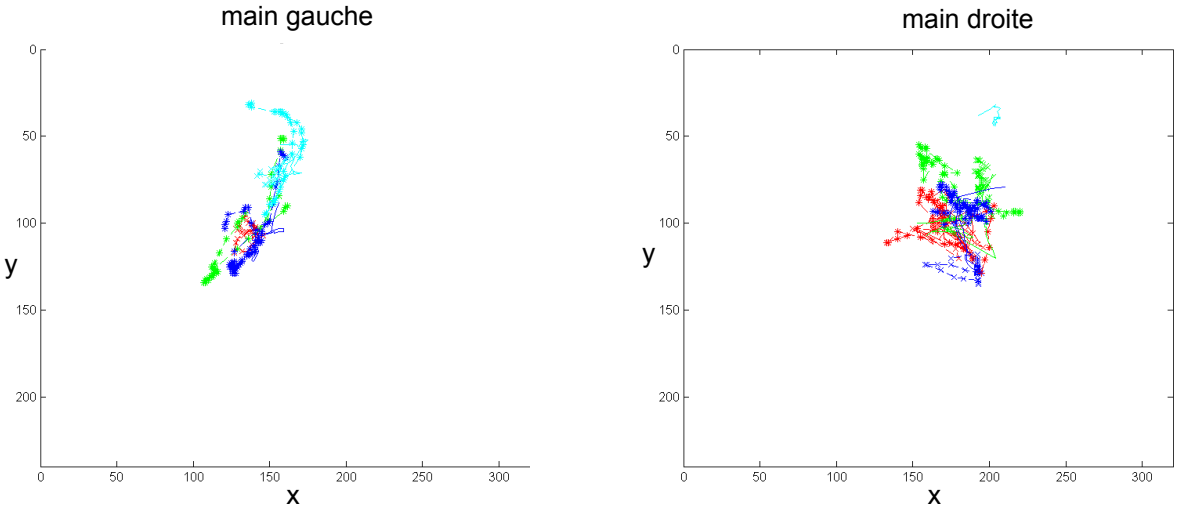


FIGURE 6.4 – Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 3, assembler deux pièces

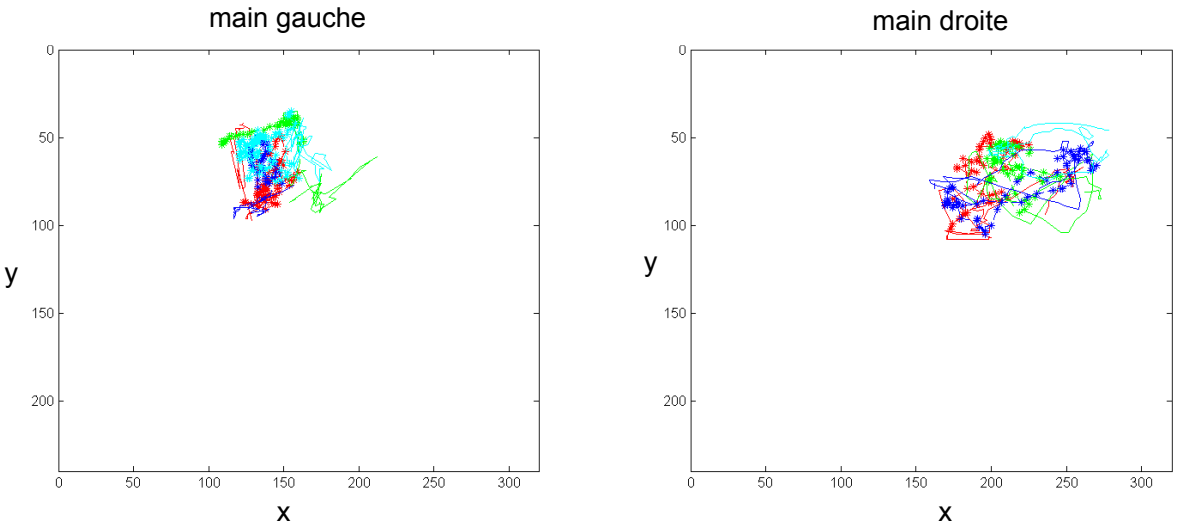


FIGURE 6.5 – Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 4, visser

différence majeure est que, pour le geste 4, la main droite doit aller chercher la visseuse posée sur la table et la reposer ensuite. On peut en effet distinguer des mouvements de la main droite se dirigeant vers la droite avant de revenir au milieu de l'image. Ces différences sont plus visibles lorsqu'on projette les positions 3D des mains effectuant ces deux gestes sur le plan XZ. Ces projections sont visibles Figure 6.6 pour le geste assembler et Figure 6.7 pour le geste visser. Sur ces deux Figures on peut observer que lors de l'exécution du geste visser la main droite fait un arc de cercle vers la droite pour aller chercher la visseuse tandis que pour l'action assembler les deux mains restent au centre de l'image.

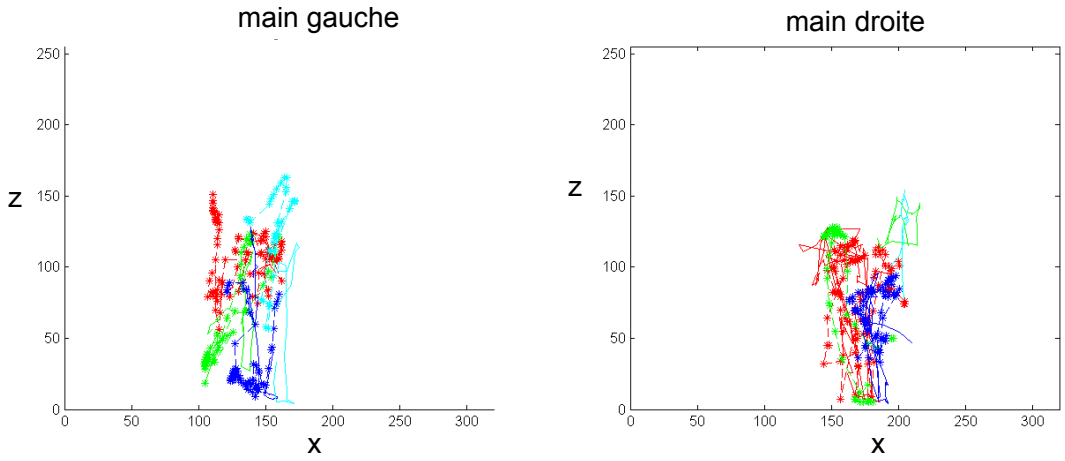


FIGURE 6.6 – Projection sur le plan XZ des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 3, assembler

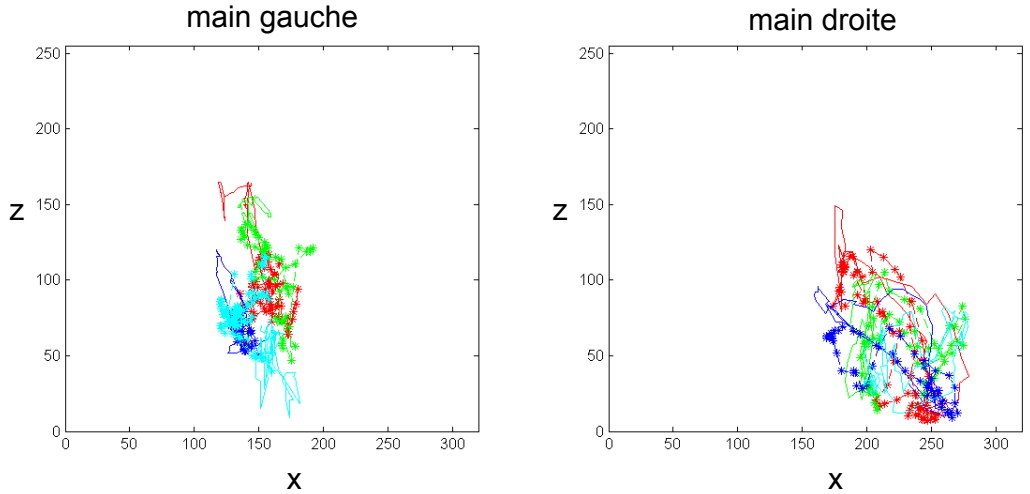


FIGURE 6.7 – Projection sur le plan XZ des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 4, visser

Pour le geste 5, Figure 6.8, poser la pièce terminée dans la boîte, on peut observer que les mains ont souvent une position "nulles". En effet, lors de la réalisation de ce geste l'opérateur s'approche de la limite droite de l'image et, par moment, la tête de l'opérateur sort du champ de vision de la caméra. Dans ce cas-là, aucune des deux mains n'est suivie et notre système renvoie alors une position nulle. Dans d'autres cas, l'opérateur a seulement la partie droite de son corps hors du champ de vue, et donc dans ce cas-là seule la main droite n'est pas suivie pendant un moment.

Grâce aux 7 Figures ci-dessus, nous avons pu voir que chacun des 5 gestes que nous avons choisi de reconnaître ont des caractéristiques qui permettent de les distinguer les

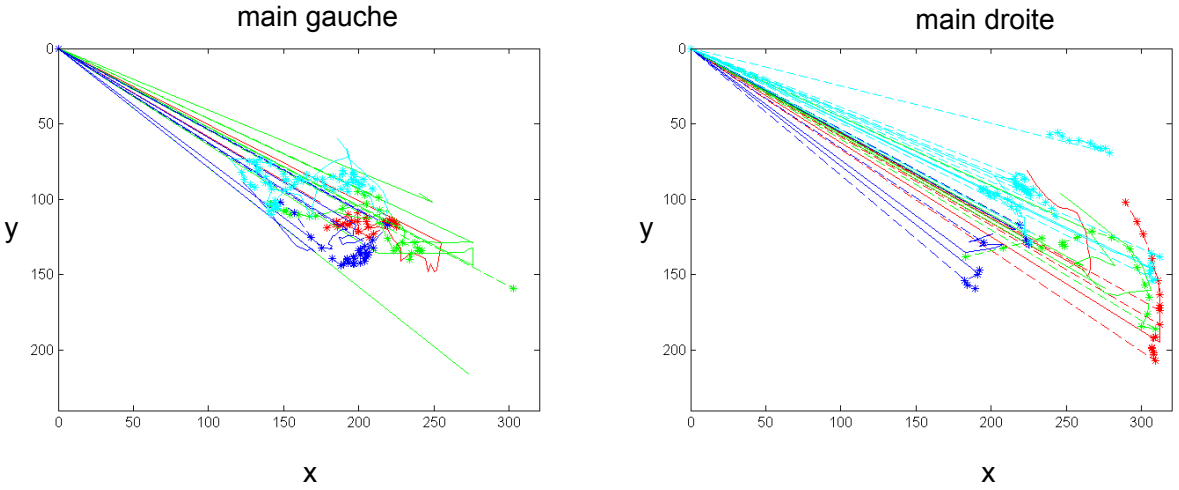


FIGURE 6.8 – Projection sur le plan XY des trajectoires 3D des mains gauche et droite lors de l'exécution du geste 5, poser la pièce terminée dans une boîte

uns des autres. Néanmoins, on peut également voir qu'il y a une grande variabilité dans la réalisation de ces gestes, lorsqu'ils sont effectués par le même opérateur ou par des opérateurs différents. De plus, le geste 3, assembler, peut ressembler à du "bruit" car on ne peut pas distinguer de mouvement vraiment caractéristique, plutôt des petits mouvements autour d'une position au centre de l'image. De plus, lors de la réalisation des gestes 1, 2 et 4 seule une main apporte des informations discriminantes, tandis que l'autre, qui n'est pas contrainte par le geste, peut faire des mouvements parasites qui pourraient peut-être mener à des fausses reconnaissances.

Finalement, les 5 gestes que nous devons reconnaître sont assez courts et fluctuent peu temporellement d'une réalisation à une autre. Néanmoins, ils sont effectués de manières plutôt variables. De plus, les mains qui ne sont pas contraintes lors de la réalisation des gestes sont susceptibles de faire des mouvements pouvant mener à des fausses reconnaissances.

6.2 Reconnaissance des gestes techniques

6.2.1 Méthodologie de reconnaissance des gestes

Nous avons décidé d'utiliser des HMMs discrets pour effectuer la reconnaissance des gestes, ils ont été présentés partie 3.6.2.2.

Comme nous souhaitons utiliser des HMMs discrets, nous avons besoin, dans un premier temps, de convertir nos vecteurs-descripteurs vers un ensemble fini de symboles. Pour cela, nous utilisons l'algorithme des K-Means . Les centroïdes de nos clusters sont appris en utilisant tous les vecteurs-descripteurs de notre base d'apprentissage. Le choix du nombre de clusters a été fait de manière heuristique en testant plusieurs valeurs. Chaque cluster représente un vecteur-descripteur moyen, c'est à dire une posture moyenne dans notre cas.

Pour apprendre nos HMMs, nous devons quantifier nos gestes de la base d'apprentissage. Chaque geste est initialement représenté par une suite de vecteurs-descripteurs. Après quantification, les gestes sont alors des suites de symboles.

Nous apprenons un HMM par geste en utilisant l'algorithme Baum-Welch. Nous avons choisi d'utiliser une topologie gauche-droite, c'est à dire que nous forçons nos HMMs à

créer des transitions ne permettant que de rester dans le même état ou de passer dans l'état suivant pour forcer la description de la dimension temporelle des gestes. Le choix du nombre d'états, tout comme le nombre de clusters dans l'algorithme des K-Means, a été fait en testant plusieurs valeurs.

Lors la reconnaissance, nous devons également encoder nos vecteurs-descripteurs. Nous utilisons les clusters appris avec la base d'apprentissage pour transformer les vecteurs-descripteurs de la base de reconnaissances en symboles.

La reconnaissance se fait en utilisant l'algorithme Forward-Backward. Pour chaque geste de la base d'apprentissage, nous avons, pour chaque HMM, la probabilité qu'il ait pu générer ce geste. Le geste reconnu est le geste qui est associé au HMM ayant la plus grande probabilité.

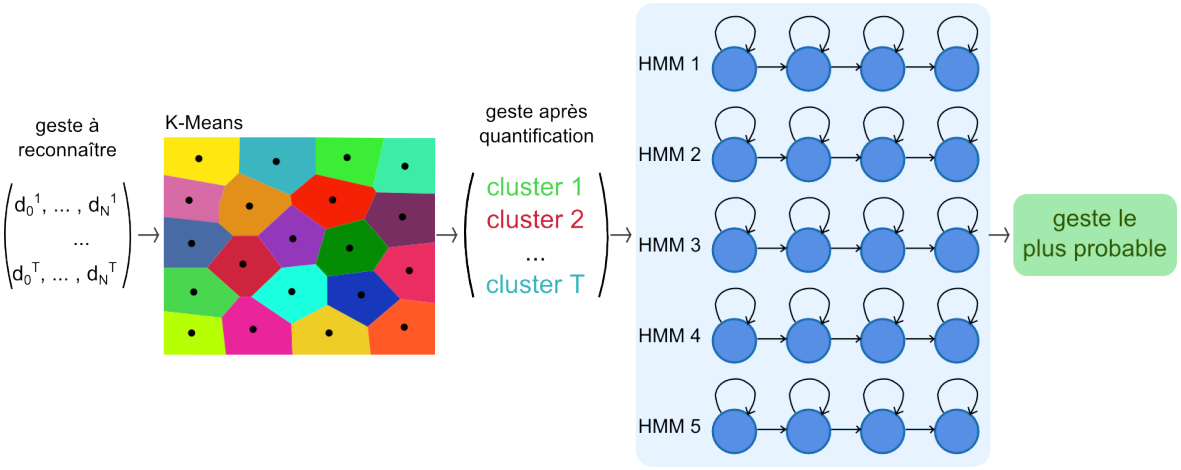


FIGURE 6.9 – Méthodologie de la reconnaissance des gestes utilisant des K-Means et des HMMs

Nous avons utilisé la bibliothèque GRT (*Gesture Recognition Toolkit*) [1] développée par Nick Gillian et décrit par Gillian et Paradiso [45]. GRT est une bibliothèque *open source* codée en C++ permettant d'avoir accès à de nombreuses fonctions d'apprentissage artificiel et spécialisée dans la reconnaissance de gestes en temps réel.

6.2.2 Choix du descripteur

Un facteur important sur la performance d'un système de reconnaissance basé sur les HMMs est le choix des données qu'on met en entrée du système. De notre algorithme de suivi des mains, nous extrayons plusieurs informations : les positions 3D des mains et de la tête et les chemins 3D les plus courts reliant la tête aux mains. Ces informations pourraient être utilisées pour reconnaître les gestes techniques des opérateurs, mais elles ne sont pas nécessairement toutes pertinentes. Nous allons tester plusieurs descripteurs, prenant en compte des informations pour décrire au mieux la posture globale de l'opérateur ou des informations relatives aux parties du corps qui sont les plus impliquées dans les gestes, en l'occurrence les mains.

Nous souhaitons utiliser les chemins 3D les plus courts pour reconnaître les gestes des opérateurs. D'une image à une autre, et donc d'une posture de l'opérateur à une autre, la longueur du chemin varie. Pour pouvoir avoir un vecteur-descripteur de taille constante après le traitement des cartes de profondeurs nous allons échantillonner les chemins les plus courts. Nous allons faire varier le nombre d'échantillons (3, 7 ou 15) pour chaque

1. http ://www.nickgillian.com/wiki/pmwiki.php/GRT/GestureRecognitionToolkit

chemin, voir Figure 6.10. Plus le nombre d'échantillons est élevé, plus nous reproduisons finement la forme du chemin, et donc la posture de la personne, donnant une bonne appréciation du mouvement complet du haut du corps de l'opérateur.

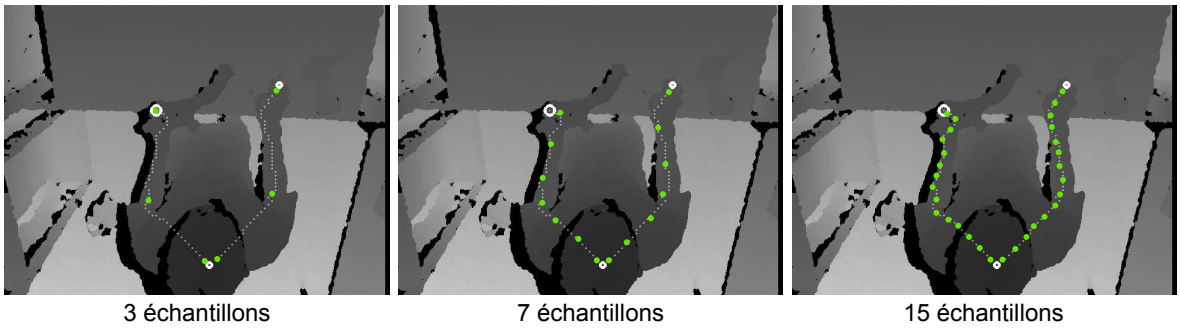


FIGURE 6.10 – Echantillonage des chemins 3D les plus courts pour créer des descripteurs des gestes

Comme il est aussi possible d'envisager que seules les parties du corps qui sont à l'origine du geste technique, notamment les mains, donnent des informations suffisantes pour reconnaître ces gestes, nous allons également essayer des descripteurs prenant en compte seulement la position 3D des mains et de la tête dans la reconnaissance des gestes, voir Figure 6.11.

