Interface gestuelle pour la commande d'un capteur 3D tenu en main

Mémoire

Kento Ôtomo-Lauzon

Maîtrise en génie électrique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Interface gestuelle pour la commande d'un capteur 3D tenu en main

Mémoire

Kento Otomo-Lauzon

Sous la direction de:

Denis Laurendeau, directeur de recherche

Résumé

Ce mémoire porte sur la conception d'une interface utilisateur basée sur la reconnaissance de gestes pour la commande d'un capteur 3D tenu en main. L'interface proposée permet à l'opérateur d'un tel équipement de commander le logiciel à distance alors qu'il se déplace autour d'un objet à numériser sans devoir revenir auprès du poste de travail.

À cet effet, un prototype fonctionnel est conçu au moyen d'une caméra *Azure Kinect* pointée vers l'utilisateur. Un corpus de gestes de la main est défini et reconnu au moyen d'algorithmes d'apprentissage automatique, et des métaphores d'interactions sont proposées pour la transformation rigide 3D d'un objet virtuel à l'écran. Ces composantes sont implantées dans un prototype fonctionnel compatible avec le logiciel VXelements de Creaform.

Abstract

This thesis presents the development of a gesture-based user interface for the operation of handheld 3D scanning devices. This user interface allows the user to remotely engage with the software while walking around the target object.

To this end, we develop a prototype using an Azure Kinect sensor pointed at the user. We propose a set of hand gestures and a machine learning-based approach to classification for triggering momentary actions in the software. Additionally, we define interaction metaphors for applying 3D rigid transformations to a virtual object on screen. We implement these components into a proof-of-concept application compatible with Creaform VXelements.

Table des matières

Liste des tableaux

Liste des figures

Introduction

La numérisation 3D est le processus de construction d'une représentation virtuelle en trois dimensions d'un objet physique en maintenant un haut degré de fidélité. La haute précision et la vitesse des approches en la matière rend la technique applicable à plusieurs domaines d'applications, notamment dans les secteurs du divertissement, de la médecine, et industriel [1, 2]. Pour ce faire, plusieurs de ces appareils de captation 3D emploient des algorithmes de vision par ordinateur pour reconstruire la géométrie de l'objet observé. Par exemple, les capteurs des familles *MetraSCAN 3D* et *Go !SCAN 3D* de *Creaform* sont des capteurs tenus en mains employant une caméra et un projecteur de patrons laser disposés en paire stéréoscopique pour la saisie des points de données [3].

Creaform se spécialise dans la conception de capteurs 3D portatifs de haute précision et offre plusieurs appareils pouvant être tenus en main (figure 0.1). L'entreprise offre avec ses capteurs la plateforme logicielle *VXelements*, laquelle permet de configurer le capteur et d'orchestrer une séance d'acquisition de données 3D dans un contexte industriel (figure 0.2).

Dans le contexte d'une séance de numérisation, il n'est pas rare que l'opérateur ait à se déplacer autour de l'objet cible afin de le capter sous tous les angles de vue, ce qui est particulièrement marqué dans le cas d'objets volumineux. Ainsi, l'opérateur qui souhaite contrôler le logiciel *VXelements* lors de la séance doit se déplacer auprès du clavier et de la souris du poste de travail, ce qui peut résulter en une perte de productivité.

FIGURE 0.1 – Capteur 3D tenu en main de *Creaform*. Tiré de [4].

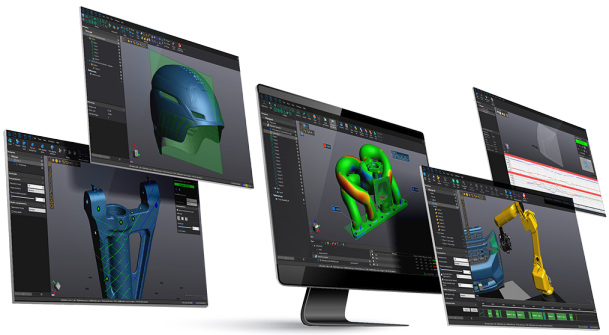


FIGURE 0.2 – Aperçu de la plateforme *VXelements*. Tiré de [5].

À cet effet, ce projet de maîtrise s'intéresse à l'utilisation d'un système d'interaction personne-machine basé sur la reconnaissance de gestes pour permettre à un utilisateur de contrôler le logiciel *VXelements* alors qu'il se déplace autour de l'objet à numériser. De tels systèmes ont effectivement été employés dans contextes où un haut degré de mobilité est requis de la part de l'utilisateur en utilisant une approche basée sur la vision artificielle. Par exemple, les capteurs *Kinect* de *Microsoft* ont démontré la pertinence d'interfaces personne-machine basées sur les gestes pour de tels contextes dans le domaine du jeux vidéo et pour la commande de systèmes informatiques de manière générale [6].

Ce mémoire décrit la conception d'un prototype fonctionnel d'interface personne-machine basée sur la reconnaissance de gestes pour le contrôle à distance du logiciel *VXelements* dans un contexte d'opération d'un capteur 3D tenu en main.

À ce sujet, nous étudierons le système développé selon ses composantes de haut-niveau décrites à la figure 0.3. Le module de capture et de suivi est responsable de l'interaction avec le capteur *Azure Kinect* sélectionné pour le projet, ainsi que d'effectuer le suivi de l'utilisateur par rapport à l'arrière-plan. Le chapitre 1 présente les algorithmes utilisés ainsi qu'une brève revue de littérature sur le sujet.

Les gestes étudiés dans le cadre de ce projet peuvent être séparés en deux catégories d'interactions, soient les interactions momentanées et isomorphes. Nous définissons les interactions momentanées comme des actions ayant un déclenchement instantané et discret sur la dimension temporelle, similairement à l'activation d'un interrupteur simple. Les chapitres 2 et 3 explorent deux approches gestuelles pour la modélisation d'interactions momentanées, soient la classification de gestes statiques et dynamiques.

Par contraste, nous définissons les interactions isomorphes comme ayant un déclenchement continu, dont l'amplitude de l'action peut être modulée par le mouvement de l'utilisateur. Les actions continues étudiées au chapitre 4 ont été choisie pour reproduire un mouvement de l'utilisateur en temps réel selon un rapport 1 : 1, d'où l'appellation *isomorphe*.

Enfin, le chapitre 5 présente l'application des métaphores d'interaction étudiées au logiciel de numérisation 3D, en expliquant les algorithmes utilisés pour le traitement en temps réel et la modélisation des flux de travail de l'utilisateur (module d'interaction), ainsi que les détails de l'interface avec le logiciel *VXelements*. Le chapitre comporte également une description détaillée des cas d'utilisation du logiciel par un utilisateur lors d'une séance de scan 3D.

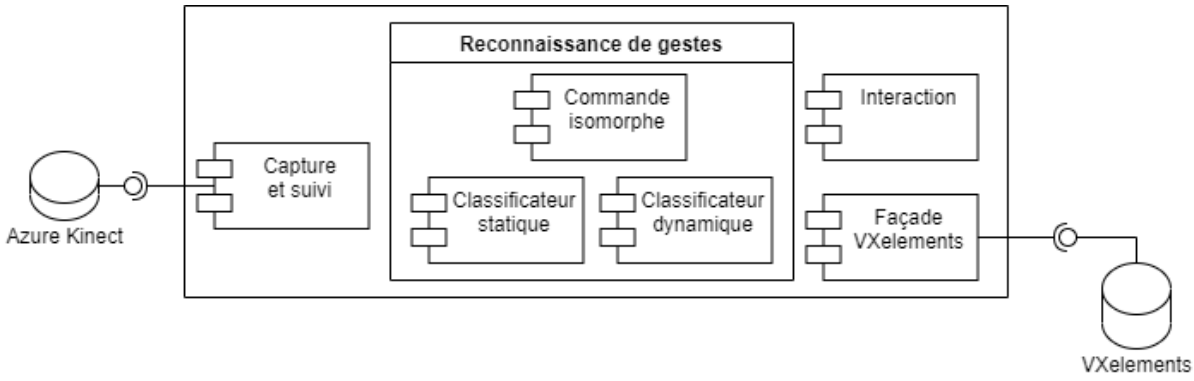


FIGURE 0.3 – Composantes de haut niveau du projet.

En guise de conclusion, nous discutons de la performance et de la pertinence des algorithmes et des métaphores d'interaction étudiées dans le cadre de ce projet, et proposons des améliorations au système qui pourraient faire l'objet de travaux futurs.

Chapitre 1

Détection et segmentation des mains

Ce chapitre détaille l'approche employée pour la localisation et la segmentation des mains de l'utilisateur du système par rapport à l'arrière-plan.

1.1 Littérature sur l'estimation de la pose humaine

Le problème de l'estimation de la pose humaine, ou la configuration d'un objet articulé de manière générale, est un problème bien connu dans le domaine de la vision artificielle [7]. Des approches classiques, telles que celles présentées par [8, 9], proposent de décrire l'objet reconnu comme une *Pictorial Structure*, laquelle qualifie la distance entre plusieurs éléments détectés individuellement. Des approches similaires basées sur la création manuelle de caractéristiques en utilisant des descripteurs tels que les histogrammes de gradients orientés (HOG) ont été employées pour la détection de parties du corps afin d'être combinées en une estimation de la pose complète du corps [10, 11, 12, 13]. Cependant, ces méthodes dépendent fortement de la qualité des détecteurs individuels des parties du corps et sont insuffisantes pour déterminer la position précise de ces parties [14].

Les méthodes basées sur l'apprentissage profond ont l'avantage d'offrir une représentation plus riche des parties du corps dans le cas des approches ascendantes, lesquelles combinent des descripteurs locaux en estimations globales, mais également la création de méthodes bout-en-bout qui considèrent le corps articulé dans son ensemble [14, 7]. Parmi celles-ci, [7] propose une cascade de réseaux de neurones profonds, laquelle permet un raffinement successif de la pose estimée, sans nécessiter la définition explicite d'un modèle d'interaction entre les articulations. D'autres méthodes telles que [15] intègrent une connaissance *a priori* du corps humain dans un contexte d'apprentissage non-supervisé ou semi-supervisé employant des réseaux antagonistes génératifs [16] (GAN). De telles méthodes d'apprentissage profond bout-en-bout ont également l'avantage de pouvoir être adaptées à la régression de la pose 3D sans modification majeure [14].

1.2 Approche expérimentale pour la segmentation des mains

L'approche utilisée dans le cadre de ce projet se base sur le *Kinect Body Tracking SDK* fourni avec le capteur *Azure Kinect*. Cette bibliothèque logicielle effectue l'estimation de la position et l'orientation d'un modèle squelettique humain à partir de captures RGB-D, lequel est illustré à la figure 1.1. Chaque point du modèle est associé à des coordonnées 3D en millimètres dans le repère de la caméra, lesquelles peuvent être projetées en coordonnées images au moyen de la calibration enregistrée par le capteur. La position 3D des mains est utilisée telle quelle dans les chapitres 3 et 4 pour la caractérisation du mouvement.

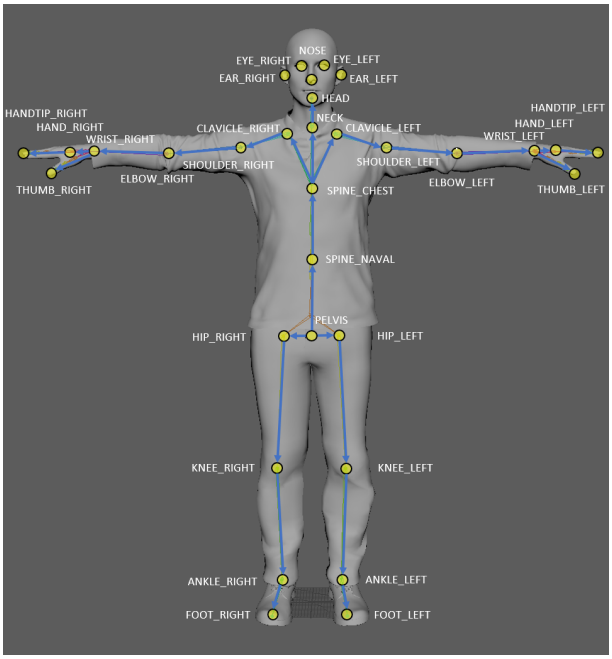
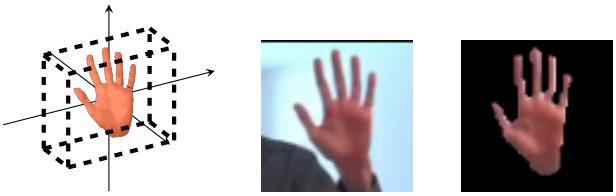


FIGURE 1.1 – Modèle squelettique estimé par le *Kinect Body Tracking SDK*. Tiré de [17].

Afin de caractériser l'apparence des mains, les coordonnées 3D détectées pour chacune d'elles sont utilisées comme point central d'une fenêtre de $25 \times 25 \times 10$ cm. Un rectangle englobant est ensuite choisi de manière à contenir tous les coins de ce volume. Finalement, l'image est rognée à ce rectangle et redimensionnée à une taille de 64×64 pixels. Une version de l'image est également créée en appliquant un seuillage sur la couleur et sur la carte de profondeur, ce qui permet de ne conserver que les pixels correspondant à la main. Le seuillage de couleur se fait dans l'espace CIELab, similairement à ce qui est proposé par [18], et le seuillage sur la profondeur conserve les pixels à la valeur de profondeur de ± 5 cm par rapport au centre de la main. La figure 1.2 illustre les étapes de ce processus. Les images résultantes sont utilisées au chapitre 2.



Volume englobant Image rognée Image segmentée

FIGURE 1.2 – Étapes du rognage de l'image des mains.

FIGURE 1.3 – Détection du squelette lors de l'opération d'un capteur tenu en main.

1.3 Validation de la résilience à l'occlusion

Étant donné que l'utilisation du logiciel VXelements nécessite la tenue en main d'un capteur 3D volumineux, une séquence d'images a été captée afin de valider la résilience du modèle du *Kinect Body Tracking SDK* à l'occlusion d'une main et d'une partie du corps. La figure 1.3 montre l'estimation de la pose humaine lors de l'opération d'un capteur tenu en main. Alors que la détection de la main tenant le capteur est inexacte à plusieurs occasions, on constate que cette occlusion ne semble pas empêcher le modèle de détecter le reste du corps de l'utilisateur incluant la main libre, qui est le point d'intérêt principal de ce projet.

Les images segmentées des mains sont utilisées aux chapitres 2 et 4 pour la classification de gestes et régression de la rotation, et la position 3D des mains est utilisée dans le chapitre 3 pour la classification de trajectoires.

Chapitre 2

Classification de symboles statiques de la main

Ce chapitre porte sur la reconnaissance de gestes statiques de la main basées sur les données d'apparence extraites au chapitre précédent. Pour ce faire, nous nous inspirons du concept des formes de la main dans les langues des signes pour définir un lexique de gestes à classifier au moyen d'algorithmes d'apprentissage machine. Nous comparons ensuite la performance de plusieurs algorithmes pour cette tâche de classification.

2.1 Littérature sur la reconnaissance de formes de la main

2.1.1 Formes de la main dans les langues des signes

Les formes de la main (*handshape*) sont l'une des composantes phonologiques des symboles constituant plusieurs langages des signes, dont la langue des signes américaine (ASL) [19]. Ces formes correspondent à la configuration de la main et des doigts et sont combinées à des composantes de mouvement, de position et d'orientation pour former les symboles de la langue. Parmi les symboles reconnus, on note les symboles de l'alphabet manuel tels qu'employés pour faire du *fingerspelling* (figure 2.1), où la majorité des symboles sont constitués d'une forme de la main sans composante de mouvement ou de position [19].