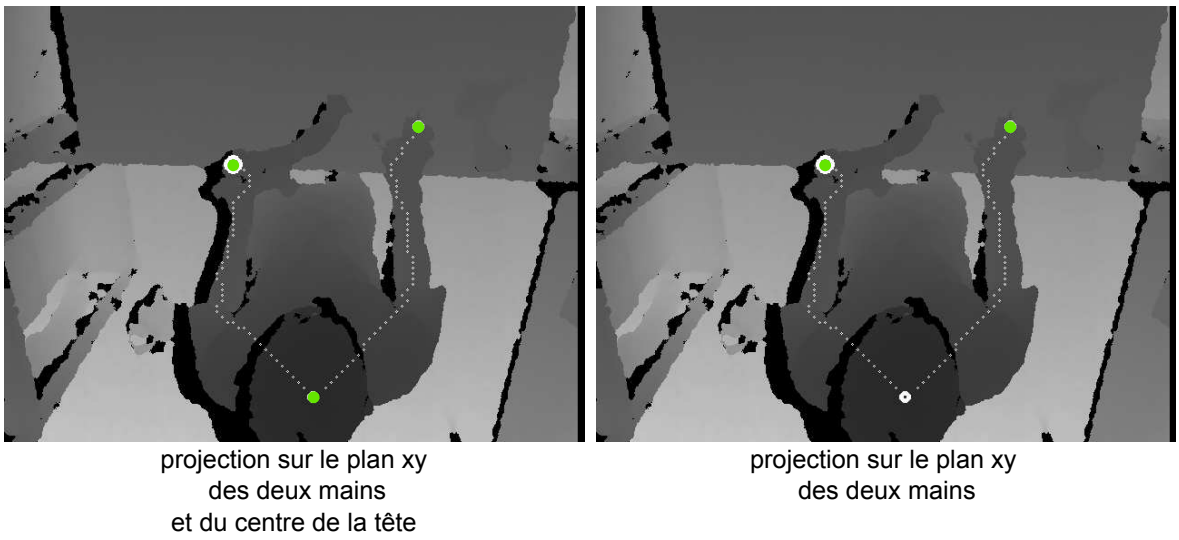


FIGURE 6.11 – Utilisation des positions 3D de la tête et des mains comme descripteurs

De plus, nous pouvons décrire ces positions 3D dans différents référentiels : un référentiel lié à la position de la caméra, un référentiel lié à la scène et un référentiel lié à la position de l'opérateur dans l'image, voir Figure 6.12. Le référentiel lié à la position de la caméra prendra le pixel en haut à gauche de l'image comme origine. Le référentiel lié l'opérateur a pour origine la position du haut de la tête de l'opérateur dans l'image. Enfin, le référentiel lié à la scène aura pour origine le milieu du côté de la table contre l'opérateur. Pour trouver ce point, nous avons utilisé les transformées de Hough pour déterminer les contours de la table, et ainsi établir la position du milieu du côté de la table contre l'opérateur. Pour optimiser notre temps de calcul, nous calculons cette position sur la première image de chaque enregistrement et nous considérons que la caméra ne bouge pas pendant le reste de l'enregistrement.

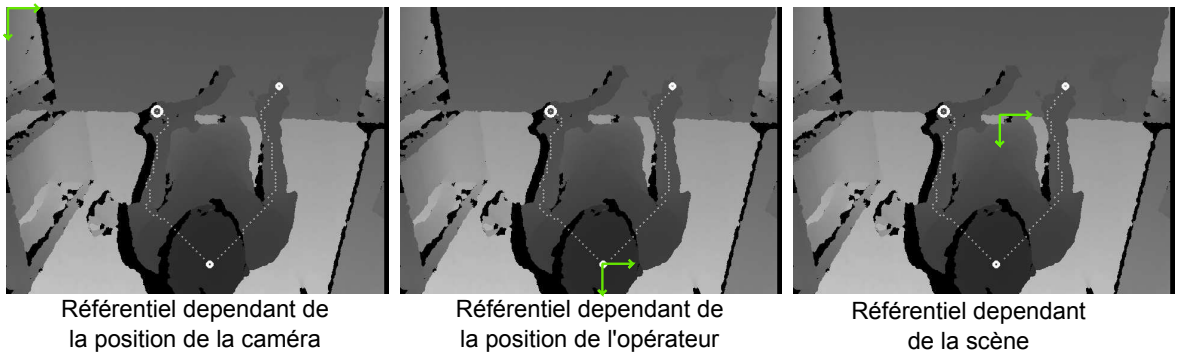


FIGURE 6.12 – Choix du référentiel

Pour ces trois référentiels, les axes x et y sont orientés respectivement horizontalement de la gauche vers la droite et verticalement du haut vers le bas. L'axe z est toujours orienté vers le bas.

En effet, la caméra peut être légèrement déplacée d'un enregistrement à un autre. Nous souhaiterions donc évaluer l'impact d'un tel déplacement et comment un référentiel permettrait de prendre en compte ces changements sans perturber le système de reconnaissance de gestes.

6.2.3 Critères d'évaluation

6.2.3.1 Evaluation de gestes isolés

Pour évaluer notre système avec des gestes isolés, nous avons dans un premier temps segmenté manuellement tous les gestes que nous avions enregistrés. Comme expliqué partie 2.2.2, nous avons sélectionné 5 gestes à reconnaître :

geste 1 : attraper une pièce dans la pince gauche du robot (**G1**)

geste 2 : attraper une pièce dans la pince droite du robot (**G2**)

geste 3 : assembler deux pièces (**G3**)

geste 4 : visser (**G4**)

geste 5 : poser la pièce terminée dans la boîte (**G5**)

Pour évaluer notre système de reconnaissance de gestes, nous avons choisi deux critères. Un premier critère, que nous appelons *jackknife*, permet d'évaluer les performances futures du système pour un nouvel opérateur inconnu. Pour ce critère, nous prenons les gestes de 12 opérateurs pour créer notre base d'apprentissage, puis nous testons notre système avec les gestes du dernier opérateur. Nous testons toutes les combinaisons possibles de 12 opérateurs pour l'apprentissage et de 1 opérateur pour la reconnaissance.

Pour notre second critère, que nous désignons par 80%-20%, nous prenons 80% de l'ensemble des gestes de la base de données pour l'apprentissage et 20% pour la reconnaissance. Avec ce critère, et contrairement au *jackknife*, les gestes dont on évalue la reconnaissance sont effectués par des opérateurs dont d'autres exemples de réalisation de gestes sont présents dans la base d'apprentissage. Ce deuxième critère évalue donc la performance potentielle de reconnaissance pour des opérateurs déjà connus et enregistrés dans la base d'exemple des gestes.

La Figure 6.13 illustre ces deux critères. Chaque point représente un geste, et chaque couleur est associée à un opérateur.

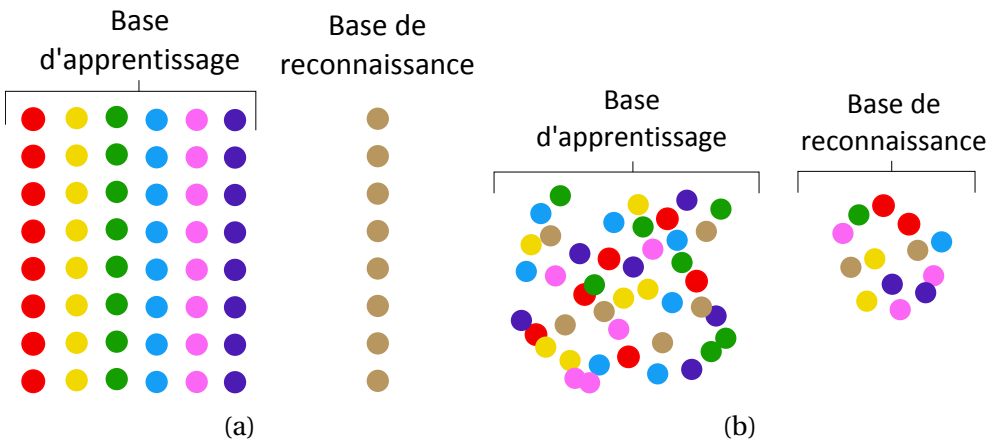


FIGURE 6.13 – Illsutration de nos critères d'évaluation des performances de reconnaissance de gestes isolés. (a) : Critère du *jackknife*, (b) : Critère 80%-20%

6.2.3.2 Évaluation de la reconnaissance des gestes enchaînés en séquence continue

Pour effectuer notre reconnaissance de gestes, nous utilisons l'algorithme Forward-Backward présenté partie 3.6.2.2. Pour la reconnaissance en temps réel, nous évaluons à chaque pas de temps quel HMM a la plus grande probabilité d'avoir généré les N derniers vecteurs-descripteurs. Nous calculons un vecteur-descripteur par carte de profondeur.

Afin de rendre la reconnaissance plus robuste, nous ajoutons des critères pour déterminer si un HMM qui génère la probabilité maximale doit être pris en compte dans le reconnaissance de gestes. En effet, pour chaque suite de vecteur-descripteurs, chaque HMM nous donne une probabilité d'avoir produit cette suite, mais pour limiter les fausses reconnaissances nous ne prenons en compte que les gestes reconnus avec une probabilité supérieure à 0.9 et si le deuxième HMM ayant la plus grande probabilité d'avoir généré ce geste a une sortie inférieure à 0.5.

Si ces conditions ne sont pas remplies, aucun geste n'est reconnu, le système de reconnaissance a alors une sortie nulle. De plus, pour assurer une continuité dans la reconnaissance des gestes, le programme renvoie le geste qui a été le plus reconnu, et au minimum 5 fois, parmi les 10 dernières reconnaissances.

La Figure 6.14 illustre cette méthodologie. L'étape de clustering des K-Means n'y est pas représentée, mais les vecteurs-descripteurs sont bien encodés avant d'être mis en entrée des HMMs.

Le choix du nombre de vecteurs-descripteurs successifs pris en compte pour reconnaître un geste influe sur les résultats de reconnaissance. Il faut que cette fenêtre temporelle soit assez large pour permettre reconnaître les gestes les plus longs, mais plus une fenêtre est large plus elle induit du retard de reconnaissance et peut noyer ensemble plusieurs gestes courts.

Pour évaluer notre système de reconnaissance de gestes en temps réel, nous utilisons la méthode du *jackknife* présentée pour la reconnaissance de gestes isolés. Nous calculons nos HMMs et KMeans sur une base de données de gestes segmentés, puis nous utilisons ces modèles pour faire la reconnaissance en temps réel.

Nous calculons ensuite le taux de reconnaissance correcte, RC, ainsi que la précision, P_i, et le rappel, R_i, de chaque geste i, ainsi que la précision moyenne $\overline{P}$. Ces valeurs sont calculées comme suivant :

$$RC = \frac{\#(gestes\ correctement\ reconnus)}{\#(gestes\ réalisés)} \tag{6.1}$$

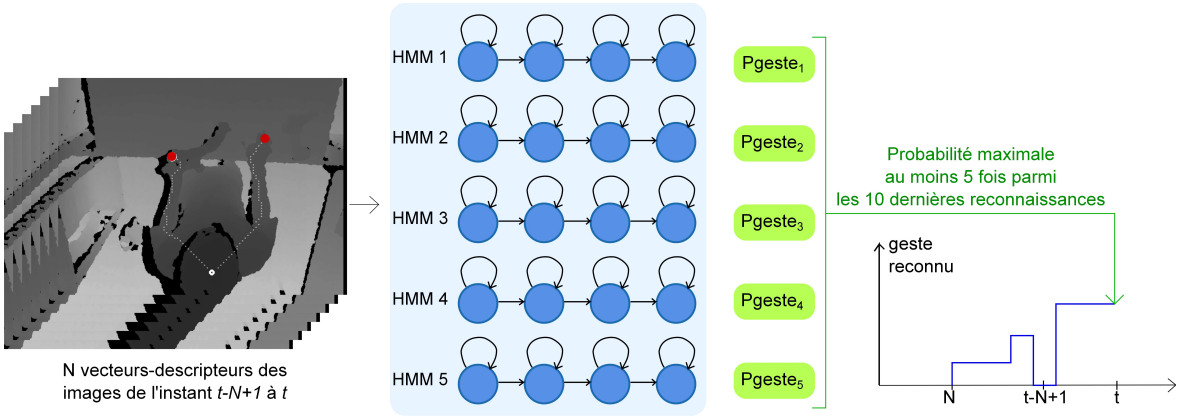


FIGURE 6.14 – Méthodologie de la reconnaissance des gestes en temps réel

$$P_i = \frac{\#(gestes\ i\ correctement\ reconnus)}{\#(gestes\ classés\ i)} \tag{6.2}$$

$$R_i = \frac{\#(gestes\ i\ correctement\ reconnus)}{\#(gestes\ i\ réalisés)} \tag{6.3}$$

$$\overline{P} = \frac{\#(gestes\ correctement\ reconnus)}{\#(gestes\ classés)} \tag{6.4}$$

avec :

- #(*gestes réalisés*) représente le nombre total de gestes réalisés par l'opérateur
- #(*gestes i réalisés*) représente le nombre total de gestes de classe i réalisés par l'opérateur
- #(*gestes correctement reconnus*) représente le nombre de gestes correctement reconnus par notre système de reconnaissance. Un geste est considéré comme correctement reconnu si, pendant l'exécution effective du geste et ensuite sur une durée égale à la taille de la fenêtre de vecteurs-descripteurs, le système l'a reconnu au moins une fois.
- #(*gestes i correctement reconnus*) représente le nombre de gestes de classe i correctement reconnus par notre système de reconnaissance.
- #(*gestes classés i*) représente le nombre de gestes labélisés de classe i par notre système de reconnaissance. Pour incrémenter cette valeur, plusieurs critères doivent être remplis : si à l'instant $t-1$ le geste reconnu par le système était non nul et différent du geste i, alors #(*gestes classés i*) est incrémenté ; si à l'instant $t-1$ le système avait une sortie nulle et qu'à l'instant t le système reconnait le geste i, alors la valeur #(*gestes classés i*) est incrémentée si le dernier geste reconnu était différent du geste i ou si le dernier geste reconnu était le geste i mais cette dernière reconnaissance date d'au moins deux fois la durée moyenne de ce geste.
- #(*gestes classés*) est égal à la somme des valeurs #(*gestes classés i*) selon toutes les classes

La grandeur RC nous donne une information globale de la performance de notre système de classification. Les grandeurs P_i nous indiquent l'exactitude de notre système lorsqu'il croit reconnaître le geste i, autrement la fiabilité que l'on peut lui accorder quand il fournit le label i en sortie. $\overline{P}$ est la moyenne de ces précisions. Enfin, les grandeurs R_i nous informent sur les performances de notre classifieur à bien détecter er reconnaître un geste spécifique.

6.2.4 Résultats

6.2.4.1 Évaluation de la reconnaissance de gestes isolés avec le critère *jackknife*

Comme expliqué ci-dessus, pour la reconnaissance des gestes isolés nous avons, dans un premier temps, segmenté manuellement l'ensemble des gestes enregistrés. Nous avons ensuite évalué notre système avec les deux critères précédemment présentés. Nous allons ci-dessous montrer et analyser les résultats obtenus en fonction du référentiel choisi et en utilisant différents vecteurs-descripteurs. Pour chaque choix de vecteur-descripteur, et selon chaque référentiel, nous avons testé plusieurs combinaisons de nombre d'états pour les HMMs (5, 7, 10, 12 et 15 états) et de nombre de clusters pour les K-Means (10, 15, 20 et 25 états). Pour chaque vecteur-descripteurs dans chaque référentiel, nous donnons le meilleur résultat selon toutes ces combinaisons possibles de nombre d'états et de clusters.

Référentiel caméra Nous avons dans un premier temps décrit les vecteurs-descripteurs dans le référentiel de la caméra, c'est à dire le pixel d'origine est le pixel en haut à gauche de l'image. Nous avons utilisé les informations provenant du chemin 3D le plus court reliant les mains à la tête de l'opérateur. Nous avons échantillonné ce chemin en 3, 7 ou 15 échantillons 3D. Les résultats de reconnaissance en utilisant le critère du *jackknife* peuvent être observés respectivement dans les Tableaux 6.1, 6.2 et 6.3.

TABLEAU 6.1 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel de la caméra, **3 échantillons** des chemins 3D les plus courts reliant la tête aux mains. HMM : 10 états, K-Means : 20 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**142**	44	26	51	8	52%
	G2	34	**418**	5	67	8	79%
	G3	19	5	**391**	89	31	73%
	G4	42	55	78	**294**	15	61%
	G5	-	8	3	6	**252**	94%
	Précision	60%	79%	78%	58%	80%	**72%**

Nous pouvons observer que plus nous augmentons le nombre d'échantillons sur les chemins les plus courts reliant la tête aux mains, moins le taux de reconnaissances correctes est élevé : 72% avec 3 échantillons, 70% avec 7 échantillons et 65% avec 15 échantillons. Dans ces trois cas, c'est le geste 4, visser, qui est le plus confondu avec les autres gestes. Il a en effet une précision qui atteint au maximum 58% avec trois échantillons, contre 54% et 48% pour respectivement 7 et 15 échantillons. De même, le rappel du geste 4 varie entre 61% pour 3 échantillons et 58% pour 15 échantillons. Du point de vue de la caméra, ces deux gestes sont similaires. Dans les deux cas, l'opérateur a ses deux mains proches l'une de l'autre, en face de lui. La différence majeure réside dans le fait que l'opérateur utilise un outil pour visser, qu'il doit prendre puis reposer sur la table.

Avoir une information générale de la forme du chemin le plus court ne semble pas permettre d'améliorer la reconnaissance des gestes techniques. En effet, la forme du chemin n'est pas liée au geste qui est en train d'être effectué et ce chemin peut varier en fonction

TABLEAU 6.2 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel de la caméra, **7 échantillons** des chemins 3D les plus courts reliant la tête aux mains. HMM : 12 états, K-Means : 20 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**164**	39	29	36	38	60%
	G2	42	**359**	14	110	7	67%
	G3	26	6	**392**	90	21	73%
	G4	33	51	96	**288**	16	59%
	G5	-	4	5	8	**252**	93%
	Précision	62%	78%	73%	54%	84%	**70%**

TABLEAU 6.3 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel de la caméra, **15 échantillons** des chemins 3D les plus courts reliant la tête aux mains. HMM : 12 états, K-Means : 25 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**165**	40	24	39	3	61%
	G2	39	**370**	2	108	13	64%
	G3	33	24	**285**	143	50	64%
	G4	57	63	64	**281**	19	58%
	G5	1	7	2	7	**252**	93%
	Précision	60%	73%	76%	48%	75%	**65%**

de la manière dont le geste est réalisé par l'opérateur (coude plus ou moins écartés, bras plus ou moins proches du corps). Lorsqu'on augmente le nombre d'échantillons, on augmente également la quantité d'informations non-discriminantes pour la reconnaissance des gestes.

Nous nous sommes également intéressés aux positions 3D des mains pour la reconnaissance des gestes, avec ou sans la position 3D de la tête. Nous obtenons les résultats présentés Tableaux 6.4 et 6.5, 74% de bonnes reconnaissances en utilisant les positions 3D des mains et de la tête, contre 79% en utilisant les positions 3D des mains seules.

En effet, la position de la tête lors de la réalisation d'un geste technique n'est pas proprement liée à la réalisation de ce geste, mais plus un choix subjectif de l'opérateur pour réaliser le geste. Certains opérateurs préfèrent se pencher pour mieux observer ce qu'ils font, tandis que d'autres préfèrent garder le dos droit. L'information sur la position de la tête n'apporte donc pas d'information pouvant aider à la reconnaissance des gestes techniques.

Avec ces deux types d'information : l' utilisation de la forme du chemin le plus court reliant la tête aux mains, et l'utilisation de la position de la tête et des mains, nous avons pu observer que le meilleur taux de reconnaissance, parmi les vecteurs- descripteurs que nous avons étudiés, est obtenu lorsque nous ne prenons en compte que les positions 3D des mains. C'est en effet avec les mains que les opérateurs réalisent les gestes techniques,

TABLEAU 6.4 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel de la caméra, **Positions 3D des mains et de la tête**. HMM : 10 états, K-Means : 20 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**179**	50	7	18	1	70%
	G2	22	**398**	6	91	7	76%
	G3	4	8	**353**	142	3	69%
	G4	16	27	98	**334**	5	70%
	G5	-	10	1	5	**252**	94%
	Précision	80%	81%	76%	57%	94%	**74%**

TABLEAU 6.5 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Méthode de **jackknife**, référentiel de la caméra, **Positions 3D des mains**. HMM : 5 états, K-Means : 20 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**196**	39	5	10	1	78%
	G2	26	**458**	3	32	5	87%
	G3	3	8	**340**	75	42	73%
	G4	4	39	94	**315**	13	67%
	G5	11	5	3	5	**240**	91%
	Précision	82%	83%	76%	72%	80%	**79%**

et ce sont donc les parties du corps qui sont le plus contraintes pour la réalisation des tâches qui leur sont demandées. Les autres parties du corps, bien que liées, ont plus de liberté lors de la réalisation des gestes et peuvent ajouter du bruit lors de la reconnaissance. C'est pour cela que nous allons principalement utiliser comme descripteur les positions 3D des deux mains pour la reconnaissance des gestes techniques dans la suite de ce manuscrit.

Enfin, nous obtenons un bon score de 79% de reconnaissances correctes. Ce résultat n'est pas suffisant pour utiliser ce système en milieu industriel et assurer une collaboration fluide, mais des améliorations, notamment en corrigeant les erreurs entre les gestes 3, assembler et 4, visser, permettrait de rendre ce système plus robuste et donc utilisable sur les chaînes de montage.

Référentiel lié à l'opérateur Nous nous sommes intéressés à un référentiel lié à la position de l'opérateur dans la scène. Nous avons pensé que décrire les vecteurs-descripteurs dans un référentiel lié à la personne qui effectue les gestes pourrait rendre le système de reconnaissance des gestes plus robuste au changement d'utilisateur. Nous avons choisi comme origine du référentiel la position du haut de la tête de l'opérateur dans l'image. Nous obtenons les résultats présentés Tableau 6.6. Nous n'avons utilisé que la position 3D des mains pour décrire les gestes et nous atteignons 72% de reconnaissances correctes.

Si nous comparons avec les résultats obtenus avec les mêmes vecteurs-descripteurs

TABLEAU 6.6 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel lié à l'opérateur, **positions 3D des mains**. HMM : 7 états, K-Means : 25 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**100**	64	12	89	6	37%
	G2	30	**428**	15	46	13	80%
	G3	1	12	**387**	96	39	72%
	G4	8	41	86	**335**	14	69%
	G5	9	1	4	2	**253**	94%
	Précision	68%	78%	77%	59%	78%	**72%**

dans le référentiel lié à la caméra (Tableau 6.5), nous observons que l'utilisation d'un référentiel lié à l'opérateur ajoute de l'incertitude entre les gestes 1, attraper une pièce à gauche et 2, attraper une pièce à droite, mais surtout entre les gestes 1 et 4, visser. En effet, au début de ces deux gestes, l'opérateur tend la main dans la même direction, vers la droite, pour aller chercher la pièce dans la pince du robot ou pour prendre la visseuse. La pince du robot est plus éloignée de l'opérateur que la visseuse, mais l'opérateur a tendance à se pencher en avant lorsqu'il effectue les gestes 1 et 2. On peut penser que le fait de décrire le vecteur-descripteur dans le référentiel de l'opérateur a rendu ces deux gestes plus similaires en ne permettant pas de différentier le mouvement de la main droite lorsqu'elle prend la visseuse ou lorsqu'elle va chercher une pièce dans la pince du robot.

Ces observations rejoignent celles qui avaient été effectuées pour le référentiel lié à la caméra. Les informations liées aux mouvements de la tête, qui ne sont pas propre à la réalisation des gestes techniques, peuvent ajouter des incertitudes dans la reconnaissance des gestes. Ce référentiel n'est donc pas un choix judicieux pour permettre la réalisation d'un système de reconnaissance robuste dans ce cas d'étude.

Référentiel lié à la scène Enfin, nous avons testé la description de nos vecteurs-descripteurs contenant les positions 3D des mains dans un référentiel lié à la scène. C'est à dire que nous avons sélectionné un pixel associé à un objet fixe dans la scène qui serait l'origine de notre référentiel, quel que soit la position de la caméra et de l'opérateur. Nous avons choisi d'utiliser le milieu du rebord de la table contre l'opérateur. Nous obtenons les résultats présentés Tableau 6.7.

En utilisant ce référentiel, nous atteignons 82% de bonnes reconnaissances. Tout comme les autres référentiels, des erreurs entre les gestes 3, assembler et 4, visser, persistent. Le geste 4 a un rappel de seulement 76% et une précision de 74%, mais les autres gestes sont biens reconnus.

L'utilisation d'un référentiel lié à une position dans la scène, plus robuste aux déplacement de la caméra d'un enregistrement à une autre, nous permet d'améliorer la taux de reconnaissance par rapport à un référentiel lié à la position de la caméra.

Tableau récapitulatif Le Tableau 6.8 récapitule les résultats que nous avons détaillés ci-dessus. Nous avons pu observer que pour un référentiel donné, celui de la caméra dans notre cas, l'utilisation d'informations relatives à l'exécution du geste des mains seules permet de donner de meilleurs résultats de reconnaissance que l'utilisation d'informations décrivant un mouvement plus globale de la posture de l'opérateur.

TABLEAU 6.7 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel lié à la scène, **positions 3D des mains**. HMM : 12 états, K-Means : 25 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**202**	48	10	11	-	75%
	G2	22	**476**	2	27	5	89%
	G3	16	3	**420**	91	5	78%
	G4	6	30	73	**367**	8	76%
	G5	6	9	3	3	**248**	92%
	Précision	80%	84%	83%	74%	93%	**82%**

De plus, l'utilisation d'un référentiel fixe, d'un enregistrement à l'autre, lié à la position d'un objet immobile dans la scène, permet également d'obtenir de meilleurs résultats de reconnaissance.

TABLEAU 6.8 – Tableau récapitulatif des résultats de reconnaissance selon plusieurs référentiels et plusieurs vecteurs-descripteurs avec le critère *jackknife*

Référentiel caméra					**Référentiel opérateur**	**Référentiel scène**
Chemin 3D 3 éch.	Chemin 3D 7 éch.	Chemin 3D 15 éch.	Positions 3D mains et tête	Positions 3D mains	Positions 3D mains	Positions 3D mains
72%.	70%	65%	74%	79%	72%	**82%**

Pour la reconnaissance des gestes, des informations liées à la réalisation du geste, et non à la posture plus globale de la personne effectuant le geste, et l'utilisation d'un référentiel lié à une position fixe dans la scène d'un enregistrement à un autre, nous a permis d'obtenir les meilleurs résultats.

6.2.4.2 Évaluation de la reconnaissance de gestes isolés avec le critère 80%-20%

Nous avons ci-dessus reconnu des gestes de manière indépendante de l'utilisateur, nous allons maintenant utiliser le critère 80%-20% décrit dans la partie 6.2.3.1 ci-dessus. Nous sélectionnons aléatoirement 80% de l'ensemble des gestes de notre base de données pour apprendre les HMMs, les 20% restants sont utilisés pour tester le système. Avec ce critère, pour chaque geste dont on évalue la reconnaissance, la base d'apprentissage contient d'autres exemples de réalisation du même geste effectué par le même opérateur. Nous évaluons ici les performances de reconnaissance de notre système sur de nouvelles réalisations de gestes, mais pour des opérateurs déjà connus et inclus dans la base d'apprentissage (contrairement au critère *jackknife*).

Basé sur les résultats obtenus ci-dessus, nous nous sommes simplement intéressés à la description des gestes techniques grâce aux positions 3D des mains décrites dans le référentiel lié à la scène. Nous obtenons les résultats présentés dans le Tableau 6.9.

Nous obtenons 88% de reconnaissances correctes en utilisant ce critère. Nous pouvons observer que les gestes 3 et 4, assembler et visser restent des gestes qui sont souvent confondus, tout comme le geste 1 avec le geste 2, respectivement attraper à gauche et

TABLEAU 6.9 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **80%-20%**, référentiel lié à la scène, **positions 3D des mains**. HMM : 7 états, K-Means :20 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**29**	12	3	1	-	64%
	G2	2	**116**	1	3	1	94%
	G3	-	-	**89**	9	2	89%
	G4	1	3	9	**77**	-	85%
	G5	-	-	1	3	**77**	93%
	Précision	91%	86%	86%	83%	95%	**88%**

attraper à droite. Néanmoins, 88% est un très bon score de reconnaissance, qui montre que notre système a des performances très satisfaisantes pour des opérateurs déjà connus dont la base d'apprentissage connait des exemples de réalisation de gestes.

6.2.4.3 Évaluation de la reconnaissance de gestes enchaînés en séquences continues

Nous avons évalué notre système pour de la reconnaissance de gestes lors de séquence continues. En effet, nous avons enregistré les opérateurs en train d'effectuer des assemblages complets de pièces de moteurs, et nous souhaiterions évaluer comment notre système se comporte lorsqu'il doit reconnaître des gestes qui ne sont pas segmentés. La méthodologie que nous avons utilisée est présentée partie 6.2.3.2.

Suite à l'étude faite sur le choix des vecteurs-descripteurs pour la reconnaissance des gestes isolés, nous allons utiliser comme descripteurs les positions 3D des mains dans le référentiel lié à la scène pour la reconnaissance de gestes lors d'une séquence continue de gestes. Nous allons effectuer des tests de reconnaissance en faisant varier la taille de la fenêtre temporelle contenant les N derniers vecteurs-descripteurs. Nous utilisons des K-Means et des HMMs qui ont été entrainés sur des gestes isolés. Nous conservons ceux qui ont donné les meilleurs résultats, soit 25 clusters et 12 états cachés.

Nous calculons un vecteur-descripteur par carte de profondeur et notre caméra fonctionne à une fréquence de 30 FPS. Notre geste le plus court dure en moyenne 1.13 seconde, soit environ 34 images, et notre geste le plus long dure en moyenne 2.58 secondes, soit environ 77 images. Nous allons faire varier la longueur de la fenêtre temporelle de manière à ce que sa plus petite taille ne puisse pas contenir un geste entier, jusqu'à sa plus grande taille pouvant contenir 60 vecteurs-descripteurs. En effet, 60 vecteurs-descripteurs correspond à 2 secondes d'images, et pour permettre une collaboration fluide, nous ne souhaitons pas avoir un retard de reconnaissance supérieur à ce délai.

Nous avons utilisés les critère *jackknife* et 80%-20% pour évaluer notre système. Les résultats sont récapitulés dans les Tableau 6.10 et 6.11.

Évaluation avec le critère *jackknife* Les valeurs en rouge sur le Tableau 6.10 représentent les plus mauvais taux pour chaque précision ou rappel de chaque geste, selon toutes les tailles de fenêtre testées. Les nombres en vert sont les meilleurs résultats selon toutes les tailles de fenêtre.

Nous obtenons jusqu'à 74% de reconnaissances correctes selon notre méthode de calcul. 74% est un bon score, mais qui reste un peu faible pour un système devant être utilisé

TABLEAU 6.10 – Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
15	**59%**	54%	76%	67%	66%	83%	69%	54%	65%	38%	63%	88%
30	**74%**	67%	83%	78%	70%	87%	76%	68%	80%	60%	80%	88%
45	**74%**	64%	85%	76%	72%	83%	76%	69%	75%	65%	81%	84%
60	**71%**	64%	88%	78%	74%	87%	78%	66%	64%	67%	79%	82%

dans l'industrie.

Nous pouvons observer que c'est pour la plus petite fenêtre, de largeur 15, que nous obtenons les plus de mauvais résultats, surtout en précision. En effet la précision moyenne augmente avec la largeur de la fenêtre, obtenant 78% de précision moyenne pour la fenêtre de largeur 60. Le geste ayant la plus faible précision est le geste 1, attraper une pièce à gauche, avec un maximum de 67% pour la fenêtre de largeur 30. Les autres gestes ont tous une précision au moins égale à 74% pour la fenêtre de taille 60.

Dans un même temps, on obtient de meilleurs résultats de rappel pour des tailles intermédiaires, 30 et 45. En effet, utiliser une fenêtre assez large pour contenir un geste du début à la fin de sa réalisation ne semble pas donner de meilleurs résultats car les gestes plus courts pourraient être noyés au milieu d'autres informations et être moins bien reconnus. De plus, cela ne semble pas non plus altérer la reconnaissance des gestes les plus longs. Le geste 4, visser, qui dure en moyenne 2.7 secondes, soit 77 frames, obtient de meilleurs résultats de rappel pour les fenêtres de taille 30 et 45 que pour la fenêtre de taille 60. Bien que ce geste soit le plus long, les parties qui en sont les plus discriminantes par rapports aux autres gestes : aller chercher la visseuse et la reposer, sont plutôt courtes. On peut penser que la reconnaissance de ces parties discriminantes, plus courtes, a permis de différencier ce geste par rapport aux autres.

Nous pouvons également remarquer que nous avons de très bons taux de rappel pour certains gestes, au-dessus de 80% pour les gestes 2, 4 et 5, respectivement prendre une pièce à droite, visser et poser la pièce terminée, tandis que les gestes 1 et 3, respectivement prendre une pièce à gauche et assembler, obtiennent des taux de rappels maximaux de seulement 69% et 67% respectivement. Cela rejoint les résultats que nous avons obtenus pour les gestes isolés où le geste 1 était parfois confondu avec le geste 2 et le geste 3 avec le geste 4.

Pour conclure, l'obtention d'un taux de reconnaissance de 74% avec le critère *jackknife* est satisfaisant mais perfectible. Dans le cas de notre scénario collaboratif, la reconnaissance, le rappel et la précision des gestes 1, 2 et 5 sont à privilégier pour permettre une bonne interaction. Ce sont en effet les trois gestes qui mettent en contact le robot et l'opérateur, 1 et 2, ou qui indiquent la fin d'un cycle, 5, et qui donc peuvent permettre une synchronisation optimale entre l'opérateur et le robot. Parmi ces trois gestes, c'est le geste 1 qui obtient les moins bons résultats avec un rappel de 69% et une précision de 64%, pouvant obliger l'opérateur à faire souvent deux fois la même action avant que le robot ne réagisse en conséquence.

Évaluation avec le critère 80%-20% Le code couleur pour le Tableau 6.11 est le mêmes que pour le Tableau 6.10.

TABLEAU 6.11 – Résultats de reconnaissance en temps réel en utilisant le critère 80%-20% (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
15	**76%**	37%	78%	71%	77%	80%	68%	68%	94%	55%	74%	94%
30	**83%**	43%	80%	82%	81%	81%	74%	70%	94%	75%	83%	92%
45	**82%**	46%	72%	76%	95%	84%	76%	70%	91%	77%	80%	89%
60	**79%**	58%	82%	80%	84%	78%	78%	67%	77%	80%	82%	89%

Nous pouvons observer que les meilleurs résultats sont obtenus, comme précédemment, pour les fenêtres de largeur 30 et 45. Avec ce critère d'évaluation nous atteignons 83% de bonnes reconnaissances et une précision moyenne de 84%. L'utilisation d'exemples de gestes de l'opérateur test dans la base de donnée permet d'avoir un système plus robuste. De plus, si on compare plus en détails les résultats obtenus avec le critère *jackknife* et ceux obtenus avec le critère 80%-20% on se rend compte que les précisions du geste 1, attraper une pièce à gauche, sont sensiblement plus faibles. Nous avons en effet, avec une fenêtre de largeur 30, 67% de précision du geste 1 avec le critère *jackknife* contre 43% avec le critère 80%-20%. Cela semblerait indiquer que la réalisation du geste 1 est très variable, même intra-opérateur.

6.2.5 Temps de calcul

Une fois l'apprentissage fait, le temps de calcul pour traiter chaque image correspond au temps de calcul pour trouver les mains dans l'image plus le temps de calcul pour tester la reconnaissance sur les N dernières données.

Pour suivre les mains dans la vidéo, nous avons expliqué, partie 5.3.2, que nous traitons au moins 10 images par seconde. Nous avons donc besoin d'au plus 100 ms pour traiter une image.