Bien que la reconnaissance d'une langue des signes soit en dehors de la portée de ce projet, nous emprunterons tout de même des concepts et des termes issus de cette littérature pour définir des symboles pour interagir avec le système développé dans notre projet.

2.1.2 Reconnaissance de gestes statiques de la main

Parmi les approches basées sur la vision pour la reconnaissance de gestes statiques, certaines se basent sur l'utilisation de marqueurs visuels tels que des gants colorés de manière à identifier chaque doigt [20, 21]. En raison des restrictions imposées par de telles approches, celles-

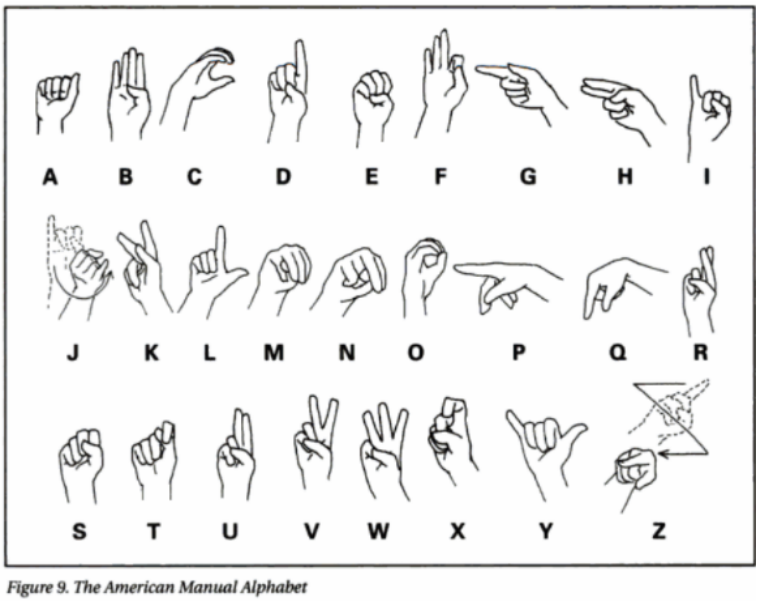


FIGURE 2.1 – Alphabet manuel de la langue des signes américaine. Tiré de [19].

ci sont mises de côté au profit d'approches pouvant reconnaître l'apparence naturelle des mains. Certaines approches visent la caractérisation précise de la posture de la main pour ensuite procéder à l'identification d'une forme de la main au moyen de modèles prédéfinis [20, 22, 23, 24], alors que d'autres n'utilisent que l'apparence depuis certains points de vue spécifiques afin de procéder à la classification directement [20, 25, 26, 27].

L'apprentissage automatique est employé dans plusieurs approches pour la classification des configurations de la main. Des approches basées sur une description de l'apparence en histogragrammes de gradients orientés (HOG) ont donné de bons résultats pour la classification de gestes dans la langue des signes [28, 29, 30, 31] et plus largement dans la reconnaissance de gestes de la main [32]. Des analyses en composantes principales (PCA) ont également été employées avec succès pour informer un classificateur de posture de main par [18], qui détermine que 35 composantes est un compromis suffisant pour qualifier l'essentiel de la variation des formes de mains étudiées. D'autres approches ont également utilisé des descripteurs SIFT [28, 33, 34, 35] ou des caractérisques pseudo-Haar [36, 37] comme représentation de la posture de la main. Ces représentations ont été employées conjointement avec des algorithmes tels que le SVM (*Support Vector Machine*) [38, 31, 20], le k plus proches voisins [38], ou le *Random Forest* [31] pour obtenir une classification finale.

Plusieurs méthodes basées sur des réseaux de neurones profonds à convolution (CNN) ont également été utilisées avec succès pour des tâches de classification et de segmentation d'images dans des contextes variés [39, 40, 41], pour permettre dans plusieurs cas l'entraînement d'un réseau bout-en-bout où les tâches de description de caractéristiques et de classification sont apprises simultanément. Plus spécifiquement, de tels réseaux ont été employés pour la recon-

naissance de gestes de la main à partir de données *RGB* ou monochrome [42]. De tels réseaux ont été employés par [43, 44] pour classifier individuellement les images d'une séquence vidéo afin de reconnaître des mots épelés par *fingerspelling*. La même approche a été exploitée pour la classification de gestes employant tout le corps ou pour la reconnaissance d'action [45, 46]. Cette manière d'utiliser les réseaux à convolution 2D a également l'avantage de permettre un apprentissage par transfert de larges ensembles de données tels qu'ImageNet [40] pour améliorer la performance en classification [43].

2.1.3 Utilisation de données synthétiques pour améliorer les réseaux de neurones profonds

Étant donné que les réseaux de neurones profonds profitent de vastes ensembles de données en entraînement, plusieurs auteurs proposent l'ajout de données synthétiques pour réduire l'ampleur de la tâche d'annotation manuelle [47]. Par exemple, [47, 48, 49, 50] utilisent des images de synthèse pour informer la détection d'objets spécifiques dans des scènes variées, et montrent une amélioration lorsque la quantité de données réelles est limitée [47]. Cette approche a été employée avec succès pour le suivi en 6 degrés de liberté par [51], et par [52] pour la détection de la posture de la main et de chacun des doigts.

2.2 Description des différents ensembles de données utilisés dans le système

Afin de pouvoir comparer différents classificateurs de manière uniforme, les mêmes ensembles de données sont utilisés pour l'entraînement et l'évaluation de chacun d'eux. Le tableau 2.1 recense les ensembles de données créés pour la classification des formes de la main. Chaque ensemble a été enregistré à l'aide du capteur *Azure Kinect*, dans un environnement identique avec le même utilisateur à une cadence de 5 images par seconde sous la forme d'une séquence vidéo en couleur avec canal de profondeur. Les images de main ont par la suite été créées en redimensionnant une région de 25×25 cm autour du centre de la main détectée par le *Kinect Body Tracking SDK* en une image de 64×64 pixels. Les images sont ensuite adaptées pour chacun des montages expérimentaux, par exemple en incluant ou en excluant certains canaux de couleur ou de profondeur.

La figure 2.2 montre les formes de la main étudiées dans les ensembles à 4 classes et à 10 classes. On note que l'ensemble de 10 classes contient les 4 classes de l'ensemble précédent en plus des six nouvelles classes illustrées sous `entraînement10`.

Titre	Description	Nombre d'images
`entraînement4`	Ensemble d'entraînement de base à 4 symboles. Capté entre 1.5m et 2.5m de la caméra.	988
`entraînement10`	Ensemble d'entraînement de base à 10 symboles. Capté entre 1.5m et 2.5m de la caméra.	2 196
`validation10`	Exemples de validation des 10 symboles pris en conditions idéales. Capté entre 2m et 2.5m de la caméra.	2 892
`validation10-rot`	Exemples de validation des 10 symboles comportant une variabilité plus importante dans la qualité des gestes (orientation, alignement des doigts). Capté entre 2m et 2.5m de la caméra.	1 276
`synthétique10`	Données de rendu synthétiques générées au moyen du logiciel *Blender*. Voir 2.5.1.	10 000

TABLE 2.1 – Description des ensembles de données utilisés.

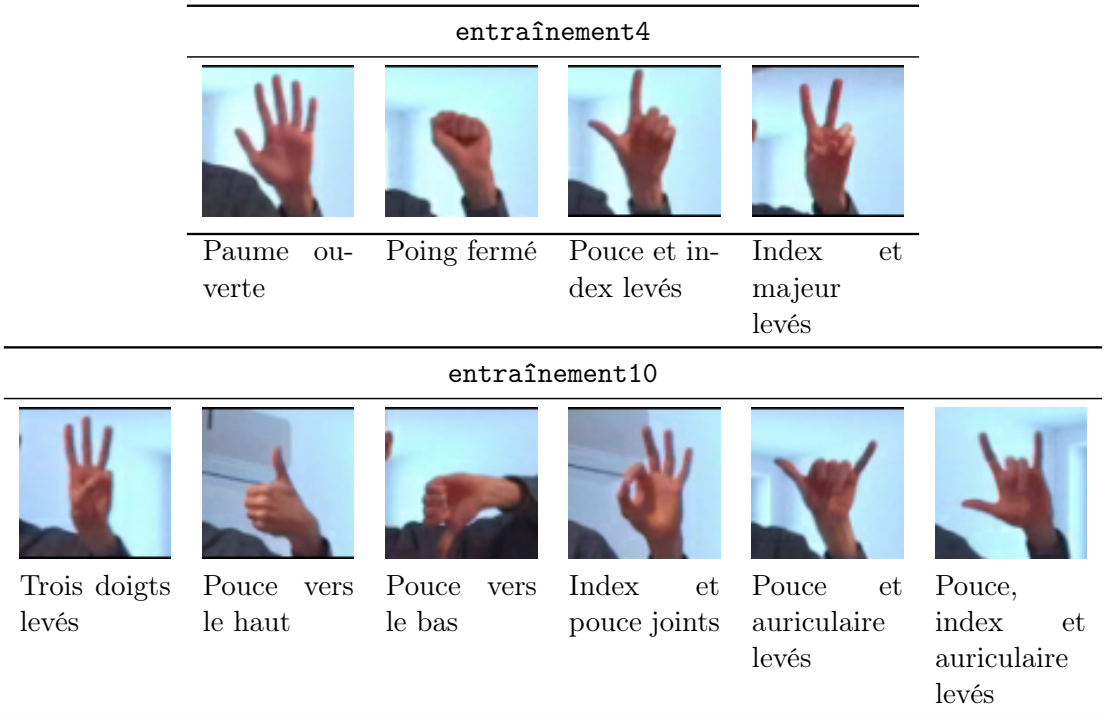


FIGURE 2.2 – Formes de la main comprises dans les ensembles `entraînement4` et `entraînement10`.

2.3 Classification par décomposition en composantes principales

Une première version du classificateur de gestes statiques est élaborée à partir de la décomposition en composantes principales (PCA) des images segmentées de la main. Le tableau 2.2

Classificateur	Gestes	Nombre composantes	validation10
SVC	4	35	77.4
k-NN	4	35	74.4
SVC	10	35	71.2
k-NN	10	35	70.0
SVC	10	10	63.0
k-NN	10	10	59.8
SVC	10	100	70.4
k-NN	10	100	69.4
SVC (RGB)	10	35	72.5
k-NN (RGB)	10	35	69.9

TABLE 2.2 – Exactitude (%) du module de classification basé sur la décomposition en composantes principales.

présente l'exactitude des variantes du classificateur étudié.

Sur l'ensemble `entraînement4`, la décomposition en 35 composantes [18] parvient à expliquer 84% de la variance pour les quatre configurations de la main incluses. Cela se traduit en une exactitude en test de 77% et 74% respectivement pour les classificateurs SVC et k-NN sur l'ensemble `validation10` limité aux quatre gestes présents dans l'ensemble d'entraînement.

La performance diminue légèrement lorsque le nombre de gestes étudiés passe à 10, alors que la décomposition en 35 composantes ne parvient plus qu'à expliquer 67% de la variance des données, ce qui se traduit en une exactitude de 71% et 70% pour les classificateurs SVC et k-NN. Dans ce contexte, faire varier le nombre de composantes principales ne contribue pas à améliorer la performance du classificateur, car une décomposition en 100 composantes principales offre la même exactitude en test que la variante en 35 composantes. Diminuer le nombre de composantes se traduit également en une baisse de la performance du classificateur, soit une perte de près de 10% lorsque l'on passe à 10 composantes.

De manière générale, on observe que le classificateur SVC offre une performance égale ou supérieure au classificateur de la méthode k plus proches voisins avec un voisinage de trois points. On note également que l'ajout des canaux de couleurs ne se traduit pas en une amélioration notable de la performance pour justifier le coût supplémentaire en traitement.

2.4 Classification par histogrammes de gradients orientés

Une seconde version du classificateur de formes de la main est élaborée à partir des histogrammes de gradients orientés similairement aux travaux de [32, 31]. L'extraction des descripteurs HOG est implantée d'après [53] en adaptant les coefficients pour accommoder des

Variante	**Classificateur**	`validation10`	`validation10-rot`
Base (SVC)	SVC	89.6	82.9
Base (k-NN)	k-NN	86.0	78.5
RGB (SVC)	SVC	90.1	83.6
Rotation (SVC)	SVC	89.6	82.5

TABLE 2.3 – Exactitude (%) du module de classification basé sur les histogrammes de gradient orientés.

images de taille 64×64 pixels, ce qui permet d'exprimer le descripteur sous la forme d'un vecteur de 324 composantes pour les images monochromes, lequel est utilisé en entrée au classificateur. Les images en entrée subissent une étape de pré-traitement afin de séparer la main de l'arrière-plan en utilisant les données de profondeur, comme dans le cas du classificateur PCA.

Le tableau 2.3 présente l'exactitude en test sur l'ensemble des classificateurs entraînés sur les dix gestes du corpus. Chaque classificateur a été entraîné sur l'ensemble `entraînement10` à l'exception de la variante *Rotation*.

Comme dans le cas du classificateur PCA, on constate que le classificateur SVC offre une meilleure performance à données égales comparé au classificateur des k plus proches voisins avec un voisinage de 3, se traduisant en une différence de près de 4% pour les deux ensembles de validation étudiés. On conclut que ce classificateur parvient à mieux généraliser pour des points de vue légèrement différents des données d'entraînement que le classificateur k-NN, ce qui est attendu de manière générale avec des algorithmes de cette famille.

L'ajout de canaux de couleurs RGB n'amène pas d'amélioration appréciable de la performance du classificateur, ce qui est cohérent avec les observations de [31].

De plus, l'ajout d'un ensemble de données spécialement conçu pour ajouter de la rotation dans les points de vue lors de l'entraînement ne s'est pas traduit en une amélioration notable de la performance en test sur l'ensemble `validation10-rot`, indiqué dans les résultats par la variante *Rotation*. Il est donc possible que le classificateur de base arrive suffisamment bien à généraliser dans la plage de rotation possibles pour un geste en conditions normales. Alors que [31] effectue un alignement de chaque image selon l'axe principal de la main segmentée pour pallier la faiblesse des descripteurs HOG en rotation, cette approche n'a pas été mise en place pour ces classificateurs-ci, puisque certains gestes du corpus ne sont différenciés que par leur orientation (ex. pouce vers le haut et pouce vers le bas). Il est également attendu que l'opérateur qui emploie le système pour commander un logiciel gardera une orientation constante lors de l'exécution de ses gestes, ce qui n'est pas une caractéristique des langues des signes.

2.5 Classification par réseaux à convolution

Une version du classificateur de gestes statiques a également été élaborée à l'aide de réseaux de neurones profonds. Pour cette section, des architectures de réseaux de la famille ResNet [54] et MobileNet [41] ont été étudiées respectivement pour leur performance établie en classification sur de vastes ensembles tels que ImageNet [54], leur applicabilité à des contextes de traitement en temps réel [41] et leur présence suffisamment courante dans la littérature pour qu'il existe des implantations de référence dans des librairies telles que *PyTorch* [55]. Bien que ces architectures réseau ne soient plus considérées comme l'état de l'art pour des tâches de classification de référence comme *ImageNet* [56], nous avançons que cela n'est pas un problème pour un ensemble de données de taille limitée, tel que le corpus de 10 gestes étudié dans le cadre de ce projet.