Nous avons évalué le temps que nous prenons pour tester nos 5 HMMs lorsque nous utilisons une fenêtre de largeur 30, les résultats sont présentés Tableau 6.12. Nous avons fait une moyenne sur 150 reconnaissances pour obtenir ces données.

TABLEAU 6.12 – Temps de calcul pour la reconnaissance des gestes

		Nombre d'états				
		5	7	10	12	15
Nombre de clusters	10	0,45 ms	0,43 ms	0,56 ms	0,67 ms	0,73 ms
	15	0,45 ms	0,45 ms	0,53 ms	0,68 ms	0,77 ms
	20	0,42 ms	0,45 ms	0,57 ms	0,61 ms	0,71 ms
	25	0,50 ms	0,53 ms	0,56 ms	0,81 ms	0,80 ms

Bien que le temps de calcul pour la reconnaissance tend à augmenter avec le nombre d'états des HMMs et de clusters de K-Means, il reste inférieur à la milliseconde. Ce temps de calcul est donc négligeable face au temps nécessaire pour détecter et suivre les mains dans l'image, qui est de l'ordre de 100ms, comme indiqué partie 5.3.2.

6.3 Adaptation de la base de données pour un nouvel opérateur

Nous avons pu observer que les résultats de reconnaissance de gestes isolés sont meilleurs avec le critère d'évaluation 80%-20% (88% de reconnaissances correctes sur des gestes isolés et 83% sur des gestes enchaînés) qu'avec le critère *jackknife* (82% de reconnaissances correctes sur des gestes isolés et 74% sur des gestes enchaînés). On peut en effet pressentir que lorsque la base d'apprentissage contient des exemples de gestes d'un opérateur, ses futurs gestes seront mieux reconnus par le système. Dans un même temps, nous pouvons nous demander combien de gestes sont nécessaires pour avoir une amélioration significative du taux de reconnaissance. Il est en effet possible d'imaginer que dans les usines du futur les opérateurs qui arrivent sur un poste collaboratif aient, au préalable, une formation avec le robot. Au cours de cette formation, on pourrait enregistrer l'opérateur en train d'effectuer des gestes techniques qui seraient ensuite ajoutés à la base d'apprentissage pré-existante.

Le Tableau 6.13 présente les différents temps de calcul nécessaires pour apprendre nos HMMs avec notre base de données en fonction du choix du nombre d'états ou de clusters. Le temps maximal est inférieur à 6 minutes, il est donc tout a fait envisageable de réapprendre les HMMs avec une base de donnée adaptée lorsqu'un nouvel opérateur arrive sur le poste collaboratif.

TABLEAU 6.13 – Temps de calcul pour l'apprentissage des HMMs

		Nombre d'états				
		5	7	10	12	15
Nombre de clusters	10	37 s	59 s	101 s	184 s	219 s
	15	43 s	70 s	124 s	196 s	284 s
	20	52 s	79 s	138 s	228 s	325 s
	25	61 s	95 s	161 s	290 s	351 s

Pour cette étude, nous avons utilisé les positions des mains décrites dans le référentiel dans la scène.

6.3.1 Méthodologie

Nous allons détailler la méthodologie que nous avons adopté pour évaluer à partir de quelle quantité de gestes nous obtenons une amélioration sensible lors de la reconnaissance des gestes. Comme illustré Figure 6.15, nous avons pour cela modifié notre critère *jackknife*.

Pour cela, nous ajoutons des gestes de l'opérateur de la base de reconnaissance dans la base d'apprentissage. Nous ajoutons les gestes par sets, un set correspondant à un exemple de chacun des cinq gestes à reconnaître. Une fois les HMMs appris, nous testons notre système avec les gestes restants de l'opérateur test. Nous testons, de manière analogue au *jackknife*, toutes les combinaisons de N – 1 opérateurs pour l'apprentissage, avec des exemples de gestes de l'opérateur test, 1 opérateur pour le test, moins les gestes ajouté dans la base d'apprentissage. Nous avons ajouté 1, 2, 3, 4, 5 et 10 sets, nous avons

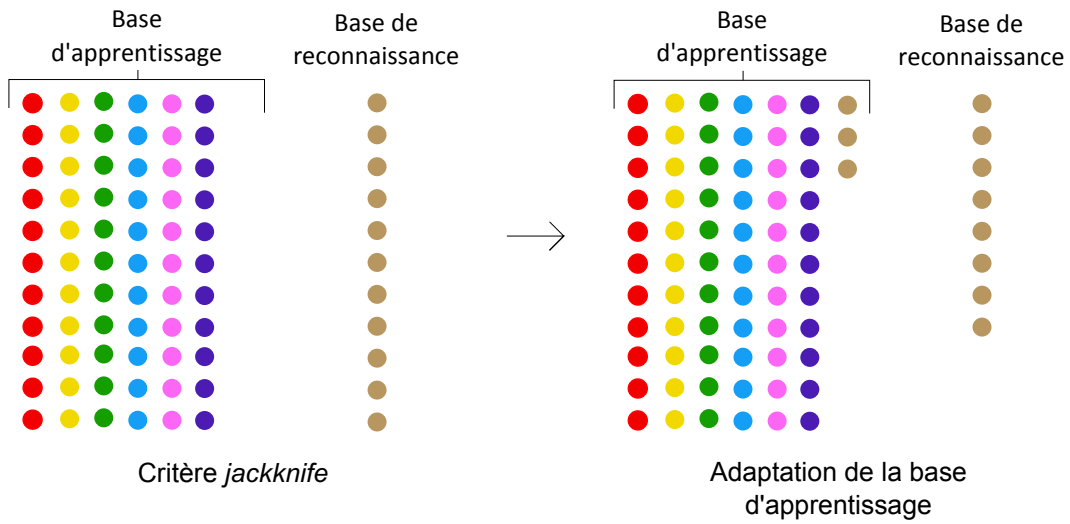


FIGURE 6.15 – Méthodologie pour adapter notre base d'apprentissage et notre critère d'évaluation

testés les résultats de reconnaissance pour chacune de ces configurations en permutant les combinaisons d'opérateur test et d'opérateurs dans la base d'apprentissage.

6.3.2 Résultats

Nous comparons les résultats obtenus pour la reconnaissance de gestes isolés en adaptant la base de données au résultat que nous avions obtenu précédemment avec le critère *jackknife*, avec les mêmes vecteurs-descripteurs, les positions 3D des mains, et décrits dans le même référentiel, lié à la scène. Nous avons testé notre méthode sur des HMMs dont nous avons fait varier les paramètres, nombre d'états, et dont le nombre de clusters pour les K-Means variait également. Nous gardons le meilleur résultat selon tous ces paramètres. Les résultats sont visibles au Tableau 6.14.

TABLEAU 6.14 – Évolution des taux de reconnaissance en fonction du nombre de sets ajoutés de l'opérateur "test" dans la base d'apprentissage

	Nombre de sets ajoutés					
	1	2	3	4	5	10
Amélioration des taux de reconnaissance	+2.4%	+3.5%	+3.6%	+3.1%	+4.1%	+5.7%

Nous pouvons observer qu'avec un petit nombre de gestes ajoutés de l'opérateur test dans la base d'apprentissage, le taux de reconnaissance peut rapidement augmenter pour atteindre une amélioration de +4.1% avec seulement 5 sets de gestes. Ces résulats montrent qu'une adaptation de la base d'apprentissage par ajout d'un nombre limité de gestes de l'opérateur qui utilise le système, permet d'améliorer sensiblement les taux de reconnaissance des gestes.

Pour rentrer plus dans les détails, les Tableaux 6.15 et 6.16 présentent les évolutions de la précision et du rappel pour chaque geste en fonction du nombre de sets ajoutés.

On peut observer que le geste qui bénéficie le plus de cette adaptation de la base de données est le geste 4, visser. En effet, si on regarde la matrice de confusion présentée au Tableau 6.7 qui donne les résultats du *jackknife* pour les mêmes vecteurs-descripteurs

TABLEAU 6.15 – Évolution de la précision en fonction du nombre de sets ajoutés

		G1	G2	G3	G4	G5
Nombre de sets	**1**	-1.9%	-1.1%	2.9%	6.1%	0.0%
	2	0.1%	0.1%	0.5%	7.3%	-1.1%
	3	-0.8%	0.4%	3.6%	5.9%	0.2%
	4	-2.2%	1.0%	3.2%	5.7%	-0.8%
	5	0.1%	0.5%	3.6%	8.9%	0.2%
	10	6.3%	3.7%	3.7%	16.4%	-0.9%

TABLEAU 6.16 – Évolution du rappel en fonction du nombre de sets ajoutés

		G1	G2	G3	G4	G5
Nombre de sets	**1**	-0.4%	-0.1%	0.6%	6.9%	-0.4%
	2	-3.2%	0.5%	2.9%	8.8%	-0.4%
	3	-0.3%	1.0%	1.9%	8.8%	0.0%
	4	1.1%	0.3%	3.0%	6.1%	-2.2%
	5	-0.1%	0.6%	1.9%	7.1%	-2.3%
	10	-0.8%	3.1%	5.4%	8.1%	6.0%

dans le même référentiel, on peut observer que le geste 4 est le geste qui a le plus faible taux en rappel et en précision. Le fait qu'adapter la base de données améliore significativement ces taux pour ce geste laisse penser que son exécution est très variable d'une personne à une autre. Le second geste qui bénéficie de ces adaptations est le geste 3, assembler. En effet, ces deux gestes , assembler et visser, sont les gestes qui sont les moins contraints par l'organisation du cas d'étude. Pour ces deux gestes, l'opérateur n'a pas à interagir avec le robot, dont la position des pinces est fixe lorsqu'il donne une pièce, ou poser la pièce terminée dans une boité qui est également immobile et fixée. Avoir des exemples de ces gestes, dont l'exécution est plus variable, par l'utilisateur dans la base de donnée d'apprentissage permet donc de mieux le reconnaître.

Finalement, une adaptation de la base d'apprentissage avec un faible nombre des gestes de l'utilisateur permet de sensiblement améliorer les taux de reconnaissance des gestes. De plus, ce sont les gestes qui ont une plus grande variabilité lors de leur exécution qui bénéficient au mieux de cette adaptation.

6.3.3 Comparaison avec une base d'apprentissage mono-opérateur

Une fois qu'on a enregistré des gestes d'un nouvel opérateur, on pourrait se demander pourquoi n'utiliserait-on pas seulement ces gestes pour apprendre les HMMs qui seraient ensuite utilisés pour reconnaître les gestes de ce même opérateur ?

Nous allons donc apprendre nos HMMs avec un nombre croissant de sets provenant d'un opérateur et ensuite effectuer le reconnaissance de gestes isolés du même opérateur avec ces HMMs. On appelle ce type de reconnaissance mono-opérateur car les gestes de la base d'apprentissage et de la base de reconnaissance proviennent du même opérateur.

Nous avons testé, comme précédemment, notre système de reconnaissance en faisant varier le nombre d'états (5, 7, 10, 12 et 15) et le nombre de clusters (10, 15, 20 et 25). Nous allons présenter, pour chaque nombre de set ajoutés le meilleur résultat moyenné sur les 13 opérateurs selon toutes les combinaisons de nombre d'états et nombre de clusters. On

peut observer les résultats Tableau 6.17.

TABLEAU 6.17 – Reconnaissance mono-opérateur. Résultats en fonction du nombre de sets de gestes dans la base d'apprentissage

	Nombre de sets de gestes dans la base d'apprentissage					
	1	2	3	4	5	10
Reconnaissance mono opérateur	70%	79%	81%	83%	83%	85%

En ayant un très petit nombre de gestes dans la base d'apprentissage ; 1 set soit un exemple de chaque geste, nous obtenons un taux de reconnaissance plutôt bas de 70%. Ce résultat s'améliore rapidement avec 2 sets de gestes dans la base d'apprentissage en passant à 79%, avant d'atteindre 83% pour 4 et 5 sets dans la base d'apprentissage puis 85% avec 10 sets.

Avec un nombre restreint de gestes dans la base d'apprentissage, 4 sets, on arrive a obtenir de meilleurs résultats que ceux obtenus avec le critère *jackknife* pour les mêmes opérateurs : 83% pour une base d'apprentissage mono-opérateurs de 4 sets contre 82% pour un apprentissage multi-opérateurs sans exemple de l'utilisateur testé (évaluation par le critère *jackknife*). Néanmoins, ces résultats restent en dessous de celui obtenu en apprenant sur une base multi-opérateurs contenant des exécutions de gestes par la personne testée (évaluation avec le critère 80%-20%) avec la méthode 80-20%, 88% de reconnaissances correctes. De même, les performances avec la base mono-opérateur sont aussi inférieures à celles qu'on pourrait obtenir en ajoutant le même nombre de sets dans la base d'apprentissage composés des gestes de N – 1 opérateurs. En effet, en ajoutant 4 sets dans cette base d'apprentissage, on améliore le taux de reconnaissance sur les gestes du dernier opérateur de 3.1% par rapport au *jackknife* (cf Tableau 6.14), soit un résultat de 85.1%.

En faisant une reconnaissance mono-opérateur on aurait pu penser que les gestes de la base d'apprentissage et de reconnaissance seraient plus proches, et donc donneraient lieu à un meilleur taux de reconnaissance, que si on fait une reconnaissance multi-opérateurs. Néanmoins, avoir un grand nombre d'exemples de gestes dans la base d'apprentissage multi-opérateurs permet de mieux prendre en compte les variabilités dans l'exécution de ces gestes, qui peuvent également avoir lieu en mono-opérateur. C'est pourquoi, contrairement à l'intuition, nous obtenons de moins bons résultats en apprenant sur une base mono-opérateur qu'en utilisant une base multi-opérateurs incluant des exemples de gestes de l'utilisateur testé.

6.4 Utilisation du capteur inertiel sur les outils pour affiner la reconnaissance des gestes basée sur la vision

Nous avons décidé d'utiliser des capteurs inertiels pour améliorer nos résultats de reconnaissance utilisant les données provenant du traitement des cartes de profondeur. Comme nous ne souhaitons pas faire porter d'équipement à l'opérateur, nous avons choisi d'équiper la scène, et plus particulièrement les outils. Dans notre cas d'étude, nous avons donc mis un Motion Pod de Movea sur la visseuse. De plus, aujourd'hui dans les

usines automobiles, les outils sont pour la plupart déjà connectés, il est possible de savoir quand ils sont utilisés par les opérateurs. Mettre un capteur inertiel sur la visseuse permet donc de simuler l'accès à ces informations. Néanmoins, la différence majeure avec l'utilisation d'un capteur inertiel est que nous savons quand l'outil est en mouvement, mais pas quand celui-ci est effectivement utilisé. En effet, l'opérateur peut déplacer l'outil sans s'en servir dans sa tâche d'assemblage.

6.4.1 Méthodologie

Nous avons décidé d'utiliser les données issues du capteur inertiel pour affiner les résultats de reconnaissance, et donc de les utiliser après la reconnaissance via notre système composé de K-Means et HMMs, voir Figure 6.16.

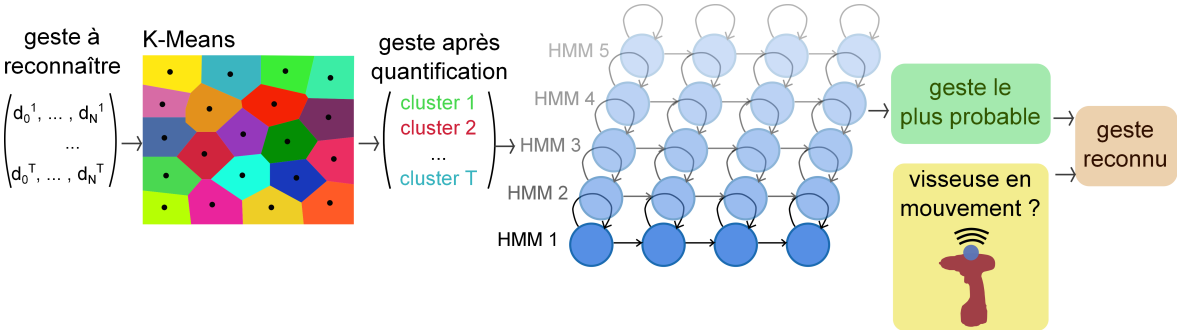


FIGURE 6.16 – Utilisation d'un capteur inertiel sur les outils pour affiner la reconnaissance de gestes

La visseuse n'est sensée être utilisée que lors de l'exécution du geste visser. Savoir si elle a été utilisée, ou déplacée dans notre cas, est une information suffisante pour pouvoir affiner la reconnaissance de ce geste. L'utilisation des informations provenant du capteur inertiel ne nous donnent en revanche pas d'indice sur l'exécution des autres gestes à reconnaître. C'est pour cela que nous préférons utiliser les informations issues du capteur inertiel après la reconnaissance des gestes via les HMMs plutôt que de fusionner ces données avec les vecteurs-descripteurs en amont de la reconnaissance.

Les informations provenant de la reconnaissance de gestes et du capteur inertiel peuvent se contredire dans deux cas :

cas 1 : la visseuse n'a pas bougé et le geste "visser" est reconnu

cas 2 : la visseuse a bougé et le geste "visser" n'est pas reconnu

Dans le premier cas, il n'est pas possible que le geste visser ait eu lieu sans bouger la visseuse, aucun geste n'est alors reconnu, le système de reconnaissance a une sortie nulle. Dans le deuxième cas, il est possible qu'il y ait eu une erreur de reconnaissance du geste visser. Si la probabilité émise par le HMM associé à ce geste est supérieure à 0.5, alors le geste reconnu est finalement le geste visser, sinon aucun geste n'est reconnu, comme pour le cas précédant.

6.4.2 Acquisition et traitement des données issues du capteur inertiel

Nous avons enregistré les opérateurs avec la caméra de profondeur et le capteur inertiel en utilisant le logiciel RTMaps décrit partie 5.3. Les deux capteurs ne fonctionnent pas à la même fréquence, 30 FPS pour la caméra, 200 Hz pour le MotionPod. Lors des enregistrements, RTMaps nous fournit, pour les deux capteurs, un fichier regroupant les

timestamps de chaque acquisition de chaque capteur. Cela nous a permis de pouvoir ensuite recaler temporellement entre elles les données provenant du capteur inertiel et de la caméra, et de connaître pour chaque carte de profondeur acquise par la caméra la donnée provenant du capteur inertiel qui lui est associée.