L'implantation des modèles étudiés dans cette section a été faite un moyen de la librairie *PyTorch* en modifiant les implantations de référence des réseaux pour adapter la couche d'entrée au nombre de pixels des images (64×64 pixels) et au nombre de canaux de couleurs utilisés (1 à 4), ainsi que pour faire correspondre le nombre de sorties de la dernière couche au nombre de classes présentes dans l'ensemble de données (10 classes). Comme pour l'entraînement des modèles basés sur les histogrammes de gradients orientés, l'étude des ensembles `entraînement4` et `validation4` est omise puisqu'elle est considérée triviale, étant donnée la performance sur les ensembles à 10 gestes. Chaque modèle a été entraîné sur 20 périodes d'entraînement, avec un taux d'apprentissage de 0.001, sur 80% des données de l'ensemble d'entraînement. Afin d'éviter le sur-entraînement, la meilleure version du réseau pour chaque expérimentation est par la suite choisie de sorte à conserver les poids du réseau à l'époque d'entraînement ayant obtenu la meilleure performance sur les 20% restants.

Réseau	**Mode**	**Entraînement**	**Validation**	`validation10`
ResNet18	Gris	99.9	98.4	69.7
ResNet18	Gris-D	99.8	98.6	85.7
ResNet18	RGB-D	100	99.1	70.9
ResNet18	D	99.8	94.1	22.9
MobileNet2	Gris	99.6	90.7	46.6
MobileNet2	Gris-D	99.6	92.7	75.5
MobileNet2	RGB-D	99.9	94.3	79.8
MobileNet2	D	99.9	92.7	18.3
ResNet50	Gris	99.9	98.4	5.9
ResNet50	Gris-D	99.5	98.2	89.3
ResNet50	RGB-D	100	97.9	86.2

TABLE 2.4 – Comparaison des architectures de l'exactitude (%) des réseaux de neurones pour la classification de formes de la main.

Le tableau 2.4 présente la performance en classification des variantes entraînées sur l'ensemble `entraînement10`. Parmi les architectures de réseau étudiées, les réseaux de la famille ResNet offrent une meilleure performance sur les ensembles de validation et de test que le réseau MobileNet2, malgré que l'on atteigne une exactitude quasi-parfaite dans les deux cas sur l'ensemble d'entraînement. On en déduit que l'architecture ResNet permet une meilleure généralisation sur les données du problème.

En ce qui a trait à la sélection des données en entrée, la meilleure performance est atteinte en combinant les données de profondeur aux données du spectre visible (RGB ou Gris), avec la variante Gris-D offrant la meilleure performance pour les réseaux ResNet18 et ResNet50, et la variante RGB-D pour le réseau MobileNet2. Il apparaît, en outre, que le canal de profondeur seul ne permette pas une classification adéquate, n'atteignant que des niveaux d'exactitude de l'ordre de 20% pour les deux variantes testées. Ce résultat est attendu étant donné la basse résolution des cartes de profondeurs générées par le capteur Kinect en comparaison aux images RGB [17].

Il est à noter que les images utilisées dans cette section n'ont pas subit d'étape de prétraitement pour isoler l'avant-plan, contrairement aux sections précédentes. Une comparaison rigoureuse entre les approches nécessiterait d'étudier l'effet de cette opération de segmentation.

2.5.1 Augmentation des ensembles de données à l'aide d'images synthétiques

Suivant les travaux de [52], l'ensemble de données `synthetique10` a été construit. En utilisant le modèle de main articulé réaliste présenté par les auteurs de [52], une posture de référence a été sauvegardée manuellement pour les 10 gestes du corpus étudié. À l'aide de l'API de scriptage du logiciel de rendu 3D *Blender*, 1000 images ont été synthétisées pour chaque geste en faisant varier la position de la caméra et de la source de lumière dans la scène, l'intensité de la source lumineuse, ainsi qu'en ajoutant une faible variation sur chaque angle de rotation 3D de chaque jointure de la main par rapport à la posture de référence. Le tableau 2.5 présente les plages de paramètres utilisées pour chacun d'eux. Le système de coordonnées utilisé dans la scène 3D est illustré à la figure 2.3. Pour chaque prise de vue, on conserve l'image RGB visible, une carte de profondeur où chaque pixel représente la distance à la caméra en millimètres, un masque de segmentation de l'avant-plan pour permettre le remplacement facile de l'arrière-plan (figure 2.4) et les matrices intrinsèque et extrinsèque de la caméra à titre de référence.

Procédure d'entraînement

Pour chacune des variantes expérimentales étudiées, l'entraînement du réseau est fait sur 5 périodes, avec un taux d'apprentissage de 0.001. Pour les variantes utilisant des données synthétiques, l'ensemble d'entraînement est constitué à 80% des images de l'ensemble `synthétique10` et à 80% d'`entraînement10`. L'ensemble de validation comprend les 20% restants.

Paramètre	**Plage de valeurs**
Distance Caméra (m)	$[0.25, 2[$
Rotation axe projecteur caméra XYZ (°)	$[0,0]; [-30, 30[; [-30, 30[$
Intensité source (W)	$[750, 1250[$
Position source lumineuse XYZ (m)	$[3,4[; [-5, 5[; [-4, 4[$
Variation angle jointure (°)	2

TABLE 2.5 – Plages permises pour les valeurs aléatoires assignées à chacun des paramètres de rendu.

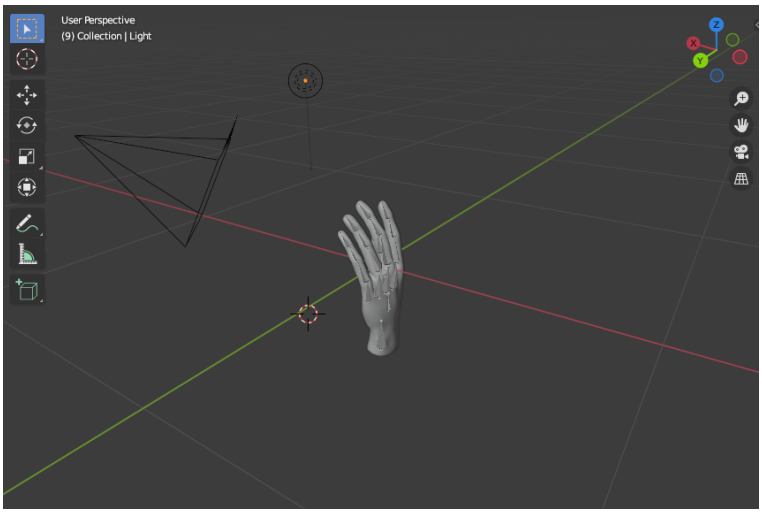


FIGURE 2.3 – Système de coordonnées de la scène Blender. La paume de la main est face à l'axe X positif, et les doigts pointent selon l'axe Z dans le sens positif.

Rendu RGB | Carte de profondeur (uint16) | Masque de remplacement de l'arrière-plan | Image finale

FIGURE 2.4 – Étapes de rendu d'images synthétiques.

Ainsi, les valeurs d'exactitude en entraînement et en validation de ces variantes sont évaluées à la fois sur des données synthétiques et des données réelles. On applique également une étape de pré-traitement à chaque image synthétique donnée en entrée, lors de laquelle on remplace l'arrière-plan par une image de l'ensemble *INRIA Holidays* [57], et on applique du bruitage sur le canal de profondeur, lorsque ce dernier est requis, pour simuler les imperfections présentes sur les captures réelles.

Résultats expérimentaux

Le tableau 2.6 présente la performance des réseaux entraînés sur `entraînement10` et `synthétique10`. À l'instar des réseaux étudiés à la section précédente, les variantes n'employant pas l'information de profondeur peinent à généraliser vers les ensembles de test et de validation malgré un haut pourcentage d'exactitude en entraînement, ce qui est particulièrement visible pour la variante ResNet18 RGB qui ne parvient qu'à une exactitude de 25.5% sur l'ensemble `validation10`, ce qui est nettement en deçà de celle des réseaux n'ayant été entraînés que sur des données réelles. Les variantes employant les cartes de profondeur sont celles qui atteignent la meilleure performance en test, avec des valeurs d'exactitudes allant de 91.1% à 93.7%.

Réseau	**Mode**	**Entraînement**	**Validation**	`validation10`
ResNet18	Gris	97.6	95.9	78.8
ResNet18	Gris-D	99.6	99.2	91.8
ResNet18	RGB	70.1	53.6	25.5
ResNet18	RGB-D	99.5	89.6	93.7
ResNet50	RGB-D	99.5	98.7	91.1

TABLE 2.6 – Comparaison de l'exactitude (%) des réseaux de neurones entraînés sur `entraînement10` et `synthétique10`.

Le tableau 2.7 compare directement l'effet de l'ajout des données synthétiques lors de l'entraînement. On constate que l'ajout des données synthétiques augmente l'exactitude pour toutes les variantes étudiées, avec l'écart le plus marqué pour les variantes *Gris* et $RGB - D$. En comparant les réseaux ResNet18 et ResNet50, on n'observe pas de gain appréciable de la performance en classification pour le problème étudié. On peut conclure que le problème n'est pas d'une complexité suffisante pour justifier le coût supplémentaire en temps de traitement de ResNet50 par rapport à ResNet18. De plus, il semble également que la variété des points de vue présente dans les images synthétiques a pour effet d'améliorer sensiblement la qualité de la prédiction pour l'ensemble `validation10-rot`, lequel comporte une plus grande variabilité dans l'orientation de la main ainsi que dans le positionnement des doigts. On passe de 52.2% à 79.9% en exactitude pour la variante *Gris* et de 77.4% à 86.2% et pour la variante $RGB - D$, faisant de celle-ci la meilleure de toutes les variantes de classificateur essayées.

Variante	Mode	validation10	validation10-rot
Réel ResNet18	Gris	46.6	52.2
Synthétique ResNet18	Gris	78.8	79.9
Réel ResNet18	Gris-D	89.3	83.9
Synthétique ResNet18	Gris-D	91.8	83.5
Réel ResNet18	RGB-D	70.9	77.4
Synthétique ResNet18	RGB-D	93.7	86.2
Réel ResNet50	RGB-D	86.2	80.0
Synthétique ResNet50	RGB-D	91.1	81.9

TABLE 2.7 – Effet de l'utilisation des données synthétiques lors de l'entraînement sur l'exactitude (%).

Améliorations possibles

À des fins de simplicité, les ensembles d'entraînement n'ont été formés que par une simple concaténation des ensembles d'images réelles et synthétiques, malgré que ceux-ci n'aient pas la même taille. Il serait pertinent de comparer les résultats obtenus par cette approche naïve à ceux que l'on aurait pu obtenir soit en dupliquant les données réelles jusqu'à obtenir proportion égale d'image réelles et synthétiques, soit en exploitant l'apprentissage par transfert en faisant d'abord une phase d'entraînement sur les données synthétiques, puis une seconde phase d'entraînement sur les données réelles.

2.6 Conclusion

Parmi tous les classificateurs présentés dans ce chapitre, le modèle ResNet18 entraîné sur les ensembles `entraînement10` et `synthétique10`, et recevant en entrée les canaux RGB-D est celui qui offre la meilleure qualité de prédiction pour la classification des gestes statiques avec une exactitude de 93.7% sur l'ensemble `validation10`.

Il a cependant été démontré qu'une performance légèrement plus faible pouvait être obtenue uniquement à partir d'images en couleurs ou en teintes de gris. Par exemple, le classificateur en histogramme de gradients orientés atteint une exactitude de 89.6% sur le même ensemble. Alors que dans l'état actuel, les manipulations nécessitent l'utilisation des données de profondeur pour la segmentation de la main par rapport à l'arrière-plan, [31, 18] montrent qu'une telle segmentation est également possible par seuillage basé sur la couleur. Ainsi, on peut considérer que ce niveau de performance pourrait être atteint sans l'utilisation de quelque information de profondeur que ce soit, ce qui rend l'approche optimale pour des contextes où on ne bénéficierait pas de cartes de profondeur telles que celles fournies par le capture *Azure Kinect*.

Pour les fins du prototype fonctionnel, le classificateur statique utilisé est une variante ResNet18 en mode RGB-D, puisque l'utilisation du capteur *Azure Kinect* est une prémisse dans le projet.

Chapitre 3

Reconnaissance de gestes dynamiques

Ce chapitre couvre la reconnaissance de gestes dynamiques de la main, en utilisant les données de position extraites au chapitre 1. Nous proposons un corpus de gestes exprimés sous la forme de trajectoires dans l'espace 3D et présentons l'approche expérimentale utilisée pour leur classification ainsi que la performance du classificateur en prédiction.

3.1 Littérature sur la reconnaissance de gestes dynamiques

La reconnaissance de gestes dynamiques ajoute la dimension temporelle à la tâche de classification et plusieurs stratégies sont étudiées dans la littérature pour l'incorporer dans des modèles [43].

À l'instar des méthodes de classification de gestes statiques, certaines approches se basent sur la détection et la localisation de certaines partie du corps [20], en utilisant des méthodes telles que le seuillage par couleur ou arêtes, ou la soustraction de l'arrière-plan entre les images [58, 59]. À l'aide de caméras de profondeur telles que celles de la famille *Kinect*, certaines approches considèrent le mouvement comme un patron rigide dans l'espace 3D [60]. Dans le domaine de la reconnaissance des langages des signes, des capteurs *Kinect* ont été employés avec succès pour la reconnaissance de phrases employant un vocabulaire simplifié [31, 61] en utilisant des modèles de Markov cachés (HMM) [62, 63].

Étant donné que les gestes dynamiques s'exécutent sur plusieurs images d'une séquence vidéo, une segmentation sur la dimension temporelle est nécessaire. D'une part, si l'on suppose que les séquences vidéo étudiées correspondent directement à l'exécution d'un seul geste, le problème est réduit à un problème de classification simple sans composante temporelle [20]. Par exemple, une heuristique externe telle que des changements abrupts dans la vitesse de déplacement [64, 65] ou le déplacement par rapport à une position de repos ou la sortie de la main du champ de vision de la caméra [66] pourrait être utilisée pour inférer les instants de début et de fin d'un geste dans certaines application contrôlées. D'autre part, on peut inclure la

tâche de segmentation dans le problème de classification, par exemple en utilisant des modèles adaptés aux séries temporelles tels que le *Dynamic Time Warping (DTW)* [67, 68], les modèles de Markov cachés (HMM) [69, 58, 70], ou en appliquant continuellement un algorithme de classification sur une fenêtre coulissante de taille fixe sur les dernières données observées [20, 71, 72, 67]. Dans ce second cas, on considère qu'un geste est formé lorsqu'un classificateur reconnaît un geste dans la fenêtre coulissante avec un haut taux de confiance [20].

La métrique de *Dynamic Time Warping* exprime une distance entre deux signaux. Contrairement à la distance euclidienne simple, elle n'est pas sensible à la distortion sur la dimension temporelle, ce qui la rend utile à la classification de séries temporelles [73]. Plusieurs approches de normalisation de la métrique de distances sont employées, telles que la normalisation sur la longueur de la fenêtre de signal optimal ou celle de l'un ou l'autre des signaux comparés [73]. Cette métrique de distance a été employée avec succès avec des algorithmes de classification classiques tels que le k plus proches voisins [67, 68].