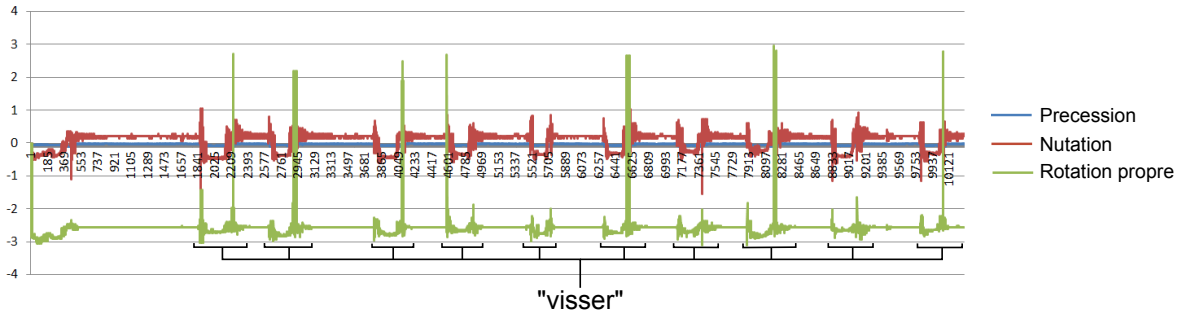


FIGURE 6.17 – Données brutes issues d'un capteur inertiel

Le capteur inertiel nous donne accès à son orientation grâce aux trois angles d'Euler, décrits en radian entre $-\pi$ et π, à chaque pas de temps. Sur la Figure 6.17, on peut observer la valeur de ces trois angles lors de plusieurs assemblages de pièces consécutifs. Lors de l'utilisation de la visseuse, on voit nettement des fluctuations pour les valeurs de chacun des angles. Au début de l'enregistrement, on peut également observer que le capteur, et donc la visseuse, était en mouvement. Cela correspond à la préparation du plan de travail par l'opérateur, c'est lui qui positionne la visseuse à l'endroit qui lui convient le mieux sur la table.

Nous considérons que la visseuse a bougé si la différence de chacun des trois angles d'Euler à l'instant t avec chacun des trois angles d'Euler à l'instant $t-1$ est supérieure, pour chacun des angles, à 0.1 rad.

6.4.3 Résultats

Pour évaluer notre système, nous nous sommes concentrés sur la reconnaissance des gestes en utilisant les positions 3D des mains décrites dans le référentiel lié à la scène. Nous avons ensuite affiné nos résultats avec les données provenant du capteur inertiel. Nous allons présenter ci-dessous des résultats obtenus sur des gestes isolés et sur des séquences continues de gestes.

6.4.3.1 Reconnaissance de gestes isolés

Évaluation avec le critère *jackknife* Nous avons utilisé le critère *jackknife* pour la reconnaissance de nos gestes obtenue par notre algorithme basé sur des K-Means et des HMMs, puis nous avons affiné ces résultats en utilisant les données provenant du capteur inertiels, comme expliqué ci-dessus. Pour comparer avec les résultats sans capteur inertiel, voir Tableau 6.7, nous avons utilisé le même nombre d'états et de clusters pour nos HMMs et K-Means. La matrice de confusion est présentée Tableau 6.18.

Nous pouvons observer que le taux de reconnaissances correctes s'est amélioré de 7% en utilisant les données du capteur inertiel, passant de 82% à 89%. De plus, les taux de rappel des gestes 2, 3 et 4, respectivement attraper une pièce à droite, assembler et visser se sont sensiblement améliorés. En effet, les erreurs de reconnaissance entre les gestes 3 et 4, qui étaient les deux gestes les plus confondus, ont nettement diminué. On pouvait

TABLEAU 6.18 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife**, référentiel lié à la scène et **utilisation des données issues du capteur inertiel**, **positions 3D des mains**. HMM : 12 états, K-Means : 25 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**195**	40	23	1	1	75%
	G2	22	**456**	2	14	5	91%
	G3	19	1	**415**	3	5	93%
	G4	1	13	11	**357**	1	93%
	G5	4	10	2	22	**229**	93%
	Précision	80%	88%	92%	90%	95%	**89%**

également s'attendre à ne pas avoir de faux-positifs pour le geste 5, poser la pièce terminée. Néanmoins, il arrive que l'opérateur bouscule la visseuse lorsqu'il effectue d'autres gestes que visser, ce qui peut indure notre système en erreur. Dans le cas réel, c'est à dire en usine et avec des visseuses connectées, ces erreurs pourront facilement être corrigées améliorant encore le taux de reconnaissance.

D'une manière générale, l'utilisation du capteur inertiel a permis une nette amélioration du taux de reconnaissances de gestes isolées, bénéficiant principalement aux gestes 4 et 5. Nous arrivons à obtenir un score de 89%, ce qui commence à devenir acceptable pour une application industrielle.

Évaluation avec le critère *jackknife* et une adaptation de la base de données Nous avons testé notre système avec une adaptation de la base de données avec 5 sets de l'opérateur test. Les résultat obtenus sont présentés Tableau 6.19.

TABLEAU 6.19 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **jackknife** avec **adaptation de la base de données avec 5 sets de gestes**, référentiel lié à la scène et **utilisation des données issues du capteur inertiel**, **positions 3D des mains**. HMM : 12 états, K-Means : 25 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**141**	41	13	1	0	72%
	G2	22	**394**	2	13	10	89%
	G3	15	3	**371**	3	9	93%
	G4	3	6	8	**310**	1	89%
	G5	7	6	0	17	**170**	85%
	Précision	75%	88%	94%	90%	89%	**89%**

Nous obtenons, comme précédemment, 89% de reconnaissances correctes. Nous avons en effet observé que l'adaptation de la base d'apprentissage bénéficiait principalement au geste 4, visser, voir Tableaux 6.15 et 6.16. La visseuse permet déjà de corriger ce geste, les informations apportées par l'adaptation de la base de données sont donc

redondantes et ne permettent pas, dans ce cas d'étude, une amélioration des taux de reconnaissance. Comme nous l'avions précédemment observé, l'adaptation de la base de données à également fait baisser la précision du geste 1, attraper une pièce à gauche, et a par contre augmenté la précision du geste 3, assembler. Ces résultats sont également en accord avec ce que nous avions pu observer sur les Tableaux 6.15 et 6.16 qui décrivent l'évolution des taux de précision et de rappel en fonction du nombre de sets de gestes ajoutés pour adapté la base d'apprentissage.

Évaluation avec le critère 80%-20% Nous avons utilisé le critère 80%-20% pour évaluer notre système. Nous obtenons les résultats présentés Tableau 6.20. Si nous comparons avec les résultats obtenus avec les mêmes vecteurs-descripteurs sans les données issues de la visseuse, Tableau 6.9, nous pouvons observer que les données issues de la visseuse permettent d'améliorer les résultats de +2%. Le geste 4 est, comme nous pouvons nous y attendre, le geste qui est le mieux reconnu (96% de rappel et 90% de précision).

TABLEAU 6.20 – Matrice de confusion pour la reconnaissance de 5 gestes techniques. Critère **80%-20%**, référentiel lié à la scène et **utilisation des données issues du capteur inertiel**, **positions 3D des mains**. HMM : 12 états, K-Means : 25 clusters.

		Sortie (Probabilité maximale)					
		G1	G2	G3	G4	G5	Rappel
Entrée	G1	**47**	6	2	-	-	86%
	G2	4	**94**	-	2	3	91%
	G3	6	1	**88**	-	3	90%
	G4	-	1	2	**71**	-	96%
	G5	1	-	-	6	**46**	87%
	Précision	81%	92%	96%	90%	88%	**90%**

6.4.3.2 Reconnaissance de séquences continues

De manière analogue à la partie 6.2.4.3, nous avons effectué des reconnaissances sur des séquences continues de gestes. Nous avons pris en compte le mouvement de la visseuse pour affiner nos reconnaissances. Nous avons utilisé les mêmes vecteurs-descripteurs, c'est à dire les positions 3D des mains décrits dans le référentiel lié à la scène, et les mêmes K-Means et HMMs, soit 25 clusters et 12 états cachés. Nous avons également testé plusieurs largeurs de fenêtre de vecteurs-descripteurs selon nos deux critères d'évaluation : *jackknife* et 80%-20%. Les résultats sont réunis dans les Tableaux 6.21 et 6.23. Nous avons également évalué notre système en utilisant une base d'apprentissage adapté avec 5 sets de gestes, Tableau 6.22.

Évaluation avec le critère *jackknife* Avec le critère *jackknife* nous évaluons les performances de notre algorithme lorsqu'un nouvel opérateur, qui n'est pas dans la base d'apprentissage, effectue des gestes. Nous pouvons comparer nos résultats avec ceux obtenus sans les données provenant du capteur inertiel mais avec les mêmes vecteurs-descripteurs, Tableau 6.10.

Nous remarquons surtout une amélioration des taux de rappel et de précision pour le geste 4, visser, qui restent quasiment contant quelle que soit la taille de la fenêtre. C'est

TABLEAU 6.21 – Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* avec des données provenant de la vision et du capteur inertiel sur la visseuse (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
15	**65%**	57%	81%	71%	92%	80%	79%	54%	62%	38%	94%	83%
30	**77%**	68%	87%	79%	92%	86%	84%	68%	79%	60%	95%	87%
45	**77%**	67%	87%	76%	92%	84%	83%	67%	75%	64%	95%	84%
60	**74%**	66%	90%	76%	92%	87%	83%	65%	64%	64%	95%	82%

en effet ce geste qui est visé par l'ajout de données issues du capteurs inertiel, et donc qui en bénéficie le plus. Dans un même temps, les précisions et rappels des gestes 1 et 3, respectivement prendre une pièce à gauche et assembler restent plutôt bas pour une application industrielle. Les meilleurs taux de reconnaissance sont obtenus, tout comme pour la reconnaissances de séquences de gestes avec l'aide du capteur inertiel, pour des fenêtres de taille intermédiaire, avec une amélioration de 3% par rapport aux résultats obtenus sans le capteur inertiel (77% contre 74%). Nous retrouvons la même tendance pour les précisions moyennes avec un résultat maximal de 84% avec les données provenant de la vision et du capteur inertiel, tandis que nous avions au maximum 78% seulement avec les données vision.

Tout comme pour les gestes isolés, l'utilisation du capteur inertiel a permis d'améliorer significativement (+3%) le taux de reconnaissances correctes, principalement grâce au progrès pour les gestes 4 et 5.

Évaluation avec le critère *jackknife* et une adaptation de la base de données avec 5 sets de gestes Nous avons adapté notre base de données en ajoutant 5 sets de gestes de l'opérateur tests dans la base d'apprentissage. Nous nous sommes concentrés sur une fenêtre de largeur 30, largeur donnant, avec la fenêtre de largeur 45, les meilleurs résultats avec le critère *jackknife*. Les résultats sont présentés Tableau 6.22

TABLEAU 6.22 – Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* et une adaptation de la base de données avec 5 sets de gestes, utilisation des données provenant de la vision et du capteur inertiel sur la visseuse (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
30	**90%**	60%	87%	81%	92%	77%	81%	76%	91%	87%	97%	91%

Nous obtenons le bon résultat de 90% de reconnaissances correctes. La précision du geste 1 reste plutôt faible, 60%, mais les autres résultats, précision et rappel, sont tous au dessus de 80%. Nous obtenons une précision moyenne de 81%. L'utilisation des données provenant de la visseuse avec une adaptation de la base d'apprentissage permet d'avoir un système globalement robuste pour la reconnaissance de gestes techniques enchainés.

Évaluation avec le critère 80%-20% Avec ce critère, nous évaluons les performances de notre système lorsqu'un opérateur, qui a également des exemples de gestes dans la

base d'apprentissage, effectue des gestes. Nous pouvons, comme pour le critère *jackknife*, comparer les résultats avec ceux obtenus avec les mêmes vecteurs-descripteurs mais sans les données issues de la visseuse, Tableau 6.11.

TABLEAU 6.23 – Résultats de reconnaissance en temps réel en utilisant le critère 80%-20% avec des données provenant de la vision et du capteur inertiel sur la visseuse (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
15	**80%**	38%	85%	76%	95%	77%	74%	68%	95%	55%	92%	89%
30	**85%**	44%	87%	87%	96%	80%	80%	70%	94%	75%	92%	89%
45	**85%**	55%	86%	80%	95%	86%	82%	70%	91%	77%	92%	89%
60	**81%**	64%	88%	80%	95%	81%	83%	67%	77%	76%	92%	89%

Tout comme pour l'utilisation des données provenant seulement de la vision, le critère 80%-20% donne de meilleurs résultats que le critère *jackknife*, 85% contre 77%. La précision du geste 1, attraper une pièce à gauche, reste faible (44%). La précision moyenne suit la même tendance que pour les résultats sans capteur inertiel (Tableau 6.11), s'améliorant avec l'augmentation de la taille de la fenêtre. L'utilisation des données provenant du capteur inertiel sur la visseuse a permis d'améliorer les résultats de reconnaissance correctes de +3%, tout comme pour le critère *jackknife*. De plus, avec l'utilisation d'une fenêtre de largeur 30, nous obtenons de bons taux de rappel, entre 70% et 94%.

Les résultats obtenus avec le critère 80%-20% restent inférieurs à ceux obtenus avec le critère *jackknife* et une adaptation de la base de données avec 5 sets de gestes. Cela laisse penser que le fait d'avoir une base d'apprentissage contenant principalement des gestes d'autres opérateurs avec une petit ajout de gestes provenant de l'opérateur test permet de mieux prendre en compte la variabilité dans l'exécution des gestes techniques tout en adaptant les HMMs à l'opérateur.

6.5 Gestion des gestes inattendus

6.5.1 Présentation du problème

Lors de sa collaboration avec un robot, un opérateur peut faire un geste inattendu, c'est à dire un geste qui ne fait pas partie des gestes que nous souhaitons reconnaître. Il faut alors s'assurer que le robot ne confond pas ce geste avec un des gestes à reconnaître et qu'il ne soit donc pas induit en erreur dans la réalisation de ces tâches. La Figure 6.18 illustre des exemples de gestes parasites.

Les Figures 6.19 et 6.20 représentent les trajectoires des mains des opérateurs, projetées respectivement sur les plans XY et XZ dans le repère lié à la scène, lors de l'exécution de ces gestes parasites. Il est difficile d'y voir une tendance ou une localisation particulière dans l'image qui permettraient à premier abord de caractériser facilement ces gestes.

Dans cette partie, nous allons seulement nous intéresser à la reconnaissance de gestes avec les données issues de la vision, sans adapter la base d'apprentissage. En effet, nous avons pu observer ci-dessus que l'utilisation du capteur inertiel sur la visseuse ainsi que d'ajouter des gestes exemple de l'opérateur utilisateur du système permettaient d'améliorer les résultats de reconnaissances. Néanmoins, il est dans un premier temps nécessaire

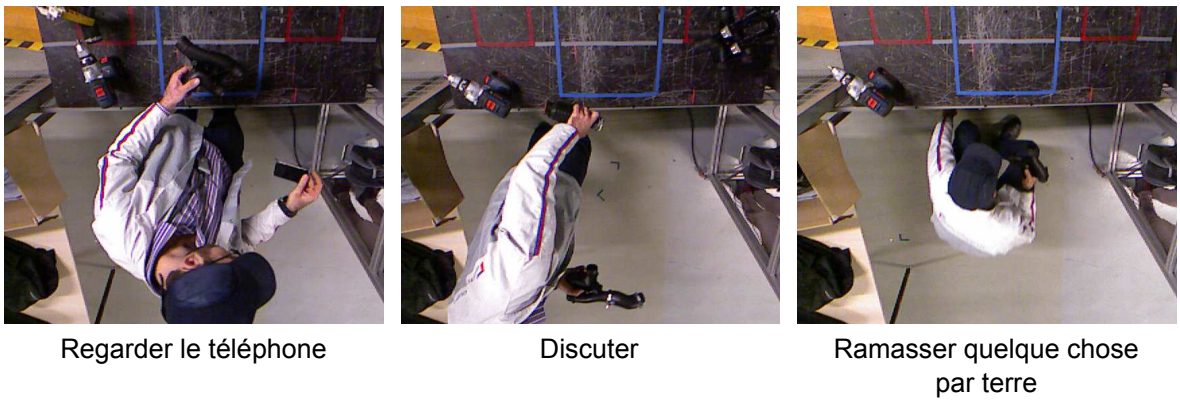


FIGURE 6.18 – Exemples de gestes parasites

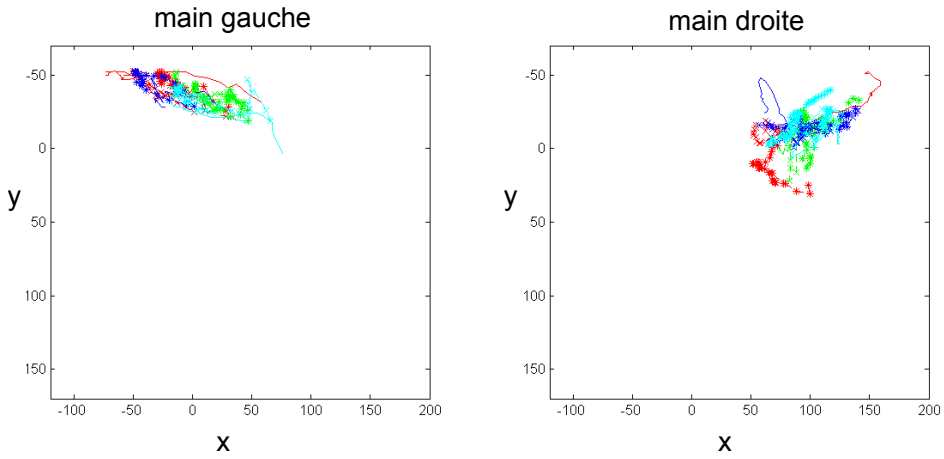


FIGURE 6.19 – Projection, sur le plan XY, dans le repère lié à la scène, des trajectoires 3D des mains gauche et droite lors de l'exécution de gestes parasites

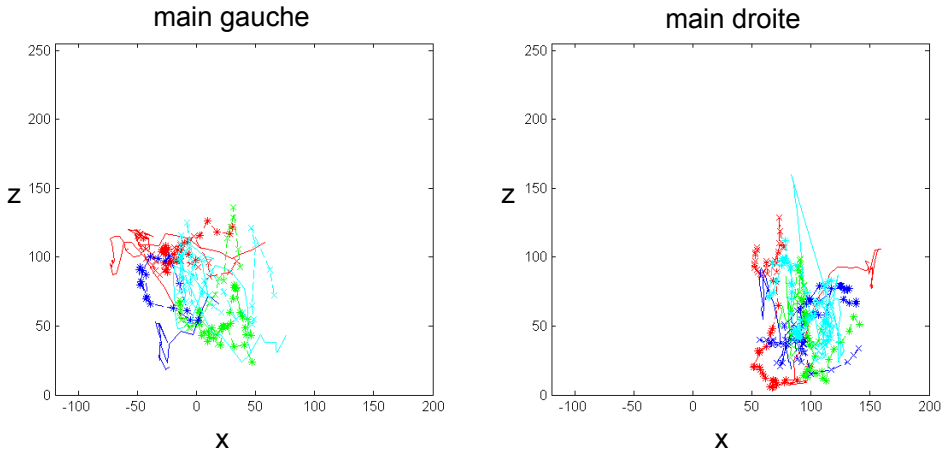


FIGURE 6.20 – Projection, sur le plan XZ, dans le repère lié à la scène, des trajectoires 3D des mains gauche et droite lors de l'exécution de gestes parasites

d'avoir un système de reconnaissance basé sur la vision robuste, qui pourra ensuite bénéficier de ces améliorations.