Des approches basées sur l'apprentissage profond ont également été appliquées avec succès au problème de la reconnaissance de gestes et la reconnaissance d'activité [31]. Certaines méthodes appliquent des réseaux à convolution 2D individuellement sur chaque image de la séquence pour extraire l'information visuelle en vue d'appliquer l'un des algorithmes précédents [43, 74, 44, 75]. D'autres méthodes tentent d'incorporer la composante temporelle en qualifiant un déplacement. Par exemple, [76] ajoute à une architecture de réseau convolutionnel des informations de déplacement 3D telles que la vitesse. De manière similaire, des réseaux à convolutions 3D ont été appliqués sur des images volumétriques pour informer le réseau de la composante de déplacement 3D [43, 77, 78, 79].

Des architectures spécialisées de réseaux de neurones dans le traitement de données temporelles telles que les réseaux récurrents (RNN) et les LSTM (*Long Short-Term Memory*) ont également été employées à cause de leur succès dans d'autres domaines pour modéliser les dépendances temporelles dans de vastes ensembles de données [43]. Par exemple, [80, 81, 82] proposent des architectures de réseaux profonds comportant à la fois des composantes des CNN et des RNN ou LSTM pour l'apprentissage de gestes de durée variable. Sans employer directement les données visuelles, [83] applique un réseau RNN aux données de posture fournies par un capteur *Leap Motion* pour reconnaître l'alphabet manuel de l'ASL.

3.2 Description des ensembles de données

La figure 3.1 montre l'ensemble de gestes étudié. L'ensemble comprend quatre translations (balayage vers le haut, bas, gauche, droite), des trajectoires circulaires en sens horaire et anti-horaire dans un plan parallèle au plan image de la caméra et sur le plan parallèle au sol, ainsi que des trajectoires 3D pour toucher l'épaule opposée et imiter le geste de placer un objet derrière son épaule. On nommera le lexique de gestes comportant les quatre translations et les

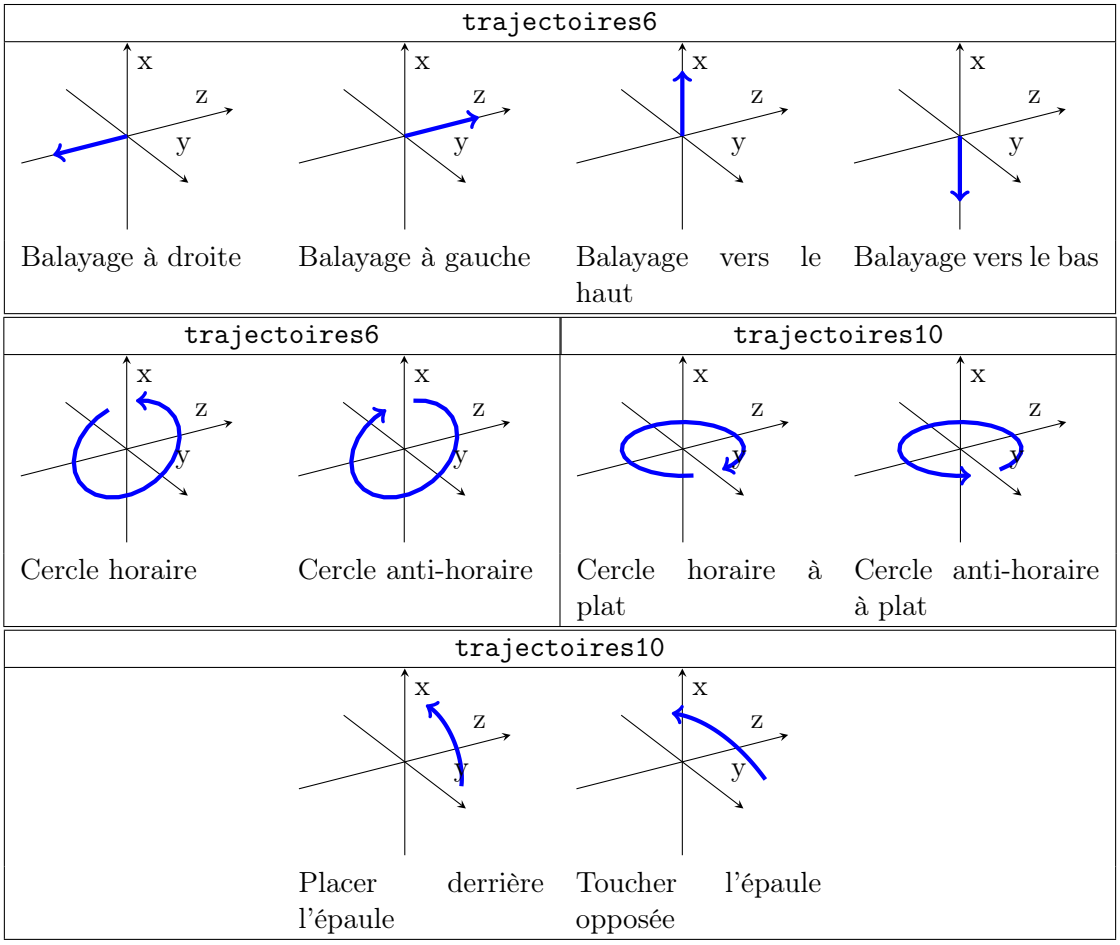


FIGURE 3.1 – Représentation stylisée des gestes dynamiques des ensembles `trajectoires6` et `trajectoires10` dans le repère de l'utilisateur. L'origine est la position de l'utilisateur et la caméra est placée sur l'axe y positif et pointe vers l'origine.

cercles verticaux `trajectoires6`, et `trajectoires10` celui contenant l'ensemble des gestes.

Les jeux de données utilisés pour l'entraînement prennent la forme d'une séquence vidéo captée à au moyen du capteur *Azure Kinect* à un rythme de 5 images par seconde, en mode RGB-D en résolution 1280×720. Un fichier d'annotation est adjoint à chaque séquence et contient l'identifiant de chaque geste, en plus du numéro des images marquant le début et la fin du geste. Les ensembles `entraînement6` et `entraînement10` sont les séquences d'entraînement utilisées respectivement pour l'apprentissage des lexiques `trajectoires6` et `trajectoires10`, et l'ensemble `valitation10` est l'ensemble pour l'évaluation des modèles.

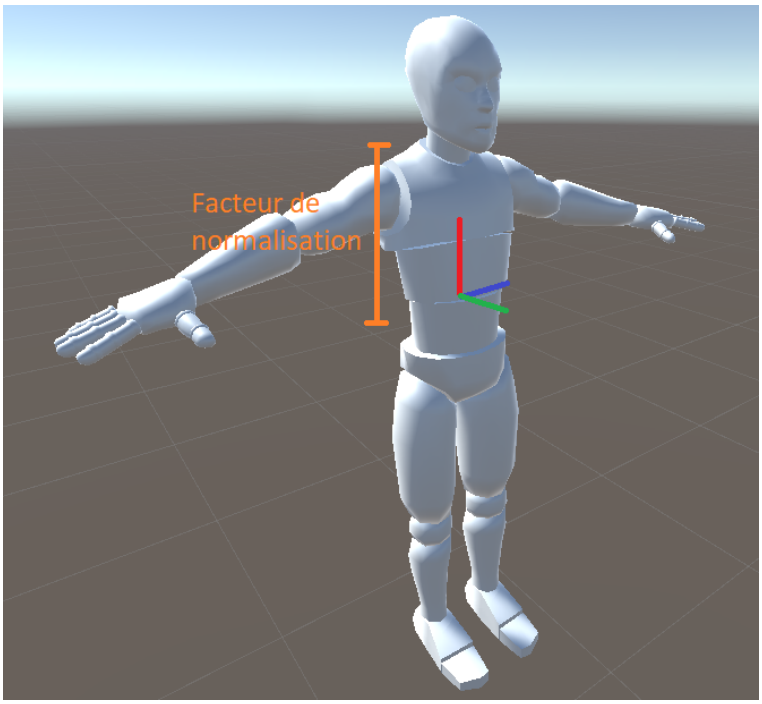


FIGURE 3.2 – Système de référence local au corps. Les axes XYZ sont représentés en rouge, vert, bleu respectivement. Le facteur de normalisation est illustré en orange. Modèle 3D tiré de [85].

3.3 Approche expérimentale pour la reconnaissance de gestes dynamiques

Le module de classification des gestes dynamiques est créé en appliquant un algorithme des k plus proches voisins en utilisant le *Dynamic Time Warping* comme métrique de distance. La position dans l'espace 3D de la main fournie par le *Kinect Body Tracking SDK* est convertie dans un repère égocentrique (figure 3.2) puis accumulé sur 10 images au moyens d'une fenêtre coulissante. Une normalisation des coordonnées est également effectuée d'après les proportions du corps de l'utilisateur, en utilisant la distance entre le cou et le bas de la colonne vertébrale comme dénominateur, d'après les travaux de [84] sur le capteur *Kinect* de première génération.

Une approche basée sur la classification [20] est employée pour résoudre le problème de la segmentation continue des gestes. On considère qu'un geste est reconnu lorsque la métrique DTW par rapport à l'une des trajectoires en exemple est inférieure à un seuil fixe normalisé de 0.1, lequel a été déterminé empiriquement. Cette approche correspond à un algorithme d'un seul plus proche voisin (1NN) tel qu'exprimé par l'équation 3.1, lequel offre un niveau de performance de base adéquat pour plusieurs applications [86].

Variante	Lexique	Complet	Balayages exclus
Base	`trajectoires6`	40	83
Axes normalisés	`trajectoires6`	53	94
Base	`trajectoires10`	27	27
Axes normalisés	`trajectoires10`	46	64

TABLE 3.1 – Exactitude (%) du classificateur 1NN pour la reconnaissance de gestes dynamiques.

$$\begin{cases} c & \text{si } \exists\, c \,:\, \frac{DTW(t_{exemple}, t_{évalué})}{n} < 0{,}1 \\ \emptyset & \text{sinon} \end{cases} \tag{3.1}$$

où c est la classe reconnue par le classificateur, $t_{exemple}$ est le vecteur de positions 3D correspondant à une trajectoire de l'ensemble d'entraînement, $t_{évalué}$ celui correspondant à la fenêtre coulissante de longueur n, et n est la longueur de $t_{exemple}$.

3.3.1 Discussion des résultats

Le tableau 3.1 montre la performance en prédiction des variantes de classificateur étudiées. L'exactitude en prédiction de chacun des modèles est mesurée en évaluant chacun des modèles sur les séquences pré-segmentées de l'ensemble `validation10`. Il est donc à noter que la mesure d'exactitude ne permet pas d'évaluer la qualité de la segmentation des gestes, mais sert plutôt à comparer les variantes des modèles entre elles.

Pour toutes les variantes testées, on note que la qualité de la prédiction des quatre trajectoires de balayage (figure 3.1) est nettement plus basse que pour les autres gestes. Dans le cas de base sur le lexique à 6 gestes, l'exactitude passe de 40% à 83% en ignorant ces gestes et le même constat peut être énoncé pour les autres variantes, à l'exception du cas de base sur le lexique de 10 gestes. Étant donné que ces gestes ont une durée plus courte que les autres (3 à 5 images en moyenne comparé à 8 à 10 pour le reste), une tolérance plus petite est permise dans l'exécution du geste, ce qui a le défaut de rendre la reconnaissance de ces gestes impraticable dans la réalité. Il serait pertinent de comparer la présente approche avec une qui ne normaliserait pas la métrique DTW sur la longueur de la trajectoire.

Il semble que l'ajout des gestes de l'ensemble `trajectoires10` ait un effet néfaste sur l'exactitude générale de la prédiction, faisant chuter l'exactitude à 64% et 27% les variantes évaluées sur ce lexique, même après avoir exclu les trajectoires de balayage.

L'une des améliorations apportées aux modèles de base consiste à appliquer une normalisation des axes sur certaines des trajectoires. En effet, pour les gestes étant limités à un plan de l'espace 3D, tels que les gestes de balayage et les trajectoires circulaires, une normalisation sur la moyenne est effectuée uniquement pour l'axe normal au plan sous-tendu par la trajectoire,

ce qui a pour effet de permettre une plus grande variabilité dans les gestes reconnus (figure 3.3). Dans le cas des modèles reconnaissant l'ensemble `trajectoires6`, cette amélioration est liée à une augmentation de l'ordre de 10% de l'exactitude, et cette augmentation est encore plus marquée sur l'ensemble `trajectoires10`.

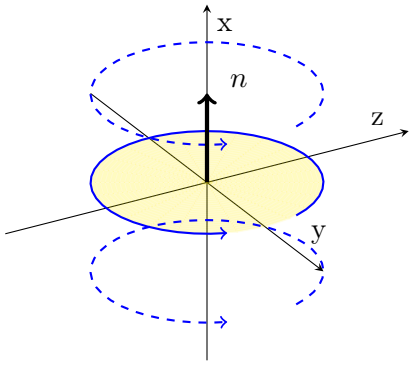


FIGURE 3.3 – Effet de la normalisation sur la reconnaissance de trajectoires. Des trajectoires parallèles à l'exemple sont ajoutées à l'ensemble reconnu.

3.3.2 Conclusion

L'approche étudiée dans cette section pour la classification de gestes dynamiques comporte plusieurs lacunes importantes limitant sont applicabilité. D'une part, les seuls gestes pour lesquels on observe une haute exactitude sont les gestes plus longs et plus complexes, avec un temps d'exécution entre 1 et 2 secondes. Cela a pour effet d'ajouter une charge mentale supplémentaire à l'utilisateur et peut même aller jusqu'à ralentir le processus de travail d'un utilisateur expert. D'autre part, la faible tolérance des gestes ainsi que la présence de plusieurs faux-positifs lors de séances de test en direct, accentuent encore davantage ce problème. Il serait pertinent d'explorer davantage l'utilisation d'autres approches telles que celles employant des réseaux de neurones profonds RNN ou LSTM, ou même des HMM, lesquelles auraient l'avantage de permettre une plus grande flexibilité par rapport à la composante temporelle [43].

De plus, l'implantation actuelle se base sur un algorithme de type k plus proches voisins, ce qui entraîne une charge indue en temps de calcul, laquelle est difficilement applicable à de vastes volumes de données, soit pour accommoder un large lexique de gestes ou plusieurs variantes acceptées d'un même geste.

Malgré ces inconvénients, l'approche a comme avantage de permettre un apprentissage de type *one-shot* ou *few-shot*, ce qui ouvre la possibilité de concevoir un système où le corpus de gestes pourrait être personnalisé à la volée par l'utilisateur. Ce genre d'interaction n'est pas étudiée dans le cadre de ce projet, mais pourrait se révéler très pertinent dans le domaine de l'interaction personne-machine en général.

Chapitre 4

Interactions isomorphes

En plus d'actions momentanées, un logiciel d'édition 3D tel que VXelements comporte des actions pour lesquelles l'utilisateur peut souhaiter moduler l'amplitude. Le déplacement du curseur de souris d'un ordinateur est un exemple de ce genre d'interaction isomorphe, où l'amplitude du mouvement du poignet est directement corrélée à l'amplitude du mouvement du curseur à l'écran. Dans ce chapitre, nous étudierons deux types d'interaction isomorphe dans le contexte d'un logiciel d'édition 3D, soient la translation et la rotation d'un objet virtuel à l'écran.