6.5.2 Réaction de notre système de reconnaissance face à des gestes parasites

Pour pouvoir tester notre système de reconnaissance, lors des enregistrements nous avons demandé aux opérateurs de faire des gestes "parasites", c'est à dire faire une action qui n'est pas prévue dans le processus d'assemblage des pièces de moteur. Pour avoir le plus grand nombre possible de gestes parasites différents, nous ne leur avons pas demandé de faire des gestes particuliers . Les opérateurs ont spontanément fait semblant de téléphoner, de saluer un collègue, de nettoyer leurs lunettes, de ramasser quelque chose par terre ou de tomber, par exemple. Nous avons enregistré au total 34 de ces gestes parasites. En conditions réelles, l'exécution de ce type d'actions est plutôt rare, mais elles peuvent arriver, notamment si l'opérateur fait un malaise par exemple.

Dans un premier temps, nous avons observé comment se comportait notre système face à ces gestes. Nous avons donc effectué des reconnaissances des séquences continues de gestes dans lesquelles se trouvaient les gestes parasites. Nous avons utilisés les K-Means et HMMs appris sur des gestes isolés avec comme vecteurs-descripteurs les positions 3D des mains dans le référentiel lié à la scène. Nous avons choisi d'utiliser une fenêtre de largeur 30 vecteurs-descripteurs pour faire la reconnaissance. Nous obtenons les résultats décrits dans le Tableau 6.24

TABLEAU 6.24 – Sortie(s) notre système (geste reconnu) lors de l'exécution de gestes parasites avec des données provenant seulement de la vision en utilisant le critère d'évaluation *jackknife*

	0	**G1**	**G2**	**G3**	**G4**	**G5**	**Total de faux-positifs**
Sortie(s) notre système lors de l'exécution des gestes parasites	**10**	3	4	10	8	13	**38**

Nous considérons que la sortie de notre système est nulle si aucun geste n'a été reconnu pendant tout de temps de l'exécution du geste parasite. Dans les autres cas, plusieurs gestes peuvent être reconnus pendant que l'opérateur effectue le geste parasite, c'est pourquoi il y a plus de 34 gestes reconnus dans le Tableau 6.24.

Près de 1/3 des gestes parasites n'ont pas menés à de faux-positifs, 10 sur 34, mais de nombreuses fausses reconnaissances ont tout de même eu lieu, 38 au total. Notamment avec le geste 3, assembler, le geste 4 visser et surtout le geste 5, poser la pièce terminée. Pour ce dernier geste, cela peut s'expliquer par le fait que lors de son exécution l'opérateur peut partiellement sortir du champs d'acquisition de la caméra. C'est également ce qui se produit lorsque l'opérateur fait semblant de tomber ou prétend aller saluer un collègue un peu plus loin sur la chaîne de montage. Notre système ne semble donc pas assez robuste pour faire face à ces gestes parasites.

La Tableau 6.25 présente les résultats de précisions et de rappels que nous obtenons ces séquences de gestes comportant des gestes parasites.

Si nous comparons avec les résultats obtenus sur les séquences sans gestes parasites en utilisant les mêmes informations provenant des cartes de profondeur et les mêmes configurations de K-Means et HMMs, Tableau 6.10, nous pouvons observer une chute des

TABLEAU 6.25 – Résultats de reconnaissance en temps réel en utilisant le critère *jackknife* sur des séquences de gestes comprenant des gestes parasites (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
30	**68%**	70%	67%	58%	43%	56%	56%	62%	72%	63%	67%	80%

taux de précision des gestes 2, 3, 4 et 5, ce qui est en accord avec les fausses reconnaissances des gestes parasites. La précision moyenne chute donc également, avec un taux de 56%. Dans une moindre mesure, les taux de rappel des gestes 2 et 5 sont également en baisse, entrainant un taux de reconnaissance total de 68%, soit 6% de moins que pour les séquences sans gestes parasites. En effet, lorsque les opérateurs font des gestes parasites, ils divisent parfois un geste à reconnaitre en y incorporant le geste parasite, empêchant alors la reconnaissance du geste technique.

Nous avons également évalué notre système une fois que nous l'avons adapté à l'opérateur en ajoutant des exemples de ses gestes dans la base d'apprentissage, 5 sets de gestes, et en utilisant les données provenant du capteur inertiel sur la visseuse. Les résultats sont présentés Tableau 6.26.

TABLEAU 6.26 – Résultats de reconnaissance en temps réel en utilisant le critère *jackknife*, avec une adaptation de la base d'apprentissage avec 5 sets de gestes, sur des séquences de gestes comprenant des gestes parasites en utilisant des données provenant de la vision et du capteur inertiel (RC : taux de reconnaissances correctes)

Taille de la fenêtre temporelle	**RC**	**Précision**						**Rappel**				
		P_1	P_2	P_3	P_4	P_5	$\overline{P}$	R_1	R_2	R_3	R_4	R_5
30	**87%**	56%	63%	68%	72%	56%	65%	69%	81%	89%	97%	96%

Nous pouvons comparer ces résultats avec ceux du Tableau 6.22, obtenus avec des séquences de gestes ne comportant pas de gestes parasites. Le taux de reconnaissances correctes reste plutôt élevé, 87%, soit une baisse de seulement 3%.

Dans un même temps, la précision moyenne, bien que supérieure au résultats sur des séquences de gestes enchaînés sans données du capteur inertiel et sans adaptation de la base de données, 56%, reste relativement faible avec un taux de 65%. Certains faux-positifs ont été corrigés mais pas suffisamment pour avoir un système tout à fait robuste. Nous avions en effet une précision moyenne de 81% sur des séquences de gestes enchaînés sans gestes parasites (Tableau 6.22).

Il est à noter que les séquences de gestes contenant des gestes parasites, sur lesquelles nous venons d'évaluer notre système, le nombre de gestes parasites effectués par les opérateurs est bien supérieur à ce qu'il pourrait se passer en réalité. En effet, sur ces séquences de gestes, il y a au moins un geste parasite par processus d'assemblage de pièce, alors que ces gestes surviennent de manière beaucoup plus ponctuelle sur les chaînes de montage. La dégradation des résultats, due à la présence de ces gestes, ne sera donc pas aussi importante que ce que nous obtenons sur les Tableaux 6.25 et 6.26 .

6.6 Mise en place d'un démonstrateur

Nous avons implémenté un démonstrateur pour la reconnaissance des gestes en temps réel lors de la collaboration avec le robot sur le cas d'étude présenté partie 2.2.2.

Pour piloter le robot du cas d'étude, nous n'avons accès qu'à deux commandes : l'ouverture de la pince gauche et l'ouverture de la pince droite. Jusqu'alors ces pinces étaient commandées grâce à deux boutons, un pour chaque pince. Lors des enregistrements des opérateurs, une personne tierce appuyait sur l'un ou l'autre de ces boutons lorsque l'opérateur souhaitait prendre une pièce dans une des pinces du robot.

Avec notre système, lorsque l'opérateur approche la main droite de la pince droite, si la pince est en attente de donner une pièce, cela déclenche l'ouverture de la pince. De même pour la pince gauche. Pour permettre une communication entre notre système de reconnaissance et le robot, nous avons utilisé une carte Arduino Uno[2]. Nous avons choisi d'utiliser une carte Arduino car il s'agit d'une carte électronique peu onéreuse et facile a programmer. Notre algorithme envoie un signal à l'Arduino lorsque l'on souhaite ouvrir l'une ou l'autre des pinces. L'Arduino met en sortie l'un de ses pins à l'état haut, un pin différent pour chaque pince, pour actionner l'ouverture de la pince, et donc simuler l'appui sur un bouton.

Pour commander l'ouverture des pinces, nous avons préféré utiliser des informations sur les positions des mains plutôt que sur la reconnaissance des gestes pour être sûr que la main est prête à recevoir la pièce. En effet, si le geste est reconnu alors que l'opérateur est en train d'avancer sa main vers la pince, la pince risque de s'ouvrir trop tôt et la pièce tombera alors sur la table. Cela pourrait abîmer la pièce mais serait également source d'un stress supplémentaire pour l'opérateur et ne permettrait pas de créer une collaboration agréable.

En parallèle des commandes pour l'ouverture des pinces, nous faisons apparaître sur un écran la position des mains calculées en temps réel ainsi que la reconnaissance des 5 gestes techniques en temps réel. Une capture d'écran est visible Figure 6.21.

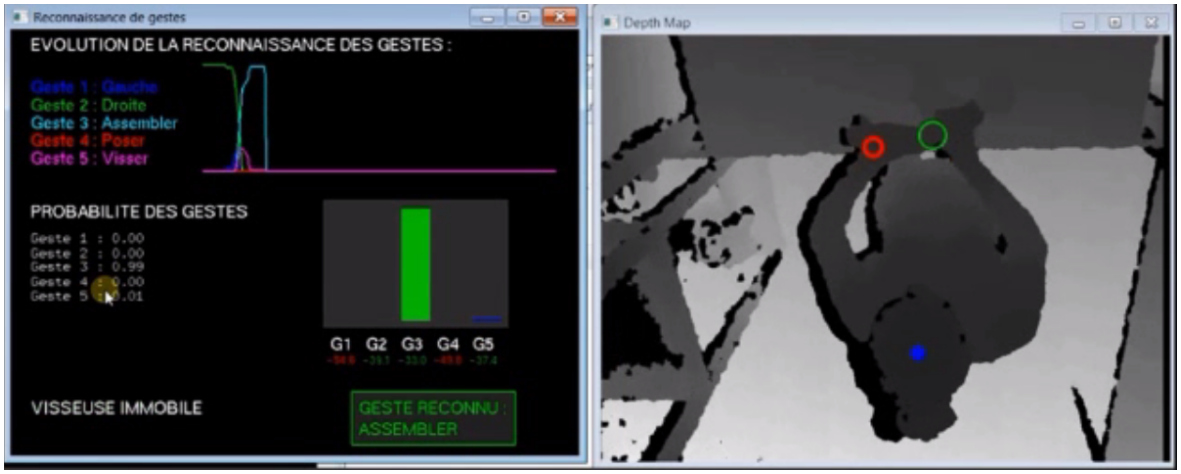


FIGURE 6.21 – Reconnaissance en temps réel et suivi de la position des mains

Sur la fenêtre de gauche, on peut voir, en haut, l'évolution des probabilités de chaque geste au cours des différentes étapes de montage des pièces de moteur. En dessous, on peut observer la probabilité de chaque geste à l'instant *t*, puis enfin sur la dernière ligne à gauche si la visseuse est en mouvement et à droite le geste technique qui est reconnu. Sur

2. https ://www.arduino.cc/

la fenêtre de droite, on peut suivre les positions des mains (ronds rouges et verts) et de la tête (rond bleu) de l'opérateur.

Ce démonstrateur nous a permis de créer un première interaction avec le robot. Grâce à ce système, la collaboration entre l'opérateur et le robot a été fluidifiée. La reconnaissance des gestes en temps réel pourrait permettre au robot d'avoir accès à plus d'informations, notamment pour adapter sa vitesse. On pourrait, par exemple, faire une vérification automatique que toutes les actions de montage ont bien été effectuées, notamment le vissage qui pourrait être parfois oublié.

6.7 Conclusion et synthèse

Dans ce chapitre, nous avons présenté notre système de reconnaissance de gestes. Nous avons étudié le cas d'étude collaboratif présenté partie 2.2.2. Après avoir étudié plusieurs descripteurs, nous avons décidé de conserver ceux décrivant au mieux les gestes, c'est à dire les positions 3D des mains. Nous utilisons deux critères d'évaluation. Le premier critère, *jackknife* évalue le système si l'opérateur test n'a pas d'exemple de ses gestes dans la base d'apprentissage, c'est à dire lorsqu'un nouvel opérateur arrive sur le poste collaboratif et n'est pas connu du système. Le deuxième critère, 80%-20%, évalue un système qui possède des gestes de l'opérateur test dans la base d'apprentissage.

Dans un premier temps, nous obtenons, pour des séquences de gestes enchaînés, 74% de reconnaissances correctes en utilisant le critère *jackknife* et 83% en utilisant le critère 80%-20%. Par la suite, nous avons étudié une méthode pour adapter la base d'apprentissage à un nouvel opérateur en y injectant un faible nombre d'exemples de gestes. Nous avons également équipé la scène en fixant un capteur inertiel sur un outil, la visseuse, afin d'affiner nos résultats de reconnaissance. Nous obtenons alors 90% de reconnaissances correctes sur des séquences de gestes enchaînés. Par la suite, nous avons observé que l'exécution de gestes inattendus baissait les performances de notre système, 87% de reconnaissances correctes lorsqu'un un geste parasite survient pour chaque cycle d'assemblage. Cette baisse reste faible, surtout si on considère que ces gestes se produisent de manière beaucoup plus ponctuelle en usine.

La Tableau 6.27 récapitule l'ensemble des résultats que nous obtenons avec le critère du *jackknife* sur des séquences de gestes enchaînés.

TABLEAU 6.27 – Tableau récapitulatif de nos résultats sur des séquences de gestes enchaînés avec le critère *jackknife*

	Données vision	Données vision + Capteur inertiel	Données vision + Capteur inertiel + Adaptation avec 5 sets de gestes	Données vision + Capteur inertiel + Adaptation avec 5 sets de gestes + Gestes parasites
Taux de reconnaissances correctes	74%	77%	90%	87%

Pour conclure, le résultat de 90% de reconnaissances correctes est un résultat très satisfaisant au vu de la variabilité des gestes que nous avons en entrée du système. Cela permet de pouvoir envisager une collaboration fluide entre un opérateur et un robot, sans que l'opérateur ait besoin de modifier ses méthodes de travail.

Cette étude nous a également permis de formaliser une méthodologie pour reconnaître des gestes techniques en temps réel avec des capteurs non intrusifs. Nous pouvons la détailler en plusieurs étapes :

1. La première étape consiste à choisir les gestes a reconnaître. Pour une collaboration avec un robot en milieu industriel, il faut mettre en évidence des gestes qui découpent les tâches en plusieurs étapes qui doivent être comprises par le robot afin qu'il puisse se synchroniser avec l'opérateur.
2. La deuxième étape revient à choisir le capteur et son positionnement en prenant en compte les contraintes du milieu industriel. Nous nous sommes orientés vers une caméra de profondeur capteur non intrusif qui donne des informations sur la géométrie de la scène. Il est nécessaire de s'assurer que son positionnement permet d'éviter le plus grand nombre d'occultations. Nous avons choisi d'utiliser une vue de dessus.
3. Une étape importante est l'acquisition de la base de données. Il est nécessaire d'avoir un nombre important d'opérateurs, au dessus de 10, pour être robuste aux variabilités dans l'exécution des gestes.
4. Pour décrire le mouvement, il est préférable de choisir des descripteurs basés sur le geste effectif, les positions des mains dans notre cas et dans le plupart des cas de reconnaissance de gestes techniques, plutôt que des descripteurs représentant la posture générale de la personne filmée.
5. Nous avons choisi d'utiliser des HMMs discrets pour apprendre et reconnaître les gestes effectués par les opérateurs. Il faut donc dans un premier temps choisir leurs paramètres, c'est à dire le nombre de clusters et le nombre d'états cachés des HMMs. Pour cela nous avons testé plusieurs configurations possibles et sélectionné celle donnant les meilleurs résultats.
6. Une instrumentation des outils peut affiner les résultats de reconnaissance. Fixer des capteurs inertiels sur des outils permet de savoir lorsqu'ils sont utilisés. Aujourd'hui, les outils sur les chaînes de montage sont pour la plupart déjà connectés, il est possible de savoir lorsqu'ils sont utilisés. Dans le cas réel, il suffira de prendre en compte directement cette information sans avoir à ajouter des capteurs.
7. Si cela est possible, une adaptation de la base d'apprentissage serait à privilégier pour améliorer les résultats de reconnaissance. On peut envisager que lorsqu'un nouvel opérateur arrive sur le poste collaboratif, on lui demande d'enregistrer plusieurs de ses gestes techniques qui seront alors ajoutés à une base de données déjà existante, regroupant des gestes de plusieurs opérateurs. En effet, avoir comme base d'apprentissage seulement les gestes du nouvel opérateur mènerait vers un système moins robuste aux variations dans l'exécution des gestes.

Toutes ces étapes décrivent une méthodologie pour mettre en place une reconnaissance de gestes techniques sur une chaîne de montage. Elles peuvent être appliquées à d'autres cas d'étude, en co-présence ou en collaboration.

Chapitre 7

Conclusions et perspectives

La robotique collaborative est un domaine en pleine expansion. La présence de robots est de plus en plus fréquente dans nos vies quotidiennes, et cela a tendance à croître. En milieu industriel, les robots collaboratifs ont également fait leur entrée dans les usines. Comparés aux robots industriels isolés, déjà utilisés depuis de nombreuses années, les robots industriels collaboratifs travaillent dans le même espace que les opérateurs, permettent de rendre une usine plus flexible, de compléter les compétences d'un opérateur ou de faire des tâches qui pouvaient mener à des troubles musculo-squelettiques.

Néanmoins l'utilisation de ces robots en milieu industriel provoque des changements dans l'organisation de l'usine et dans la manière de travailler des opérateurs. Tout d'abord, la sécurité de l'opérateur qui travaille avec le robot doit être garantie. Aujourd'hui, de nombreuses avancées technologiques ont permis de rendre les robots intrinsèquement sûrs et des systèmes peuvent être mis en place, des barrières lasers par exemple, pour éviter tout contact entre le robot et l'opérateur. Une deuxième problématique est la réaction des opérateurs qui doivent maintenant travailler avec des robots qu'on leur avait auparavant décrits comme dangereux. Cet aspect a été traité dans le premier axe de recherche de la chaire « PSA Peugeot Citroën - Robotique et Réalité Virtuelle ». Enfin, une dernière problématique est comment rendre les robots intelligents pour qu'ils puissent collaborer efficacement avec des opérateurs ? Dans cette thèse, nous avons décidé d'utiliser la reconnaissance de gestes techniques pour permettre aux robots de mieux comprendre leur environnement. En effet, avec la reconnaissance de gestes, les robots seront capables de savoir quelle tâche vient d'être effectuée par l'opérateur, comprendre si quelque chose d'anormal survient et adapter leur allure.