4.1 Métaphore de translation

La commande de translation est calculée comme le déplacement de chacune des mains dans l'espace de référence de la caméra. Pour ce faire, la position XYZ de chacune des mains est estimée pour chaque image par le *Kinect Body Tracking SDK* et est comparée à la position précédente. La différence de ces deux positions est le vecteur de déplacement pour l'intervalle de temps s'étant écoulé entre les deux images.

Le vecteur de translation correspondant à l'une des mains choisie comme la main de contrôle est ensuite interprété pour simuler le mouvement d'un objet virtuel sur un plan perpendiculaire à l'axe optique de la caméra, tel qu'il serait vu par l'utilisateur (figure 4.1). L'équation 4.1 montre la conversion entre le vecteur de translation 3D dans le repère de la caméra et le mouvement de l'objet virtuel.

$$\begin{cases} t'_y = kt_y \\ t'_z = kt_z \end{cases} \tag{4.1}$$

où t'_y et t'_z sont les coordonnées du vecteur de translation dans le repère de l'objet virtuel, t_y et t_z, les composantes du vecteur de translation dans le repère de l'utilisateur, et k, un facteur multiplicatif choisi arbitrairement pour modifier l'amplitude du mouvement.

Cette relation générale peut également être adaptée à la commande d'un curseur de souris, auquel cas la conversion devient celle décrite à l'équation 4.2.

$$\begin{cases} u = kt_y - u_0 \\ v = -kt_z - v_0 \end{cases} \tag{4.2}$$

où u et v sont les coordonnées du curseur virtuel à l'écran et (u_0, v_0) est le vecteur de translation vers le repère de l'écran, dont les composantes sont définies comme la moitié de la largeur et de la hauteur de l'écran respectivement.

Il s'agit de deux interprétations de la translation de la main adaptées à des cas d'utilisation différents du prototype fonctionnel.

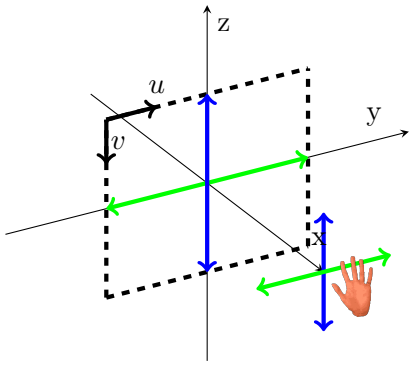


FIGURE 4.1 – Système de référence pour la métaphore de translation.

4.2 Métaphores de rotation

Deux métaphores d'interaction seront étudiées pour la reconnaissance de commandes de rotation par l'utilisateur. La première s'inspire du fonctionnement du curseur de souris dans plusieurs logiciels d'édition 3D, dont VXelements, où l'utilisateur effectue un mouvement en translation, lequel est interprété comme une rotation autour d'un point de pivot virtuel. La seconde propose d'utiliser l'orientation 3D de la main de l'utilisateur dans l'espace et de reproduire cette rotation dans le logiciel. Cette section présente les deux méthodes et discute de leur applicabilité au problème de l'encodage de commandes de rotation pour le logiciel VXelements.

4.2.1 Rotation par translation d'un curseur virtuel

Nous proposons la conversion décrite à l'équation 4.3 pour l'interprétation du mouvement de translation d'une main de l'utilisateur en une matrice de rotation 3D d'un objet situé à l'origine. La figure 4.2 illustre cette opération.

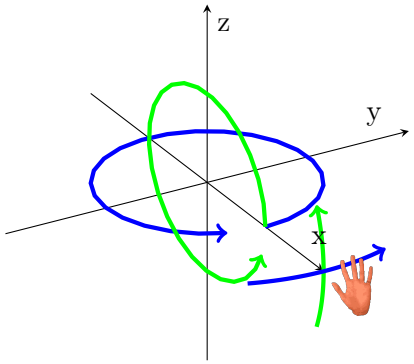


FIGURE 4.2 – Interprétation de la rotation à partir du mouvement en translation de la main.

$$Euler_{XYZ} = \begin{Bmatrix} 0 \\ -kt_z \\ kt_y \end{Bmatrix} \tag{4.3}$$

où $Euleur_{XYZ}$ est la rotation exprimée sous la forme d'angles d'Euler suivant la convention de rotation intrinsèque $x-y'-z''$, t_y et t_z sont respectivement les seconde et première composantes du vecteur de translation dans le repère de l'utilisateur et k est un facteur multiplicatif pour modifier l'amplitude du mouvement.

On note l'inversion de signe de l'angle Y, laquelle assure que le mouvement de la main selon l'axe Z produit un mouvement dans le même sens d'un point situé sur la surface de l'objet. Cette interprétation de la rotation correspond au mouvement que l'on observerait en manipulant un globe terrestre.

4.2.2 Rotation par estimation de la pose 3D de la main

Littérature sur l'estimation de pose 3D

Les méthodes classiques pour l'estimation de la pose 3D d'un objet se basent sur la géométrie du problème en effectuant l'appariement de points d'intérêt en 2D d'une ou plusieurs images avec des points 3D connus [87]. Par exemple, [88] décrit plusieurs méthodes analytiques et itératives pour l'appariement de points entre des vues projetées en 2D afin de permettre l'estimation de la pose 3D. [89] évite l'étape d'appariement de points en effectuant un alignement selon un modèle exprimé sous la forme d'un polynôme implicite.

Plus récemment, de nombreuses approches basées sur l'apprentissage profond ont été employées, entre autres pour pallier la faiblesse des approches basées sur l'appariement pour les objets peu texturés [87, 90]. Certaines méthodes tentent d'inférer la pose directement à partir des images dans une approche bout-en-bout [91, 90, 92, 93], et d'autres emploient un réseau profond uniquement pour l'étape d'appariement de point 2D-3D [94, 95, 96, 97]. Par exemple, [90] propose un réseau à convolution entraîné bout-en-bout qui permet la régression

de la translation 3D et de la rotation exprimée sous la forme d'un quaternion et enrichit son ensemble de données d'entraînement d'images de synthèse. Dans un contexte de suivi continu, [51] ajoute une composante de cohérence temporelle en incluant la prédiction faite pour l'image précédente afin de limiter l'erreur accumulée.

Certaines méthodes s'intéressent plus spécifiquement au suivi du corps humain comme objet articulé. [98] présente un réseau de neurones profond adapté précisément au suivi de la main et des doigts, entraîné à l'aide de données annotées au moyen d'un système de capture de mouvement. Les auteurs emploient la distance euclidienne comme fonction de perte lors de l'entraînement et effectuent la régression directement sur les coefficients de la matrice de rotation, en ajoutant en étape de post-traitement une opération de régularisation pour assurer en assurer la validité. [99] étudie l'estimation de pose de la main et des doigts à l'aide de données RGB-D captées dans un repère égocentrique, en employant un système similaire de capture de mouvement.

Approche expérimentale pour l'estimation de l'orientation 3D de la main

Afin d'explorer l'applicabilité de l'estimation de la pose de la main à l'interprétation de commandes de rotation, une implantation est proposée en utilisant un réseau de neurones profond à convolution. L'approche explorée propose de considérer le poing fermé de l'utilisateur comme un objet rigide pour lequel on souhaite prédire la rotation 3D, laquelle serait calquée sur un objet à l'écran (figure 4.3). Pour ce faire, les architectures de réseau ResNet18 et ResNet50 sont privilégiées pour la disponibilité d'implantations de références dans la librairie *PyTorch*, similairement à la démarche de la section 2.5. La sortie des réseaux est adaptée pour permettre une régression des composantes du quaternion de rotation comme dans [90], ou les composantes du vecteur de rotation. Pour évaluer les variantes étudiées, la métrique d'erreur de prédiction employée est la moyenne de la rotation résiduelle sur l'ensemble de données `vicon` en entier. La rotation résiduelle est exprimée comme l'amplitude de l'angle résultant du produit de la rotation prédite et de l'inverse de la rotation attendue, lorsqu'exprimée sous la notation axe-angle (équation 4.4).

$$\text{erreur} = \frac{\sum_n^N \|\text{rotvec}(R_{prédit} R_{attendu}^T)\|}{N} \tag{4.4}$$

où N est le nombre d'échantillons présents dans l'ensemeble de données, $R_{prédit}$ et $R_{attendu}$ sont les matrices de rotation prédites et attendues, et rotvec représente la fonction de conversion d'une matrice de rotation 3D en un vecteur de rotation.

Chaque réseau a été entraîné selon un horaire d'entraînement avec un taux d'apprentissage variable en trois phases, soient 5 époques à 0.001, 20 époques à 0.0001 et 20 époques à 0.00001. L'état du réseau à l'époque ayant la perte la plus faible sur l'ensemble de validation est utilisée comme point de départ à l'étape suivante. Les ensembles d'entraînement et de validation sont

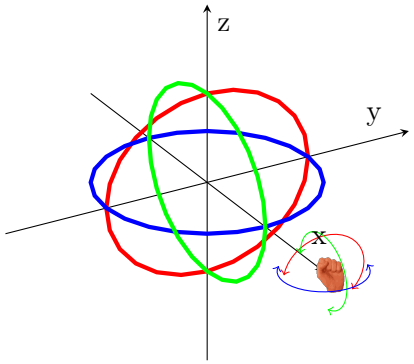


FIGURE 4.3 – Interprétation de la rotation à partir de l'orientation 3D de la main.

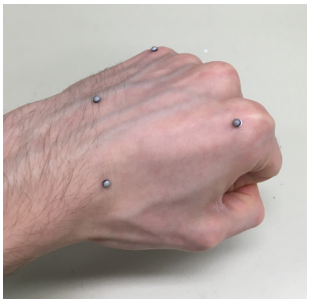


FIGURE 4.4 – Marqueurs utilisés pour la système de capture du mouvement *Vicon Tracker*.

constitués respectivement de 80% et 20% des données du ou des ensembles d'entrée. Les images d'entrée sont redimensionnées en une taille de 64×64 pixels correspondant à une fenêtre de 250×250 mm autour de la main dans l'espace 3D.

Ensembles de données utilisés pour la régression de la rotation 3D de la main

Le tableau 4.1 liste les deux ensembles de données constitués pour le problème de la régression de la rotation 3D de la main. L'ensemble `vicon` a été bâti en synchronisant les données captées au moyen du système *Vicon Tracker* aux images RGB-D du capteur *Azure Kinect*. Pour ce faire, des marqueurs réfléchissants de 2 mm de diamètre ont été apposés au dos de la main de l'utilisateur, ce qui permet le suivi de la main comme un objet rigide (figure 4.4). Pour chaque image captée par l'*Azure Kinect*, une requête est faite au moyen de l'API *DataStream* du logiciel *Vicon Tracker* et la matrice de rotation dans le repère global est sauvegardée. Finalement, la première image de l'ensemble de données est utilisée comme référence, à partir de laquelle chaque autre matrice est calculée.

L'ensemble `synthétique` est bâti à partir de la même scène *Blender* décrite à la section 2.5.1. Un ensemble de 10 000 images de synthèse sont générées pour la configuration du poing fermé en faisant varier les conditions d'éclairage et la disposition de la caméra dans la scène. Le tableau 4.2 présente les plages de valeurs permises pour chacun des paramètres.

Nom	Description	Nombre d'images
`vicon`	Données réelles annotées au moyen du système de capture de mouvement Vicon Tracker.	652
`synthétique`	Images de synthèse générées au moyen du logiciel *Blender*.	10 000

TABLE 4.1 – Description des ensembles de données pour la régression de l'orientation 3D de la main.

Paramètre	Plage de valeurs
Distance Caméra (m)	$[0.25, 2[$
Rotation axe projecteur caméra XYZ (°)	[0,360]; $[-90, 30[$; $[-180, 180[$
Intensité source (W)	$[750, 1250[$
Position source lumineuse XYZ (m)	$[-4,4[$; $[-4, 4[$; $[0, 4[$
Variation angle jointure (°)	2

TABLE 4.2 – Plage permises pour les valeurs aléatoires assignées à chacun des paramètre de rendu pour la régression de l'orientation.

Résultats pour l'estimation de la pose 3D

Cette section compare la performance de plusieurs variantes d'entraînement étudiées. Puisque l'évaluation de l'erreur est effectuée sur l'ensemble `vicon` entier, ce qui inclut les données utilisées dans les étapes d'entraînement et de validation des réseaux. Il est à noter que les résultats sont biaisés en faveur de cet ensemble de données. Néanmoins, il s'agit d'une comparaison pertinente pour mesurer l'effet de chacun des paramètres entre les variantes.

Sortie	Réseau	$P_{entranement}$	$P_{validation}$	E_{moy} (°)
Quaternion	ResNet18	0.39	6.08	9.58
Vecteur rotation	ResNet18	1.53	29.23	7.91

TABLE 4.3 – Comparaison des sorties de réseau pour la régression de rotation 3D. Les valeurs de P correspondent aux valeurs moyennes de sortie de la fonction de perte, et E_{moy}, à l'erreur moyenne de prédiction en ° sur l'ensemble `vicon`.

Le tableau 4.3 compare l'effet sur la performance en régression du réseau pour chaque variante de sortie testée. Pour chacune des variantes, la rotation 3D de référence pour chaque image est représentée sous la forme d'un vecteur, représentant soit le quaternion de rotation ou les composantes de la notation axe-angle, au moyen de la bibliothèque *SciPy*. La fonction de perte utilisée est la perte quadratique entre les composantes attendues et prédites de ces vecteurs.

On constate que le réseau ayant appris la régression des quaternions de rotation résulte en une prédiction moins fiable que le réseau ayant appris un vecteur de rotation, avec une différence de leur erreur moyenne sur `vicon` de près de 2°. Il est possible que cette différence soit explicable

par l'impossibilité d'appliquer la contrainte de norme unitaire spécifique aux quaternions de rotation. Il serait pertinent de répéter l'expérience en ajoutant une étape de normalisation pour valider cette hypothèse, similairement à ce qui est proposé par [90].

Fonction perte	**Réseau**	$P_{entranement}$	$P_{validation}$	E_{moy} (°)
Perte L1	ResNet18	2.87	24.49	6.02
Perte L2	ResNet18	1.53	29.23	7.92

TABLE 4.4 – Comparaison des fonctions de perte pour la régression de la rotation 3D. Les valeurs de P correspondent aux valeurs moyennes de sortie de la fonction de perte, et E_{moy}, à l'erreur moyenne de prédiction en ° sur l'ensemble `vicon`.

Le tableau 4.4 compare l'utilisation de différentes fonctions de perte pour l'entraînement des réseaux. On constate que l'utilisation de la perte L1 (erreur absolue moyenne) résulte en une meilleure prédiction, avec une erreur en test inférieure de près de 2° par rapport au réseau entraîné avec la fonction de perte quadratique (L2). Il semble que la pénalité additionnelle portée par la perte L2 aux grandes déviations par rapport à la valeur attendue soit néfaste à l'apprentissage dans ce contexte. Cependant, il est à noter que ces deux fonctions de perte naïve sont mal adaptées au problème, puisqu'elles ne tiennent pas compte de la qualité cyclique des fonctions trigonométriques ou de la symétrie qui peut exister dans l'espace des rotations 3D. Par exemple, l'une ou l'autre de ces fonctions pénalisera fortement une différence de 359° dans une prédiction, alors que celle-ci devrait plutôt être traitée comme une différence de 1°. Une meilleure approche aurait été d'inclure une connaissance *a priori* de l'espace des rotation, tel qu'exploré par [90], qui introduit deux fonctions de pertes spécialisées.

Réseau	**Mode**	$P_{entranement}$	$P_{validation}$	E_{moy} (°)
ResNet18	RGB-D	1.53	29.23	7.92
ResNet50	RGB-D	2.40	29.17	8.18

TABLE 4.5 – Comparaison des architectures de réseau pour la régression de la rotation 3D.

Le tableau 4.5 compare la performance des architectures de réseau utilisées. Comme dans le cas de la tâche de classification des symboles statiques de la main (section 2.5), il ne semble pas y avoir de différence significative résultant de l'utilisation du réseau ResNet50 plutôt que le réseau ResNet18 pour justifier le coût en calculs supplémentaire. Ce résultat est attendu puisqu'il s'agit de deux réseaux de la même famille. Il aurait été pertinent d'examiner l'utilisation d'architectures de réseaux spécialisées dans la régression de la pose ou de l'orientation 3D, tels que ceux proposés par [90, 94], dans le cadre de travaux futurs.

Le tableau 4.6 illustre l'effet de l'utilisation des canaux de l'image sur l'entraînement. Alors que la variante RGB sans canal de profondeur semble avoir la meilleure performance en prédiction, il semble que cet écart soit causé par un sur-entraînement plutôt que par une amélioration réelle. En effet, les variantes Gris et RGB se démarquent par le très grand écart entre la qualité

Réseau	**Mode**	$P_{entranement}$	$P_{validation}$	E_{moy} (°)
ResNet18	Gris	0.04	50.89	9.20
ResNet18	RGB	0.04	44.65	7.22
ResNet18	RGB-D	1.53	29.23	7.92
ResNet18	Gris-D	2.62	28.8	8.25
ResNet18	D	3.07	29.43	8.23

TABLE 4.6 – Comparaison des canaux de couleurs utilisés pour la régression de l'orientation 3D.

de la prédiction en entraînement et en validation, lequel est plus grand que toutes les variantes présentées jusqu'ici. On déduit que l'information contenue dans le canal de profondeur contribue à améliorer la capacité de généralisation d'un ensemble de données à l'autre. Les variantes incluant l'information du canal de profondeur (RGB-D, Gris-D, D) ont une performance similaire en classification, avec la variante RGB-D ayant une erreur moyenne légèrement inférieure aux autres.

Données en entrée	**Mode**	$P_{entranement}$	$P_{validation}$	E_{moy} (°)
Synthétiques et réelles	RGB-D	1.53	29.23	7.92
Réelles seules	RGB-D	0.052	41.12	9.89

TABLE 4.7 – Effet de l'utilisation des données synthétiques pour la régression de l'orientation 3D.

Le tableau 4.7 présente l'effet de l'inclusion des données synthétiques sur la qualité du réseau entraîné. On constate que l'exclusion des données synthétiques résulte en une augmentation de l'erreur moyenne de près de 2°, ce dont on déduit que l'inclusion de points de vue et de conditions d'éclairage variées améliore la capacité du réseau à généraliser.

Travaux futurs

La méthode présentée dans cette section est de nature exploratoire et offre ainsi une performance insuffisante pour être utilisable dans un contexte d'interaction personne-machine. En effet, l'erreur moyenne absolue de toutes les variantes étudiées se situe entre 6° et 9°, et ce, même lorsqu'évaluée sur un ensemble de données utilisé en partie lors de l'entraînement. Cette précision est nettement insuffisante pour la commande d'un logiciel nécessitant un haut degré de précision tel que VXelements.

Cette piètre performance peut s'expliquer en partie par les données utilisées pour l'entraînement. En effet, l'ensemble `vicon` ne contient que 652 images réelles, ce qui est un faible volume de données pour ce genre de problème. De plus, il est possible que la résolution de 64×64 pixels pour les images en entrée soit trop limitée pour capturer la complexité de la main. Par exemple, [100] propose un réseau prenant en entrée des images de 150×150 pixels, et ce,

pour effectuer le suivi d'objets visuellement plus riches. Il aurait donc été pertinent d'étudier l'utilisation d'ensembles de données plus vastes et captés à une résolution supérieure.

Il aurait également été pertinent d'explorer l'utilisation d'architectures de réseaux plus spécialisées dans la reconnaissance de pose 3D, telles que celle proposée par [90]. De même, alors que le modèle présenté ici n'effectuait que la régression de la rotation indépendamment sur chaque image de la séquence, il aurait été pertinent d'inclure la contrainte de cohérence temporelle dans le modèle, comme présenté dans [100]. Puisque le contexte du contrôle de haute précision implique un changement faible dans l'orientation entre deux images subséquentes, on peut supposer que cette approche permettrait également de réduire l'erreur dans ce problème-ci.

4.3 Conclusion

Deux types d'interactions isomorphes ont été étudiées dans le contexte de la commande d'un logiciel d'édition 3D. Nous avons montré que la métaphore du curseur de souris, selon laquelle un mouvement en translation de la main est reproduit à l'écran, peut être aisément appliquée à la fois à des opérations de translation et de rotation dans le logiciel. Nous avons également appliqué un réseau de neurones profond à la tâche de suivi de l'orientation 3D de la main pour tenter de la reproduire à l'écran. Nous avons cependant démontré que la précision des réseaux entraînés était insuffisante pour être applicable à l'issue de ces travaux exploratoires. Par conséquent, seules les approches basées sur la reconnaissance du mouvement en translation de la main seront utilisées dans l'implantation du prototype fonctionnel.

Chapitre 5

Prototype fonctionnel

Ce chapitre s'intéresse à l'application des modèles gestuels à la création d'un module d'interaction personne-machine dans le contexte de l'utilisation d'un scanner 3D tenu en main, tels que ceux décrits au chapitre d'introduction (figure 0.1). À cet effet, nous effectuons d'abord un survol des fonctionnalités du logiciel VXelements en portant une attention particulière aux cas d'utilisation compatibles avec une interface gestuelle. Ensuite, nous présenterons dans le détail l'implantation logicielle du prototype. Une vidéo montrant le prototype fonctionnel en action est disponible à l'adresse `https://www.youtube.com/watch?v=O2IOQPO_Xyc`.

5.1 Cas d'utilisation du logiciel VXelements

VXelements est la plateforme logicielle de métrologie de Creaform. La figure 5.1 montre la vue principale du logiciel à l'ouverture d'une session de travail, lequel se trouve alors en mode édition. Dans le haut de l'écran, on retrouve le ruban de commandes, dans lequel se trouvent des boutons pour contrôler les étapes de la numérisation, par exemple pour débuter ou arrêter la capture d'images, reprendre la numérisation à partir d'un certain point de vue et recommencer à neuf. Le ruban contient également des actions d'édition simplifiées, telles que la suppression ou la fusion de certaines surfaces de maillage dans la session de travail courante.

Lorsque l'utilisateur débute une séance de numérisation au moyen du bouton *Scan*, le logiciel bascule en mode capture, ce qui a pour effet de désactiver les fonctions d'édition du maillage 3D, et autorise l'activation du capteur au moyen de la détente se trouvant sur l'appareil. Dans ce mode, le lissage de la surface est également désactivé, ce qui a pour effet de montrer à l'utilisateur une représentation plus fidèle du nuage de points capté jusqu'à présent (figure 5.2).

Dans ces deux modes, le panneau central permet la visualisation du modèle 3D en cours de construction. L'utilisateur peut appliquer une rotation de l'objet en tenant enfoncé le bouton gauche de la souris, une translation avec le bouton milieu, et une mise à l'échelle au moyen

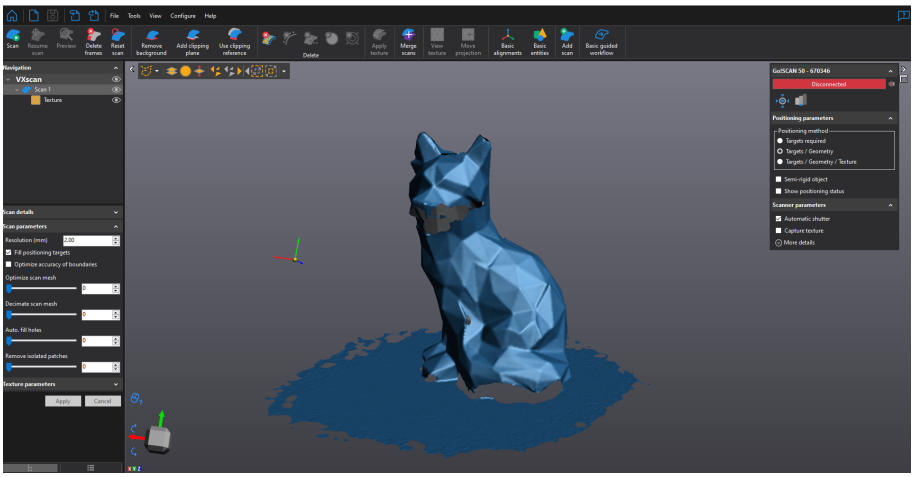


FIGURE 5.1 – Vue principale du logiciel Creaform VXelements (mode édition).

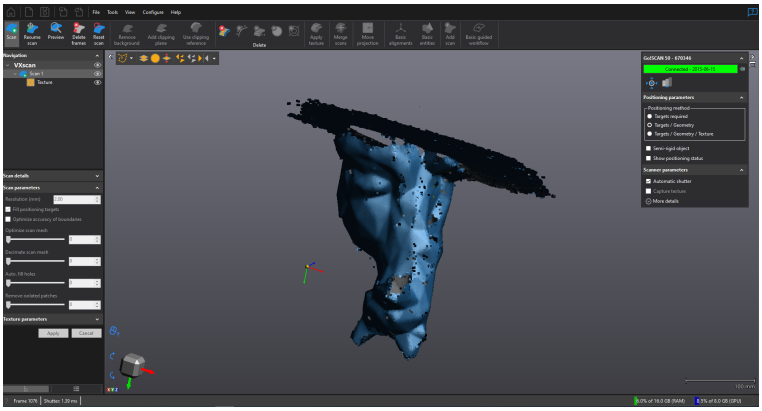


FIGURE 5.2 – Vue lorsque le mode de capture est activé.

de la molette. L'utilisation du bouton droit de la souris dévoile un menu contextuel contenant des options avancées de manipulation de la vue, telles que l'application d'un point de vue prédéfini d'après l'un des axes du système de référence de l'objet (figure 5.3). Dans le coin inférieur gauche, se trouve une visualisation des axes de référence depuis le point de vue actuel, lequel peut également être utilisé pour aligner rapidement la caméra selon l'un des axes principaux.

Durant le scan, l'opérateur peut utiliser le bouton *Resume scan* pour choisir un point de vue à partir duquel reprendre la captation d'images. Les surfaces visibles depuis le nouveau point de vue apparaissent en surbrillance à l'intérieur d'un rectangle représentant le plan image du capteur. Cette opération est utile pour informer le logiciel d'un repositionnement majeur, par exemple pour capter l'envers d'un objet, ce qui a pour effet de grandement accélérer le verrouillage de l'orientation dans le nouveau point de vue.

Le panneau de gauche montre une vue en arbre des objets présents dans la scène, lesquels

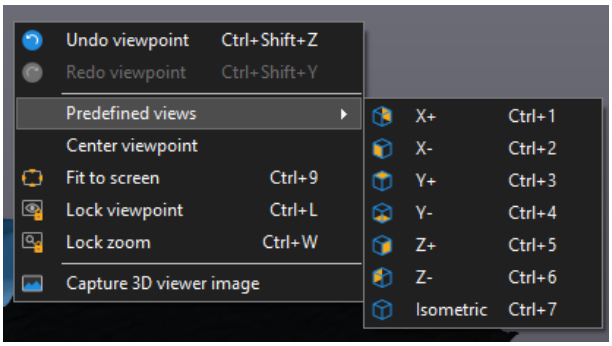


FIGURE 5.3 – Menu contextuel permettant la manipulation des points de vue.

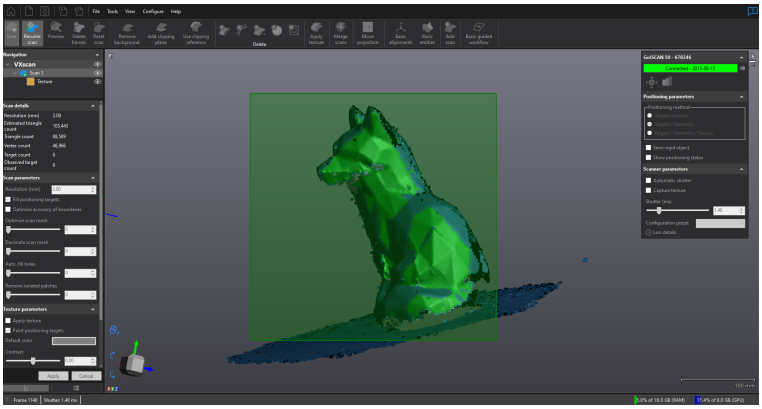


FIGURE 5.4 – Sélection du point de vue pour la reprise de la numérisation.

peuvent être individuellement affichés ou cachés du panneau de visualisation. Sous le panneau de navigation, des statistiques, telles que le nombre de faces et de sommets du maillage de la surface, et des réglages avancés de la reconstruction de surface sont affichés (figure 5.5). Les réglages ne sont accessibles que depuis le mode édition. À droite du panneau central de visualisation, on retrouve le panneau de réglages du capteur, qui montre l'état de connexion du capteur et permet de configurer la méthode de positionnement et le temps d'exposition. Les assistants de calibration et d'ajustement du temps d'exposition sont accessible depuis cet onglet.

La figure 5.6 montre l'assistant de calibration. La calibration s'effectue au moyen d'une plaque spécialisée fournie avec le capteur (figure 5.7). Il est attendu que l'opérateur effectue cette tâche régulièrement afin de garantir la précision et l'exactitude des numérisations effectuées.

La figure 5.8 montre l'assistant de configuration du temps d'exposition du capteur. Dans ce menu, le bouton *Auto. adjust* effectue l'ajustement automatique de ce paramètre de sorte à maximiser les régions d'image exposées de manière optimale. Le panneau de visualisation donne une rétroaction en direct des paramètres appliqués sur la vue actuelle du capteur. Un opérateur pourrait souhaiter utiliser ce mode durant le scan pour mieux capturer les détails

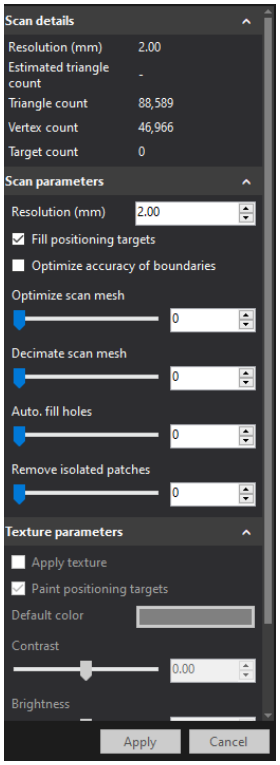


FIGURE 5.5 – Panneau de réglage des paramètres de scan.