Nous avions à notre disposition deux cas d'étude inspirés de postes réels sur les chaînes de montage de PSA Peugeot Citröen. Le premier cas d'étude était un scénario de co-présence où le robot et l'opérateur travaillent côte à côte sur la pose d'éléments sur la portière d'une voiture (des clips, des obturateurs, un lécheur de vitre et une feuille d'étanchéité). Le deuxième cas d'étude portait sur un scénario collaboratif où l'opérateur et le robot travaillent ensemble pour assembler des pièces de moteur. Le robot passe des pièces à l'opérateur grâce à ses pinces tandis que l'opérateur les assemble et les visse. Ces deux cas d'étude comportaient des défauts lorsqu'ils n'étaient pas assistés par un robot collaboratif. Pour le scénario de co-présence, la pose de la feuille d'étanchéité menait à des troubles musculo-squelettiques. Pour le scénario de collaboration, la recherche des pièces appropriées parmi d'autres était une charge cognitive supplémentaire pour l'opérateur et également une perte de temps dans le montage des pièces.

Nous avons utilisé le premier cas d'étude pour étudier la faisabilité de la reconnaissance des gestes, Chapitre 4. Pour cela, nous avons choisi d'utiliser des données fiables,

et nous nous sommes donc orientés vers des capteurs inertiels portés par l'opérateur. En utilisant le *Gesture Follower* ([11] et [12]) pour apprendre et reconnaître les gestes, nous atteignons, dans le cas mono-opérateur, un très bon score de 96% de reconnaissances correctes avec l'ensemble de capteurs du vêtement *Animazoo*. Cette étude nous a conforté dans la faisabilité de la reconnaissance de gestes. Il fallait l'étendre au cas multi-opérateurs et également, pour mieux répondre aux contraintes du milieu industriel, privilégier l'utilisation de capteurs non intrusifs.

Nous avons donc choisi d'utiliser une caméra de profondeur avec une vue de dessus. La vue de haut permet d'éviter un grand nombre d'occultations pouvant survenir sur les chaînes de montage. Nous devons, dans un premier temps, extraire des informations des images de profondeur que nous obtenons. Pour cela, nous avons mis au point un algorithme de suivi des mains, Chapitre 5. Nous nous sommes basés sur la topologie du corps humain en faisant l'hypothèse que les deux parties du haut du corps qui sont le plus éloignées du haut de la tête sont les mains. Nous avons utilisé l'algorithme de Dijkstra pour calculer la distance de chaque point du haut du corps à la tête afin de détecter la position des mains pour ensuite les suivre. Une fois ces informations extraites, nous avons étudié la reconnaissance de gestes techniques sur le second cas d'étude, avec une scénario de collaboration, Chapitre 6.

Pour apprendre et reconnaître les gestes techniques, nous avons choisi d'utiliser des HMMs discrets. Nous disposons d'enregistrements de 13 opérateurs. Dans un premier temps, nous nous sommes intéressés aux descripteurs du geste. Nous nous sommes rendu compte que nous obtenons de meilleurs résultats de reconnaissance si nous nous concentrons sur un descripteur qui représente la partie effective du geste, c'est à dire ce qui réalise directement l'action à reconnaître, des mouvements de la main dans notre cas, au lieu de descripteurs qui correspondent à une posture plus globale de l'opérateur. Nous avons utilisé deux critères pour évaluer notre système de reconnaissance. Pour le premier critère, *jackknife*, nous évaluons le système lorsqu'un nouvel opérateur arrive, c'est à dire qu'aucun de ses gestes a été utilisé pour apprendre les HMMs. Pour le second critère, appelé 80%-20%, le système connait l'opérateur qui est en train d'effectuer des gestes. Des exemples de ses gestes ont été utilisés pour apprendre les HMMs. Nous obtenons, comme attendu, de meilleurs résultats avec le second critère qu'avec le premier, sur des gestes isolés et sur des séquences de gestes enchaînés. Nous avons donc décidé d'étudier une adaptation du système à un nouvel opérateur. Nous voulions quantifier le nombre de gestes nécessaires à ajouter dans la base d'apprentissage pour adapter le système à un nouvel opérateur. Avec une faible quantité, 5 exemples de chaque geste, nous arrivons à sensiblement améliorer le taux de reconnaissances correctes, de plus de 4%. Nous observons également que notre système de reconnaissance est meilleur quand l'ensemble d'apprentissage contient aussi des réalisations des gestes effectués par d'autres opérateurs plutôt que lorsqu'il est strictement mono-opérateur. Pour encore améliorer nos reconnaissances, nous avons décidé d'équiper la scène avec des capteurs inertiels. Nous avons fixé un capteur inertiel sur la visseuse afin de savoir quand celle-ci est utilisée. Ce système permet de simuler l'utilisation d'informations qui sont déjà disponibles dans les usines, les outils étant déjà connectés. En combinant l'utilisation des données provenant de la vision et du capteur inertiel, et en adaptant la base d'apprentissage à l'opérateur, nous avons atteint 90% de reconnaissances correctes aussi bien sur des gestes isolés que sur des séquences de gestes enchaînés. Enfin, nous avons étudié l'impact de gestes inattendus sur les performances de notre système. En effet, les opérateurs peuvent effectuer des gestes qui ne sont pas censés se produire lors de l'exécution de leurs tâches : nettoyer leurs lunettes, regarder leur téléphone ou bien tomber en sont des exemples. L'exécu-

tion de ce type de geste ne diminue que légèrement les performances de notre système (87% de bonnes reconnaissances au lieu de 90%, même si la fréquence de gestes inattendus/parasites est beaucoup plus élevée que dans la réalité).

Finalement, le très bon score de 90% de reconnaissances correctes montre que l'utilisation de la reconnaissance de gestes techniques pour permettre une collaboration homme-robot sur les chaînes de montage est possible et envisageable. Nous avons d'ailleurs implémenté notre solution et utilisé le suivi des mains pour commander l'ouverture des pinces du robot afin de fluidifier le travail entre le robot et l'opérateur.

Lors de cette thèse plusieurs contributions ont été faites :

- Nous avons établi la faisabilité d'une reconnaissance de gestes techniques multi-opérateurs en utilisant des capteurs non intrusifs,
- Nous avons mis au point un algorithme de suivi des mains avec une vue de dessus et une caméra de profondeur se basant sur les distances géodésiques,
- Nous avons mis en place une méthodologie complète pour la reconnaissance de gestes techniques (choix des gestes à reconnaître, choix des capteurs, suivi des mains, algorithme de reconnaissance des gestes),
- Nous avons mis en évidence que l'utilisation d'un descripteur ne prenant en compte que la partie effective du geste, les positions des mains dans notre cas, plutôt que décrivant une posture générale de la personne les effectuant, suffit et même donne de meilleurs résultats pour reconnaître des gestes techniques,
- Nous avons étudié l'impact de l'instrumentation de la scène afin d'améliorer les résultats de reconnaissance. Fixer des capteurs inertiels sur des outils permet d'obtenir des informations complémentaires pour affiner la reconnaissance des gestes techniques et permet de lever certaines ambiguïtés lors de ces reconnaissances,
- Nous avons mis en place une méthodologie pour adapter un système de reconnaissance à un nouvel opérateur et mis en évidence qu'un faible nombre d'exemples de gestes de l'opérateur test ajoutés dans la base d'apprentissage permet de significativement améliorer le taux de reconnaissance des gestes (+4% d'amélioration pour 5 exemples de chaque geste),
- Nous avons mis en évidence que l'utilisation d'une base de données multi-opérateurs permet d'avoir un système de reconnaissance plus robuste, et donc d'avoir des meilleurs résultats de reconnaissance qu'un système appris, et reconnu, avec une base de données mono-opérateur

Des améliorations sur notre système de reconnaissance de gestes peuvent être apportées. Dans un premier temps, une meilleure prise en compte des gestes inattendus serait souhaitable pour éviter des fausses reconnaissances de la part du robot.

Une autre méthode aurait pu être testée pour le suivi des mains. Nous nous sommes orienté vers un algorithme qui ne nécessitait pas d'apprentissage. Nous aurions également pu comparer les performances que nous obtenons en testant un algorithme avec apprentissage, comme une adaptation de la squelettisation selon Shotton et al. [117] avec une vue de haut.

Il serait également intéressant, dans notre simulateur, d'échanger d'autres d'informations avec le robot en plus de la proximité des mains avec les pinces afin de créer une plus grande interaction.

Enfin, il aurait été intéressant de tester notre système sur d'autres cas d'étude pour pouvoir mieux apprécier sa généralisation sur les chaînes de montage.

Bibliographie

[1] Catherine Achard, Xingtai Qu, Arash Mokhber, and Maurice Milgram. A novel approach for recognition of human actions with semi-global features. *Machine Vision and Applications*, 19(1) :27–34, jan 2008. 38, 46, 54, 66

[2] Jake K. Aggarwal and Michael S. Ryoo. Human activity analysis. *ACM Computing Surveys*, 43(3) :1–43, apr 2011. 39

[3] Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Raja Chatila. Task planning for human-robot interaction. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence innovative context-aware services : usages and technologies - sOc-EUSAI '05*, page 81, New York, New York, USA, 2005. ACM Press. 25

[4] Peter Anderson-Sprecher and Reid Simmons. Voxel-based motion bounding and workspace estimation for robotic manipulators. *2012 IEEE International Conference on Robotics and Automation*, pages 2141–2146, 2012. 21, 22

[5] Sai K. Banala, Seok Hun Kim, Sunil K. Agrawal, and John P. Scholz. Robot Assisted Gait Training With Active Leg Exoskeleton (ALEX). *IEEE transactions on neural systems and rehabilitation engineering*, 17(1) :2–8, feb 2009. 15

[6] Alexander Bannat, Thibault Bautze, Michael Beetz, Juergen Blume, Klaus Diepold, Christoph Ertelt, Florian Geiger, Thomas Gmeiner, Tobias Gyger, Alois Knoll, Christian Lau, Claus Lenz, Martin Ostgathe, Gunther Reinhart, Wolfgang Roesel, Thomas Ruehr, Anna Schuboe, Kristina Shea, Ingo Stork genannt Wersborg, Sonja Stork, William Tekouo, Frank Wallhoff, Mathey Wiesbeck, and Michael F. Zaeh. Artificial Cognition in Production Systems. *IEEE Transactions on Automation Science and Engineering*, 8(1) :148–174, jan 2011. 25

[7] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human–robot collaboration : a survey. *International Journal of Humanoid Robotics*, 5(01) :47–66, 2008. 14

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3) :346–359, jun 2008. 45

[9] Ari Y. Benbasat and Joseph A. Paradiso. An Inertial Measurement Framework for Gesture Recognition and Applications. In *International Gesture Workshop*, pages 9–20. Springer Berlin Heidelberg, 2002. 51

[10] Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1) :1–127, 2009. 58

[11] Frederic Bevilacqua, Fabrice Guédy, Norbert Schnell, Emmanuel Fléty, and Nicolas Leroy. Wireless sensor interface and gesture-follower for music pedagogy. In *Proceedings of the 7th international conference on New interfaces for musical expression - NIME '07*, page 124, New York, New York, USA, jun 2007. ACM Press. 70, 132

[12] Frédéric Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, Fabrice Guédy, and Nicolas Rasamimanana. *Gesture in Embodied Communication and Human-Computer Interaction*, volume 5934 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, feb 2010. 70, 71, 132

[13] Rainer Bischoff, Johannes Kurth, Gunter Schreiber, Ralf Koeppe, Alin Albu-Schaffer, Alexander Beyer, Oliver Eiberger, Sami Haddadin, Gerhard Stemmer, Andreas Grunwald, and Others. The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing, 2010. 17, 22

[14] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1395–1402 Vol. 2. IEEE, 2005. 38, 46, 54, 56

[15] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3) :257–267, mar 2001. 38, 46

[16] Lubomir Bourdev and Jitendra Malik. *Poselets : Body Part Detectors Trained Using 3D Human Pose Annotations*. IEEE, sep 2009. 38, 47, 49, 54

[17] Matteo Bregonzio, Shaogang Gong, and Tao Xiang. Recognising action as clouds of space-time interest points. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1948–1955. IEEE, jun 2009. 38, 44, 54, 56

[18] Nicolas Bremard, Laurent Grisoni, and Bruno De Araujo. Interaction events in contactless gestural systems. In *Proceedings of the 2014 International Workshop on Movement and Computing*, pages 166–169, New York, New York, USA, 2014. ACM Press. 10

[19] Blandine Bril. Description du geste technique : Quelles méthodes ? *Techniques & culture. Revue semestrielle d'anthropologie des techniques*, 3(54-55) :242–244, jun 1984. 37

[20] Blandine Bril and Valentine Roux. Le geste technique. Réflexions méthodologiques et anthropologiques. 2002. 37

[21] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3) :1–33, jan 2014. 38, 51

[22] Claude Cadoz. Le geste canal de communication homme/machine : la communication 'instrumentale'. *Technique et Science Informatiques*, 13(1) :31–61, 1994. 36

[23] Sylvain Calinon and Aude Billard. Stochastic Gesture Production and Recognition Model for a Humanoid Robot. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, pages 2769–2774, 2004. 66

[24] Balasubramaniyan Chandrasekaran and James M. Conrad. Human-robot collaboration : A survey. In *SoutheastCon 2015*, pages 1–8. IEEE, apr 2015. 18

[25] Lulu Chen, Hong Wei, and James Ferryman. A survey of human motion analysis using depth imagery. 2013. 38, 48

[26] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40 :1–13, 2016. 24

[27] Andrea Cherubini, Robin Passama, Arnaud Meline, Andre Crosnier, and Philippe Fraisse. Multimodal control for human-robot cooperation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2202–2207. IEEE, nov 2013. 25

[28] Herbert H Clark and Susan E Brennan. Grounding in communication. *Perspectives on socially shared cognition*, 13(1991) :127–149, 1991. 10

[29] J. Edward Colgate, Michael A. Peshkin, and Witaya Wannasuphoprasit. Cobots : Robots For Collaboration With Human Operators. *In Proceedings of the ASME Dynamic Systems and Control Division*, 1996. 15

[30] M D Cooney, C Becker-Asano, T Kanda, A Alissandrakis, and H Ishiguro. Full-body gesture recognition using inertial sensors for playful interaction with small humanoid robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2276–2282. IEEE, oct 2010. 52

[31] Juan Antonio Corrales Ramón, Gabriel Jesús García Gómez, Fernando Torres Medina, and Véronique Perdereau. *Cooperative tasks between humans and robots in industrial environments*. InTech, 2012. 24

[32] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human Detection using Oriented Histograms of Flow and Appearance. *European Conference on Computer Vision (ECCV '06)*, 3952 :428–441, 2006. 45

[33] Matthias Dantone, Juergen Gall, Christian Leistner, and Luc Van Gool. Human Pose Estimation Using Body Parts Dependent Joint Regressors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3041–3048, 2013. 38, 47

[34] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1) :269–271, 1959. 80

[35] Pitor Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior Recognition via Sparse Spatio-Temporal Features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. IEEE, 2005. 38, 43, 44, 54, 55

[36] Liang Dong, Jiankang Wu, and Xiang Chen. A Body Activity Tracking System using Wearable Accelerometers. *2007 IEEE International Conference on Multimedia and Expo*, pages 1011–1014, 2007. 38, 51, 54

[37] Yong Du, Wei Wang, and Liang Wang. Hierarchical Recurrent Neural Network for Skeleton Based Action Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, 2015. 54, 59

[38] Markus Fischer and Dominik Henrich. 3D Collision Detection for Industrial Robots and Unknown Obstacles Using Multiple Depth Images. In *Advances in Robotics Research*, pages 111–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 21

[39] Cliff Fitzgerald. Developing baxter. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6. IEEE, apr 2013. 18

[40] Terrence Fong, Chris Provencher, Mark Micire, Myron Diftler, Reginald Berka, Bill Bluethmann, and David Mittman. The Human Exploration Telerobotics project : Objectives, approach, and testing. In *2012 IEEE Aerospace Conference*, pages 1–9. IEEE, mar 2012. 16

[41] Hironobu Fujiyoshi and Alan J. Lipton. Real-time human motion analysis by image skeletonization. In *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98*, pages 15–21. IEEE Comput. Soc, 1998. 38, 47

[42] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 755–762. IEEE, jun 2010. 38, 50

[43] D. M. Gavrila and L. S. Davis. Towards 3-D model-based tracking and recognition of human movement : a multi-view approach. *In International Workshop on Automatic Face- and Gesture-Recognition. IEEE Computer Society*, pages 272–277, 1995. 38, 54, 61

[44] Nicholas Gillian, R Benjamin Knapp, and Sile O 'modhrain. Recognition Of Multivariate Temporal Musical Gestures Using N-Dimensional Dynamic Time Warping. *NIME*, pages 337–342, 2011. 61

[45] Nicholas Gillian and Joseph A Paradiso. The Gesture Recognition Toolkit. *Journal of Machine Learning Research*, 15 :3483–3487, 2014. 102

[46] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. Safe Physical Human-Robot Interaction : Measurements, Analysis and New Insights. In *Robotics research*, pages 395–407. Springer Berlin Heidelberg, 2010. 23

[47] Martin Hägele, Walter Schaaf, and Evert Helms. Robot Assistants at Manual Workplaces : Effective Co-operation and Safety Aspects. *Proceedings of the 33rd ISR (International Symposium on Robotics)*, 7-11, 2002. 19, 20

[48] Ronald Ham, Thomas Sugar, Bram Vanderborght, Kevin Hollander, and Dirk Lefeber. Compliant actuator designs. *IEEE Robotics & Automation Magazine*, 16(3) :81–94, sep 2009. 16

[49] M. Hans, B. Graf, and R.D. Schraft. Robotic home assistant Care-O-bot : past-present-future. In *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, pages 380–385. IEEE, 2002. 18

[50] Chris Harris and Mike Stephens. A Combined Corner and Edge Detection. *The Fourth Alvey Vision Conference (1988)*, pages 147 – 151, 1988. 43

[51] Thomas Helten, Meinard Muller, Hans-Peter Seidel, and Christian Theobalt. Real-Time Body Tracking with One Depth Camera and Inertial Sensors. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1105–1112, 2013. 38, 53