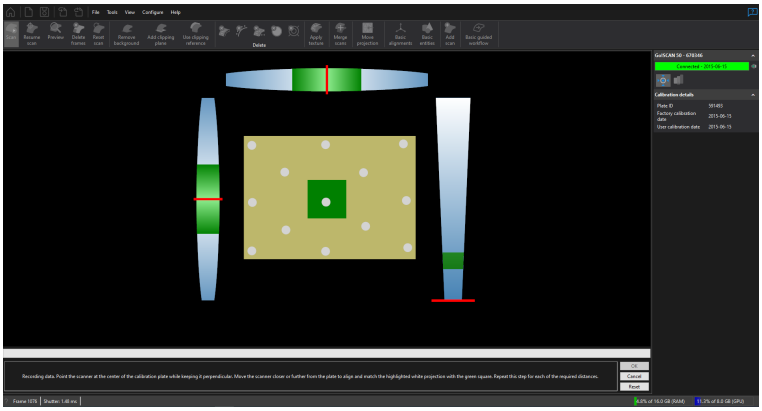


FIGURE 5.6 – Assistant de calibration du capteur.

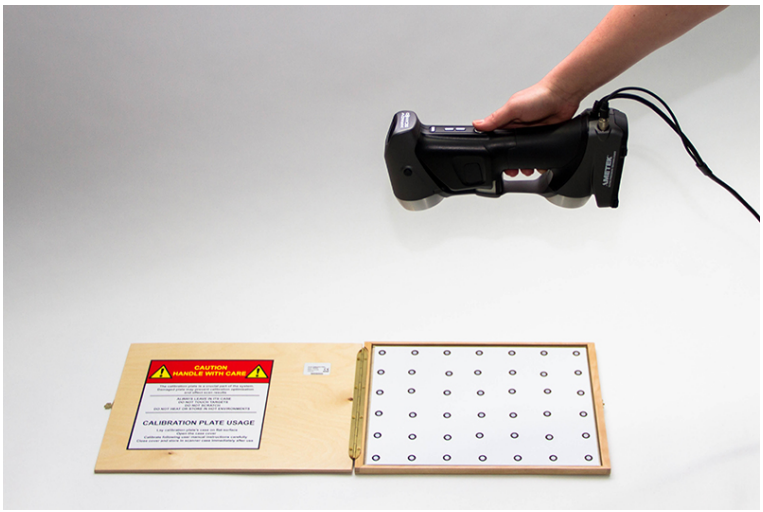


FIGURE 5.7 – Utilisation de la plaque de calibration. Tiré de [101].

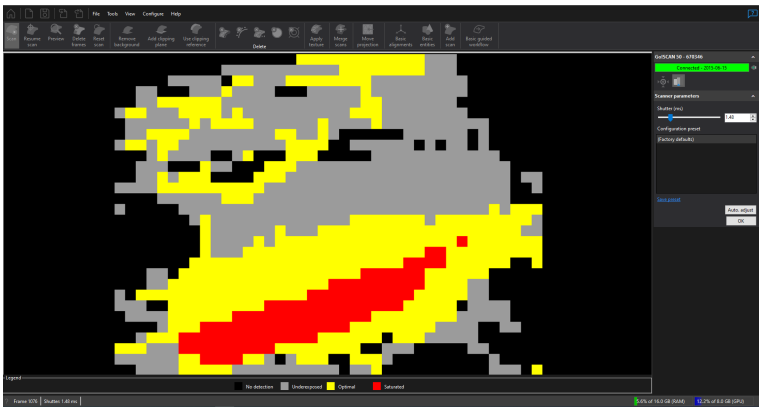


FIGURE 5.8 – Assistant de réglage du temps d'exposition.

d'une région moins bien éclairée de la scène.

Les capteurs tenus en main permettent l'utilisation de plusieurs méthodes de positionnement du capteur, qui peuvent être sélectionnées en fonction des spécificités de l'objet à numériser. La méthode *Targets required* force l'utilisation unique des cibles autocollantes rétroréfléchissantes, *Targets / Geometry* permet l'utilisation de points d'intérêts 3D identifiés dans la géométrie pour améliorer le positionnement, et *Targets / Geometry / Texture* ajoute en plus des points d'intérêts 2D tirés de la texture de l'objet. De manière générale, l'utilisation du plus grand nombre de points de référence assure une opération plus fiable et rapide. Cependant, un opérateur peut souhaiter limiter la méthode de positionnement à un ou plusieurs types de points d'intérêt dans certains contextes difficiles, par exemple dans le cas de surfaces réfléchissantes ou peu texturées.

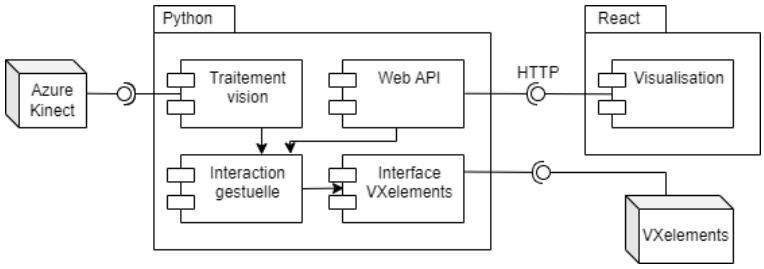


FIGURE 5.9 – Aperçu des composantes de haut niveau.

5.2 Implantation logicielle

Le prototype fonctionnel prend la forme d'une application Python responsable du traitement des données provenant du capture *Azure Kinect* et de l'interprétation des commandes gestuelles de l'utilisateur, ainsi que d'une application web *React* pour la visualisation de l'état du logiciel. La figure 5.9 montre ces composantes de haut niveau.

5.2.1 Module de traitement

Le module de traitement est responsable de la capture des données en provenance de la caméra *Kinect*, d'exécuter les modèles de classification et de suivi décrits aux chapitres précédents.

Afin d'éviter le déclenchement accidentel, le module de traitement définit un volume d'activation de gestes autour du torse et de la tête de l'utilisateur, en dehors duquel les gestes peuvent être ignorés aux étapes ultérieures du traitement. Le volume a été défini empiriquement selon l'équation 5.1.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} [\frac{d}{4}, 2d] \\ [0, d] \\ [-d, d] \end{pmatrix} \tag{5.1}$$

où X, Y, Z sont exprimés dans le repère de l'utilisateur et d est la distance entre le cou et le bas de la colonne vertébrale.

Étant donné que le module est responsable de la majorité du traitement en temps réel du système, celui-ci est séparé du processus principal et communique avec ce dernier de manière asynchrone au moyen d'une file de messages (figure 5.10). Pour chaque image captée, une instance de la classe message est créée pour contenir le résultat du traitement. Le tableau 5.1 liste les attributs contenus dans chaque instance. La file de messages utilisée est de taille fixe et les nouveaux messages écrasent les plus anciens afin de garantir une faible latence en cas de ralentissement du traitement.

Le modèle de classification des formes de la main utilisé est la variante RGB-D du réseau de neurones ResNet18 entraîné sur des données synthétiques et réelles, et le classificateur de gestes dynamiques est la variante aux axes normalisés. Le processus de traitement effectue la

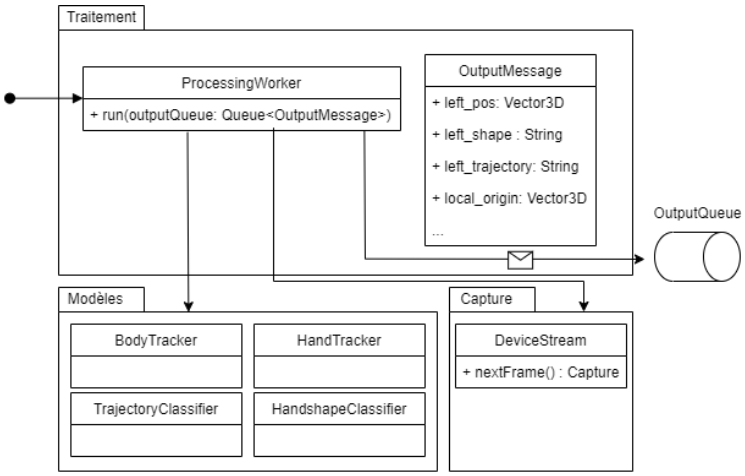


FIGURE 5.10 – Module de traitement

Attribut	Type	Description
`left_pos` `right_pos`	`Optional<Vector3D>`	Position 3D de la main dans le repère caméra. `null` si elle n'est pas détectée.
`left_class` `right_class`	`String`	Classe reconnue par le classificateur de forme de la main. Vide si aucune forme n'est reconnue.
`left_in_volume` `right_in_volume`	`Boolean`	Signifie la présence de la main à l'intérieur du volume d'activation des gestes.
`left_trajectory` `right_trajectory`	`String`	Classe reconnue par le classificateur de gestes dynamiques. Vide si non reconnue.
`local_origin`	`Optional<Vector3D>`	Origine du repère de l'utilisateur dans le repère caméra. Correspond à la position de l'articulation `SPINE_CHEST` retournée par le *Kinect Body Tracking SDK*.
`timestamp_ms`	`Integer`	Instant de capture de l'image traitée dans le format UNIX en millisecondes. Utilisé pour diagnostiquer des problèmes de performance en temps de traitement.

TABLE 5.1 – Description du format des messages de sortie du module de traitement.

captation vidéo à un rythme de 5 images par seconde, à une résolution de 1280×720 sur la caméra couleur et 640×576 sur le capteur de profondeur.

5.2.2 Module d'interaction utilisateur

Le module d'interaction est conçu comme une machine à états représentant les modes d'utilisation du logiciel (figure 5.11). Chaque transition du graphe correspond à un geste reconnu, et s'accompagne optionnellement d'une action à déclencher dans VXelements. Par exemple,

État	Description
Base	État de base à l'ouverture d'une session de travail.
Scan	Mode *Scan* de VXelements. Le capteur 3D est prêt à être activé.
Contrôle vue	Permet le contrôle de la vue (translation, rotation, mise à l'échelle). Accessible depuis le mode de base et scan (omis du schéma). Composé de sous-états pour la rotation, la translation et la mise à l'échelle.
Réglages	Contient des actions pour le réglage des méthodes de positionnement et d'exposition.
Reprise scan	Mode interactif pour activer la fonctionnalité *Resume Scan* de VXelements. Composé d'états de bas-niveau pour le contrôle de la vue et du curseur de souris pour permettre le choix de la position de reprise.

TABLE 5.2 – Description des états de haut niveau du module d'interaction utilisateur.

à l'état de base du système, le pouce en l'air permet de débuter une séance de numérisation, ce qui fait transitionner l'état courant à l'état de scan, et active le mode équivalent de VXelements. Le tableau 5.2 présente les principaux des états du système.

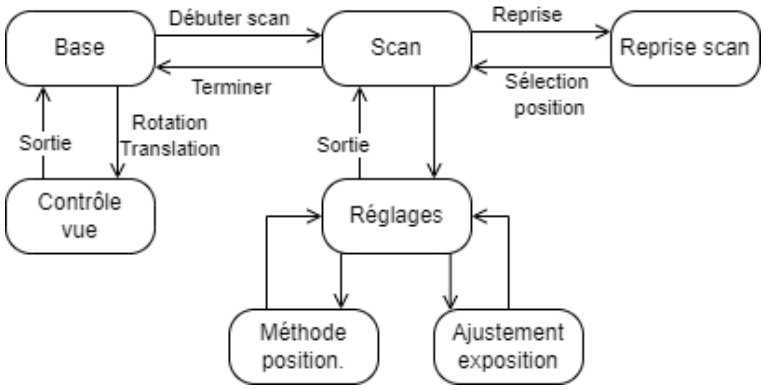


FIGURE 5.11 – Représentation simplifiée des états de l'interface utilisateur.

La machine à états est implantée au moyen d'une structure en graphe, dans laquelle chaque nœud contient la liste des transitions vers ses voisins. Chaque transition contient optionnellement la description d'une action externe à déclencher (figure 5.12). Ce graphe est contenu dans un objet `GestureStateMachine` responsable de consommer les messages en provenance du module de traitement, parcourir le graphe et déclencher les actions externes de manière synchrone.

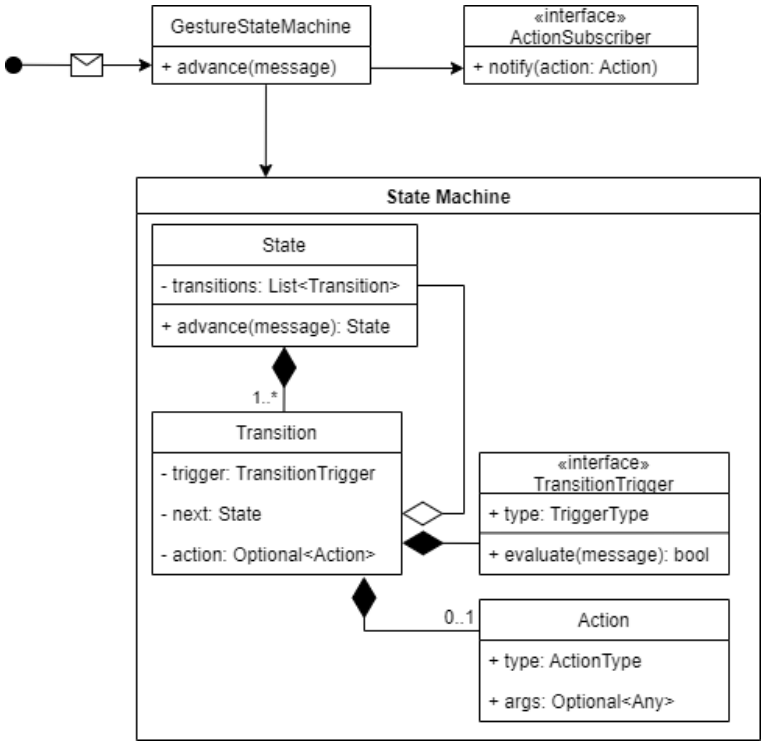


FIGURE 5.12 – Diagramme de classes du module d'interaction.

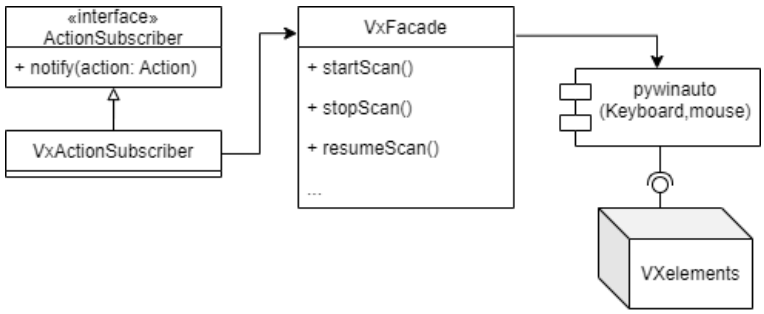


FIGURE 5.13 – Diagramme de classes de l'interface VXelements.

5.2.3 Interface VXelements

Le contrôle du logiciel VXelements est effectué par émulation de touches de clavier et de souris. Pour ce faire, la librairie Python *pywinauto* est utilisée, laquelle offre des modules d'abstraction autour de l'API d'automatisation de *Windows* [102].

Actions momentanées

Les actions momentanées sont déclenchées en simulant un clic de souris dans le ruban de commandes de VXelements ou en simulant le raccourci clavier correspondant à la commande, s'il existe. Le tableau 5.3 liste les actions momentanées déclenchées par le module d'interface.