[52] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7) :1527–54, jul 2006. 58

[53] Guy Hoffman and Cynthia Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *Proceeding of the ACM/IEEE international conference on Human-robot interaction - HRI '07*, page 1, New York, New York, USA, 2007. ACM Press. 24

[54] Brian Holt, Eng Jon Ong, Helen Cooper, and Richard Bowden. Putting the pieces together : Connected Poselets for human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1196–1201. IEEE, nov 2011. 38, 49

[55] Jianying Hu, Sok Gek Lim, and Michael K. Brown. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33(1) :133–147, 2000. 61

[56] Ioannis Iossifidis, Carsten Bruckhoff, Christoph Theis, Claudia Grote, Christian Faubel, and Gregor Schoner. CORA : An anthropomorphic robot assistant for human environment. In *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, pages 392–398. IEEE, 2002. 17

[57] Mihir Jain, Herve Jegou, and Patrick Bouthemy. Better Exploiting Motion for Better Action Recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2555–2562. IEEE, jun 2013. 38, 45, 54, 56

[58] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1) :221–231, jan 2013. 54, 58

[59] Hao Jiang and David R. Martin. Finding Actions Using Shape Flows. In *Computer Vision – ECCV 2008*, pages 278–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 47

[60] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2) :201–211, jun 1973. 47

[61] Holger Junker, Oliver Amft, Paul Lukowicz, and Gerhard Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6) :2010–2024, 2008. 51

[62] Byeongkeun Kang, Subarna Tripathi, and Truong Q. Nguyen. Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 136–140. IEEE, nov 2015. 59

[63] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1725–1732, 2014. 54, 58, 59

[64] Hami Kazerooni, Jean-Louis Racine, Lihua Lihua Huang, and Ryan Steger. On the Control of the Berkeley Lower Extremity Exoskeleton (BLEEX). In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4353–4360. IEEE, 2005. 15

[65] Yan Ke, Rahul Sukthankar, and Martial Hebert. Spatio-temporal Shape and Flow Correlation for Action Recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, jun 2007. 38, 47, 54, 56

[66] Adam G. Kirk, James FO. O'Brien, and David A. Forsyth. Skeletal Parameter Estimation from Optical Motion Capture Data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 782–788. IEEE, 2005. 52

[67] Günther Knoblich and Jerome Scott Jordan. Action coordination in groups and individuals : Learning anticipatory control. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 29(5) :1006–1016, 2003. 24

[68] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-Flight Sensors in Computer Graphics. *Proc. Eurographics (State-of-the-Art Report)*, 6, 2009. 41

[69] Kazuhiro Kosuge, Manabu Sato, and Norihide Kazamura. Mobile robot helper. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 583–588. IEEE, 2000. 17

[70] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 58

[71] Joerg Krueger, Volker Katschinski, Dragoljub Surdilovic, and Gerhard Schreck. Flexible Assembly Systems through Workplace-Sharing and Time- Sharing Human-Machine Cooperation (PISA), 2010. 17

[72] J. Krüger, R. Bernhardt, D. Surdilovic, and G. Spur. Intelligent Assist Systems for Flexible Assembly. *CIRP Annals - Manufacturing Technology*, 55(1) :29–32, 2006. 17

[73] Jörg Krüger, Terje K. Lien, and Alexander Verl. Cooperation of human and machines in assembly lines. *CIRP Annals - Manufacturing Technology*, 58(2) :628–646, jan 2009. 20

[74] Jörg Krüger, Bertram Nickolay, P Heyer, and Günther Seliger. Image based 3D Surveillance for flexible Man-Robot-Cooperation. *CIRP Annals - Manufacturing Technology*, 54(1) :19–22, 2005. 21

[75] Dana Kulić and Elizabeth Croft. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22(2) :149–164, jan 2007. 20

[76] Ivan Laptev and Tony Lindeberg. Space-time interest points. *Proceedings Ninth IEEE International Conference on Computer Vision*, 1 :432–439, 2003. 38, 43, 44

[77] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, jun 2008. 44, 54, 55

[78] Y Le Cun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in neural information processing systems*, 1990. 57

[79] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998. 57, 58

[80] Hyeon-Kyu Lee and Jin H. Kim. An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10) :961–973, 1999. 38, 54, 66

[81] Suwoong Lee and Yoshiyuki Sankai. Power assist control for walking aid with HAL-3 based on EMG and impedance adjustment around knee joint. In *IEEE/RSJ International Conference on Intelligent Robots and System*, volume 2, pages 1499–1504. IEEE, 2002. 15

[82] Claus Lenz, Markus Grimm, Thorsten Röder, and Alois Knoll. Fusing multiple Kinects to survey shared Human-Robot-Workspaces. *Technische Universität München, Munich, Germany, Tech. Rep. TUM-I1214*, 2012. 22

[83] Claus Lenz, Suraj Nair, Markus Rickert, Alois Knoll, Wolgang Rosel, Jurgen Gast, Alexander Bannat, and Frank Wallhoff. Joint-action for humans and industrial robots for assembly tasks. In *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 130–135. IEEE, aug 2008. 25

[84] André Leroi-Gourhan. *Milieu et techniques : Evolution et techniques*, volume 2000. Albin Michel, 1973. 37

[85] André Leroi-Gourhan. *Le Geste et la Parole : Technique et langage*. Albin Michel, 1989. 37

[86] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uWave : Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6) :657–675, dec 2009. 54, 61

[87] Kui Liu, Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. Fusion of Inertial and Depth Sensor Data for Robust Hand Gesture Recognition. *IEEE Sensors Journal*, 14(6) :1898–1903, jun 2014. 38, 53, 54

[88] Nianjun Liu, Brian C. Lovell, Peter J . Kootsookos, and Richard I.A. Davis. Model Structure Selection and Training Algorithms for an HMM Gesture Recognition System. *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 100–105, 2004. 38, 54, 66

[89] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2) :91–110, nov 2004. 44

[90] Cyrille Migniot and Fakhreddine Ababsa. *3D Human Tracking from Depth Cue in a Buying Behavior Analysis Context.* Number Caip. Computer Analysis of Images and Patterns, 2013. 38, 49

[91] Sushmita Mitra, Senior Member, and Tinku Acharya. Gesture Recognition : A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3) :311–324, 2007. 36

[92] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand Gesture Recognition With 3D Convolutional Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1–7, 2015. 54, 59

[93] Cory Myers, Lawrence R. Rabiner, and Aaron E. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6) :623–635, dec 1980. 60

[94] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. *International Journal of Computer Vision*, 79(3) :299–318, mar 2008. 38, 43, 44, 54, 55

[95] Donald A. Norman. The way I see it : Natural user interfaces are not natural. *interactions*, 17(3) :6, may 2010. 10

[96] Ciarán Ó Conaire, Damien Connaghan, Philip Kelly, Noel E. O'Connor, Mark Gaffney, and John Buckley. Combining inertial and visual sensing for human action recognition in tennis. In *Proceedings of the first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams - ARTEMIS '10*, page 51, New York, New York, USA, 2010. ACM Press. 38, 53, 54

[97] Susanne Oberer and Rolf Dieter Schraft. Robot-Dummy Crash Tests for Robot Safety Assessment. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2934–2939. IEEE, apr 2007. 23

[98] A. Oikonomopoulos, I. Patras, and M. Pantic. Spatiotemporal salient points for visual recognition of human actions. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 36(3) :710–719, jun 2005. 38, 44, 54, 57

[99] Allison M. Okamura. Methods for haptic feedback in teleoperated robotassisted surgery. *Industrial Robot : An International Journal*, 31(6) :499–508, dec 2004. 16

[100] Christian Plagemann, Varun Ganapathi, Daphne Koller, and Sebastian Thrun. Real-time identification and localization of body parts from depth images. In *2010 IEEE International Conference on Robotics and Automation*, pages 3108–3113. IEEE, may 2010. 38, 48, 50

[101] Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Muller, Hans-Peter Seidel, and Bodo Rosenhahn. Multisensor-fusion for 3D full-body human motion capture. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 663–670. IEEE, jun 2010. 38, 53

[102] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6) :976–990, jun 2010. 53

[103] Timo Pylvänäinen. Accelerometer Based Gesture Recognition Using Continuous HMMs. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 639–646. Springer Berlin Heidelberg, 2005. 38, 51, 54

[104] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2) :257–286, 1989. 61, 63, 64

[105] Konstantinos Rapantzikos, Yannis Avrithis, and Spyridon Kollias. Dense saliency-based spatiotemporal feature points for action recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1454–1461. IEEE, jun 2009. 38, 44, 54, 56

[106] Ilaria Renna, Ryad Chellali, and Catherine Achard. Combination of Annealing Particle Filter and Belief Propagation for 3D Upper Body Tracking. *Applied Bionics and Biomechanics*, 9(4) :443–456, 2012. 48

[107] Miguel Reyes, Gabriel Dominguez, and Sergio Escalera. Featureweighting in dynamic timewarping for gesture recognition in depth data. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1182–1188. IEEE, nov 2011. 38, 54, 61

[108] Markus Rickert, Mary Ellen Foster, Manuel Giuliani, Tomas By, Giorgio Panin, and Alois Knoll. Integrating Language, Vision and Action for Human Robot Dialog Systems. In *Universal Access in Human-Computer Interaction. Ambient Interaction*, pages 987–995. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 24, 25

[109] Paul Rybski, Peter Anderson-Sprecher, Daniel Huber, Chris Niessl, and Reid Simmons. Sensor fusion for human safety in industrial workcells. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3612–3619, oct 2012. 21, 22

[110] Rolf Dieter Schraft, Christian Meyer, Christopher Parlitz, and Evert Helms. PowerMate - A safe and intuitive robot assistant for handling and assembly tasks. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 4074–4077. IEEE, 2005. 17, 21

[111] Olivier C. Schrempf, Uwe D. Hanebeck, Andreas J. Schmid, and Heinz Worn. A novel approach to proactive human-robot cooperation. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pages 555–560. IEEE, 2005. 24

[112] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions : a local SVM approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36 Vol.3. IEEE, 2004. 38, 43, 54, 56

[113] Loren Arthur Schwarz, Artashes Mkhitaryan, Diana Mateus, and Nassir Navab. Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing*, 30(3) :217–226, mar 2012. 38, 41, 49, 80, 93

[114] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, page 357, New York, New York, USA, 2007. ACM Press. 45

[115] Samsu Sempena, Nur Ulfa Nur Ulfa Maulidevi, and Peb Ruswono Peb Ruswono Aryan. Human action recognition using Dynamic Time Warping. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–5. IEEE, jul 2011. 38, 54, 61

[116] Jane Shi, Glenn Jimmerson, Tom Pearson, and Roland Menassa. Levels of human and robot collaboration for automotive manufacturing. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems - PerMIS '12*, page 95, New York, New York, USA, mar 2012. ACM Press. 23

[117] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12) :2821–2840, dec 2013. 48, 49, 66, 80, 88, 133

[118] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time Human Pose Recognition in Parts from Single Depth Images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society. 38

[119] Matheen Siddiqui and Gerard Medioni. Human pose estimation from a single view point, real-time range sensor. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 1–8. IEEE, jun 2010. 38, 49

[120] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. 54, 59

[121] Richard Socher, Brody Huval, Bharath Bhat, Christopher D Manning, and Andrew Y Ng. Convolutional-Recursive Deep Learning for 3D Object Classification. In *Advances in Neural Information Processing Systems*, pages 665–673, 2012. 59

[122] Heung-Il Suk, Bong-Kee Sin, and Seong-Whan Lee. Recognizing hand gestures using dynamic Bayesian network. In *2008 8th IEEE International Conference on Automatic Face & Gesture Recognition*, pages 1–6. IEEE, sep 2008. 61

[123] Sy Bor Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden Conditional Random Fields for Gesture Recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, volume 2, pages 1521–1527. IEEE. 61

[124] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1250–1257. IEEE, jun 2012. 38, 54, 66

[125] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient Object Localization Using Convolutional Networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015. 38, 48

[126] Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. *Advances in neural information processing systems*, pages 1799—-1807, 2014. 38, 48

[127] Alexander Toshev and Christian Szegedy. Deeppose : Human pose estimation via deep neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1653—-1660, 2014. 48

[128] Vladimir Vapnik. Statistical learning theory. 1, 1998. 55

[129] Markus Vincze, Wolfgang Zagler, Lara Lammer, Astrid Weiss, Andreas Huber, David Fischinger, Tobias Körtner, Alexandra Schmid, and Christoph Gisinger. Towards a Robot for Supporting Older People to Stay Longer Inde- pendent at Home. In *ISR/Robotik 2014 ; 41st International Symposium on Robotics ; Proceedings of*, pages 1–7, 2014. 18

[130] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense Trajectories and Motion Boundary Descriptors for Action Recognition. *International Journal of Computer Vision*, 103(1) :60–79, mar 2013. 38, 45, 54, 59

[131] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558. IEEE, dec 2013. 38, 45, 54, 59

[132] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *Procedings of the British Machine Vision Conference 2009*, pages 124.1–124.11. British Machine Vision Association, sep 2009. 38, 44, 54, 56

[133] Limin Wang, Yu Qiao, and Xiaoou Tang. Action Recognition With Trajectory-Pooled Deep-Convolutional Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4305–4314, 2015. 54, 59

[134] Witaya Wannasuphoprasit, Prasad Akella, Michael Peshkin, and J. Edward Colgate. Cobots : a novel material handling technology. In *Proceedings of IMECE*, pages 171–178, 1998. 15

[135] Daniel Weinland and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2) :249–257, 2006. 46, 47

[136] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2) :224–241, feb 2011. 38, 39

[137] Vincent Weistroffer. *Étude des conditions d'acceptabilité de la collaboration homme-robot en utilisant la réalité virtuelle.* PhD thesis, Paris, ENMP, 2014. 8, 27

[138] Vincent Weistroffer, Alexis Paljic, Philippe Fuchs, Olivier Hugues, Jean-Paul Chodacki, Pascal Ligot, and Alexandre Morais. Assessing the acceptability of human-robot co-presence on assembly lines : A comparison between actual situations and their virtual reality counterparts. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 377–384. IEEE, aug 2014. 24

[139] Geert Willems, Tinne Tuytelaars, and Luc Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. *European Conference on Computer Vision (ECCV '08)*, 5303 :650–663, oct 2008. 38, 43, 44, 45, 54, 55

[140] Lu Xia, Chia-Chih Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3D joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27. IEEE, jun 2012. 38, 54, 66

[141] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385. IEEE Comput. Soc. Press, 1992. 38, 54, 66

[142] Alper Yilmaz and Mubarak Shah. Actions Sketch : A Novel Action Representation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 984–989. IEEE, 2005. 38, 46

[143] Hong-Min Zhu and Chi-Man Pun. Real-time Hand Gesture Recognition from Depth Image Sequences. *2012 Ninth International Conference on Computer Graphics, Imaging and Visualization*, pages 49–52, jul 2012. 38, 54, 66

Résumé

Les robots collaboratifs sont de plus en plus présents dans nos vies quotidiennes. En milieu industriel, ils sont une solution privilégiée pour rendre les chaînes de montage plus flexibles, rentables et diminuer la pénibilité du travail des opérateurs. Pour permettre une collaboration fluide et efficace, les robots doivent être capables de comprendre leur environnement, en particulier les actions humaines.

Dans cette optique, nous avons décidé d'étudier la reconnaissance de gestes techniques afin que le robot puisse se synchroniser avec l'opérateur, adapter son allure et comprendre si quelque chose d'inattendu survient.

Pour cela, nous avons considéré deux cas d'étude, un cas de co-présence et un cas de collaboration, tous les deux inspirés de cas existant sur les chaînes de montage automobiles.

Dans un premier temps, pour le cas de co-présence, nous avons étudié la faisabilité de la reconnaissance des gestes en utilisant des capteurs inertiels. Nos très bons résultats (96% de reconnaissances correctes de gestes isolés avec un opérateur) nous ont encouragés à poursuivre dans cette voie.

Sur le cas de collaboration, nous avons privilégié l'utilisation de capteurs non-intrusifs pour minimiser la gêne des opérateurs, en l'occurrence une caméra de profondeur positionnée avec une vue de dessus pour limiter les possibles occultations.

Nous proposons un algorithme de suivi des mains en calculant les distances géodésiques entre les points du haut du corps et le haut de la tête. Nous concevons également et évaluons un système de reconnaissance de gestes basé sur des Chaînes de Markov Cachées (HMM) discrètes et prenant en entrée les positions des mains. Nous présentons de plus une méthode pour adapter notre système de reconnaissance à un nouvel opérateur et nous utilisons des capteurs inertiels sur les outils pour affiner nos résultats. Nous obtenons le très bon résultat de 90% de reconnaissances correctes en temps réel pour 13 opérateurs.

Finalement, nous formalisons et détaillons une méthodologie complète pour réaliser une reconnaissance de gestes techniques sur les chaînes de montage.

Mots Clés

Robotique collaborative, Interaction homme-robot, Reconnaissance de gestes en temps réel, Apprentissage artificiel, Vision par ordinateur

Abstract

Collaborative robots are becoming more and more present in our everyday life. In particular, within the industrial environment, they emerge as one of the preferred solution to make assembly line in factories more flexible, cost-effective and to reduce the hardship of the operators' work. However, to enable a smooth and efficient collaboration, robots should be able to understand their environment and in particular the actions of the humans around them.

With this aim in mind, we decided to study technical gestures recognition. Specifically, we want the robot to be able to synchronize, adapt its speed and understand if something unexpected arises.

We considered two use-cases, one dealing with copresence, the other with collaboration. They are both inspired by existing task on automotive assembly lines.

First, for the co-presence use case, we evaluated the feasibility of technical gestures recognition using inertial sensors. We obtained a very good result (96% of correct recognition with one operator) which encouraged us to follow this idea.

On the collaborative use-case, we decided to focus on non-intrusive sensors to minimize the disturbance for the operators and we chose to use a depth-camera. We filmed the operators with a top view to prevent most of the potential occultations.

We introduce an algorithm that tracks the operator's hands by calculating the geodesic distances between the points of the upper body and the top of the head.

We also design and evaluate an approach based on discrete Hidden Markov Models (HMM) taking the hand positions as an input to recognize technical gestures. We propose a method to adapt our system to new operators and we embedded inertial sensors on tools to refine our results. We obtain the very good result of 90% of correct recognition in real time for 13 operators.

Finally, we formalize and detail a complete methodology to realize technical gestures recognition on assembly lines.

Keywords

Collaborative robotics, Human-Robot interaction, Real-time gesture recognition, Machine learning, Computer vision