Action	Description
Démarrer scan	Équivalent au bouton *Start Scan* du ruban de commandes.
Terminer scan	Terminer l'acquisition d'images. Équivalent au bouton *Stop Scan* accessible lorsqu'une session est active.
Configurer méthode positionnement	Sélection de la méthode de positionnement en simulant des clics sur les boutons de sélection *Targets required*, *Targets / Geometry* ou *Targets / Geometry / Texture*.
Activer/désactiver exposition automatique	Active ou désactive le bouton de sélection *Automatic shutter*.
Débuter ajustement exposition	Active l'assistant d'ajustement du temps d'exposition, et active le bouton *Auto. adjust.*
Terminer ajustement exposition	Met fin à l'assistant d'ajustement du temps d'exposition en activant le bouton *OK*.
Débuter reprise scan	Active le mode *Resume scan* en activant le curseur de sélection de vue.
Terminer reprise scan	Confirme la sélection de la vue de reprise en activant le clic gauche de la souris à la position actuelle du curseur.
Annuler reprise scan	Met fin au processus de reprise du scan en simulant la touche d'échappement.

TABLE 5.3 – Actions momentanées déclenchées par le module d'interaction.

Actions isomorphes

Les actions isomorphes sont limitées exclusivement au contrôle de la vue du panneau de visualisation. Celles-ci offrent une rétroaction visuelle en temps réel des changements appliqués jusqu'à ce que l'utilisateur mette fin à la commande au moyen du geste de sortie illustré.

La commande de translation accumule les vecteurs de déplacement de la main entre chaque paire d'images consécutives et calcule la position résultante du curseur de souris à chaque instant selon l'équation 5.2.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 + k\Delta x \\ v_0 - k\Delta y \end{pmatrix} \tag{5.2}$$

où u et v sont les coordonnées du curseur de souris, u_0 et v_0 les coordonnées du centre de l'écran, Δx et Δy les coordonnées du vecteur de déplacement accumulé depuis le début de la commande de translation, et k une constante de mise à l'échelle.

Le bouton central de la souris est enfoncé lors du démarrage de la commande de translation, et relâché lorsque l'utilisateur y met fin.

La commande de rotation est implantée selon le même mécanisme. Plutôt qu'un vecteur de déplacement, la commande de rotation accumule une matrice de rotation en fonction des rotations calculées entre chaque paires d'images. La conversion de la rotation en position du curseur de souris est décrite à l'équation 5.3.

$$\begin{gathered} D = \vec{r} \times [1, 0, 0]^T \\ \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 + kD_y \\ v_0 - kD_z \end{pmatrix} \end{gathered} \tag{5.3}$$

où u et v sont les coordonnées du curseur de souris, u_0 et v_0 les coordonnées du centre de l'écran, $\vec{r}$ la rotation accumulée exprimée sous la forme d'un vecteur de rotation, et k une constante de mise à l'échelle choisie empiriquement pour limiter l'erreur accumulée. Une valeur de $k = 57{,}25$ a été utilisée pour un ordinateur ayant une résolution de 1920×1080 sans mise à l'échelle des applications par le système d'exploitation.

Finalement, la commande de mise à l'échelle est modulée en simulant un déplacement variable de la molette de souris. Le nombre de déplacements discrets est calculé d'après l'observation empirique selon laquelle six déplacements engendrent un doublement de la taille de l'objet visible à l'écran (équation 5.4).

$$d = 6\log_2 f \tag{5.4}$$

où d est le nombre de déplacements discrets de la molette de souris, et f est le facteur d'échelle demandé en entrée.

Étant donné la faible résolution de la commande de mise à l'échelle, l'interface utilisateur est conçue de manière à ne permettre que l'application de facteurs prédéfinis ($\frac{3}{2}$ et $\frac{2}{3}$).

5.2.4 Visualisation

Afin d'offrir une rétroaction en temps réel à l'utilisateur, une application web a été réalisée en utilisant la bibliothèque *React*. Celle-ci affiche en tout temps l'état courant de l'application et affiche des pictogrammes pour lister toutes les actions disponibles (figure 5.14).

La rétroaction en temps réel est implantée au moyen d'un serveur HTTP mis à jour de manière asynchrone par le processus principal (figure 5.15). À chaque fois que l'état courant est mis à jour, une notification est envoyée processus web à travers une file de messages, et celui-ci met à jour sa représentation interne. Le serveur HTTP est implanté au moyen de la bibliothèque *Jivago* [103].

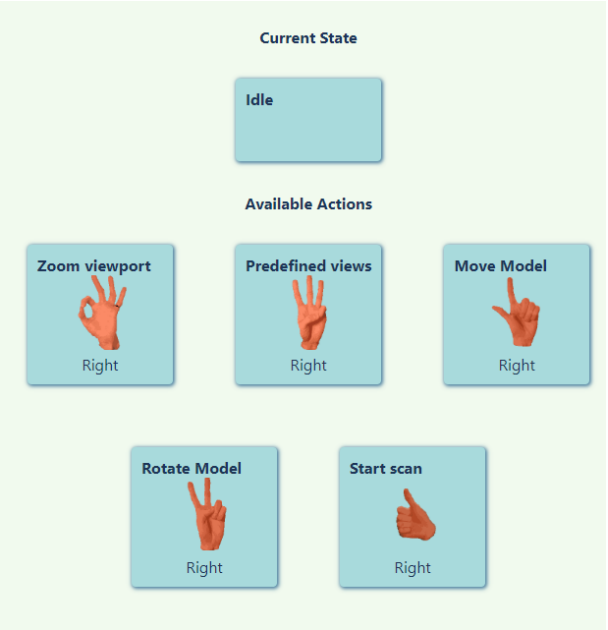


FIGURE 5.14 – Capture d'écran de l'interface de visualisation.

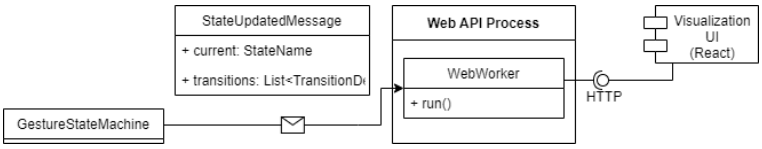


FIGURE 5.15 – Diagramme de classes simplifié du module de visualisation.

5.3 Conclusion

Le prototype fonctionnel développé parvient à démontrer l'utilisation de plusieurs cas d'utilisation du logiciel VXelements à distance lors d'une session de scan. En effet, les fonctionnalités implantées ont été choisies puisqu'elles étaient susceptible d'être employées à plusieurs reprises par un utilisateur se déplaçant autour d'un objet, lesquelles ont certaines ressemblances avec les fonctionnalités du menu *Smart control* disponible sur certains capteurs. Les réglages rapides du capteur et de l'exposition, ainsi que le contrôle de la vue 3D sont des exemples de telles fonctionnalités.

Néanmoins, l'implantation actuelle du prototype présente quelques lacunes pouvant rendre son utilisation complexe. Tout d'abord, la grande utilisation des gestes statiques pour la conception de l'interface utilisateur complexifie l'activation de certains réglages, lesquels peuvent se retrouver derrière des séquences de plusieurs gestes. Alors que plusieurs gestes ont été choisis pour être facilement mémorisés par l'utilisateur (ex : pouce vers le haut pour débuter la session), d'autres peuvent sembler arbitraires pour un utilisateur novice. De plus, des fonctionnalités telles que celle de la reprise du scan (*resume scan*) sont constituées de plusieurs

états internes, à travers lesquels la navigation dépend du rythme d'exécution des gestes.

Bien que l'organisation du prototype en une machine à états permette l'exécution de gestes d'une complexité arbitraire, elle introduit l'inconvénient d'une configuration difficile. En effet, un utilisateur souhaitant personnaliser sont interface gestuelle doit modéliser sont processus de travail sous la forme d'un graphe dans le code de l'application. De plus, l'implantation actuelle initialise cette structure de données de manière procédurale, en nécessitant de l'utilisateur une compréhension parfaite des états et des relations cycliques qui peuvent exister dans le logiciel.

La captation des données est limitée à un rythme de cinq images par seconde dans la version actuelle du logiciel. Cette limite s'explique en partie par l'utilisation de modèles de reconnaissance de gestes dynamiques restreints à cette fréquence de capture. Dans leur implantation actuelle, une plus grande fréquence engendrerait un coût en temps de calcul substantiel, rendant le traitement en temps réel difficile. Toutefois, en omettant l'utilisation de tels modèles, il serait possible de permettre un contrôle beaucoup plus fluide, ce qui serait pertinent pour le contrôle isomorphe en translation et rotation.

Dans le même ordre d'idées, les trois interactions isomorphes étudiées (translation, rotation, mise à l'échelle) souffrent d'un manque de précision en raison du recours à l'encodage sous la forme de commandes de la souris. Ce phénomène est particulièrement évident pour la mise à l'échelle, tant celle-ci est limitée à l'application de facteurs discrets. Une intégration de bas niveau à l'API d'automatisation de VXelements aurait permis de limiter cette perte de précision.

Conclusion

Dans le cadre de ce projet, nous avons conçu un prototype d'interface gestuelle pour la commande d'un logiciel de capteur 3D tenu en main en portant une attention particulière aux fonctionnalités utilisées lors du maniement du capteur. Bien que l'élaboration du prototype ait été guidée par le logiciel VXelements, nous considérons que les métaphores d'interaction et les cas d'utilisation identifiés dans le cadre de ce projet sont suffisamment génériques pour être applicable à d'autres logiciels de métrologie.

Pour ce faire, nous avons employé le capteur *Azure Kinect* et les bibliothèques logicielles associées comme point de départ pour le suivi du corps de l'utilisateur. La disponibilité du capteur et son abondance dans la litérature sur l'interaction personne-machine ont permis le développement rapide d'une base pour la modélisation des gestes de haut niveau pour l'application au problème des scanner 3D. L'inclusion d'un capteur de profondeur a également permis l'estimation précise de la position 3D des mains de l'utilisateur, ce qui a assuré une modélisation robuste des interactions isomorphes et de gestes dynamiques dans l'ensemble du champ de vue de la caméra, ce qui aurait été difficile en n'employant que des images RGB.

Parmi les fonctionnalités du logiciel VXelements, nous avons identifié deux types d'action à modéliser, soient les actions momentanées et continues. Les actions momentanées ont été modélisées au moyen de classificateurs de gestes statiques et dynamiques. Les gestes statiques ont été élaborés en s'inspirant de la littérature sur les langues des signes, spécifiquement en ce qui a trait aux formes de la main telles qu'elles peuvent apparaître dans les alphabets manuels. Nous avons exploré l'utilisation de plusieurs algorithmes de classification pour cette tâche, dont les réseaux de neurones à convolution.

Nous avons implanté la reconnaissance des gestes dynamiques au moyen d'un algorithme de plus proche voisin en employant le *Dynamic Time Warping* comme métrique de distance. La performance en classification est faible pour plusieurs des gestes étudiés, ce qui a limité leur utilisation dans le prototype fonctionnel.

Nous avons également modélisé les actions liées au contrôle de la vue 3D du logiciel sous la forme d'interactions isomorphes, en proposant une interprétation du déplacement de la main de l'utilisateur en translation et rotation d'un objet virtuel à l'écran. Bien que le niveau de

précision atteignable par un tel mécanisme soit inférieur à celui qu'on obtient en utilisant la souris, nous avons démontré l'applicabilité de la fonctionnalité pour un utilisateur devant s'éloigner du poste de travail.

Finalement, nous avons implanté un prototype fonctionnel interagissant avec le logiciel VXelements au moyen d'une simulation de clics de souris et de raccourcis clavier. Cette approche a l'avantage de ne requérir aucun développement supplémentaire de la part du concepteur logiciel, au prix d'une précision diminuée pour plusieurs fonctionnalités, telles que le contrôle du point de vue 3D. Néanmoins, le prototype proposé illustre suffisamment bien le flot de travail étudié.

Bien que les actions implantées dans le prototype logiciel aient été choisies d'après le contenu du matériel de formation au meilleur de nos connaissances, il serait pertinent de considérer le point de vue d'experts techniques dans le domaine de la métrologie dans le cadre de travaux futurs, spécifiquement en ce qui a trait à l'ergonomie et au flot de travail. Une telle étude d'utilisabilité serait nécessaire à la conception d'une interface gestuelle dans un contexte commercial pour identifier les gestes moins intuitifs et les actions les plus susceptibles d'être déclenchées lors d'une séance de scan. L'exploration de nouveaux corpus de gestes, entre autre dans le domaine des gestes dynamiques, et de métaphores d'interaction différentes constituerait aussi une piste de travail pour la mise au point d'une interface plus facile à apprendre pour des utilisateurs débutants.

En ce qui a trait aux modèles développés, il serait également pertinent d'évaluer leur performance pour des utilisateurs différents en travaux futurs. La création d'ensembles de données plus variés pourrait augmenter la robustesse des réseaux de neurones entraînés à cet égard.

En dehors du domaine de la métrologie, le prototype conçu peut servir de point de départ pour la conception d'interfaces unimanuelles en général, lesquelles pourraient être applicables dans plusieurs contextes où un opérateur d'équipement conserve une main libre en s'éloignant de son poste de travail.

Annexe A

Glossaire

Terme	Définition
SVM	*Support-vector machine*. Famille d'algorithmes d'apprentissage automatique.
SVC	*Support-vector clustering*. Algorithme de classification basé sur la méthode SVM.
k-NN	*k nearest neighbours*. Famille d'algorithmes d'apprentissage automatiques basés sur la méthode des k plus proches voisins.
HMM	*Hidden Markov Model*. Modèle statistique utilisé, entre autres, en intelligence artificielle.
RGB-D	Image en couleurs (RGB) comportant un canal de profondeur (D).
API	*Application Programming Interface*. Interface entre deux composantes logicielles.
RNN	*Recurrent Neural Network*. Architecture de réseau de neurones profond spécialisée dans le traitement de données temporelles.
LSTM	*Long short-term memory*. Architecture de réseau de neurones profond spécialisée dans le traitement de données temporelles.

Bibliographie

[1] M. Javaid, A. Haleem, R. Pratap Singh, and R. Suman, "Industrial perspectives of 3d scanning : Features, roles and it's analytical applications," *Sensors International*, vol. 2, p. 100114, 2021.

[2] A. Haleem and M. Javaid, "3d scanning applications in medical field : A literature-based review," *Clinical Epidemiology and Global Health*, vol. 7, no. 2, p. 199–210, 2019.

[3] Creaform, "Professional and industrial portable 3d scanners." [Online]. Available : https://www.creaform3d.com/en/portable-3d-scanners

[4] ——, "Go !scan 3d." [Online]. Available : https://www.creaform3d.com/en/handheld-portable-3d-scanner-goscan-3d

[5] ——, "3d scanning software platform and application suite." [Online]. Available : https://www.creaform3d.com/en/metrology-solutions/3d-applications-software-platforms

[6] Wikipedia, "Kinect," Mar 2022. [Online]. Available : https://en.wikipedia.org/wiki/Kinect

[7] A. Toshev and C. Szegedy, "Deeppose : Human pose estimation via deep neural networks," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[8] P. Felzenszwalb and D. Huttenlocher, "Efficient matching of pictorial structures," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662).*

[9] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. C-22, no. 1, p. 67–92, 1973.

[10] "Learning to parse images of articulated bodies," *Advances in Neural Information Processing Systems 19*, 2007.

[11] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited : People detection and articulated pose estimation," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[12] M. Eichner and V. Ferrari, "Better appearance models for pictorial structures," *Proceedings of the British Machine Vision Conference 2009*, 2009.

[13] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," *CVPR 2011*, 2011.

[14] Q. Dang, J. Yin, B. Wang, and W. Zheng, "Deep learning based 2d human pose estimation : A survey," *Tsinghua Science and Technology*, vol. 24, no. 6, p. 663–676, 2019.

[15] Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet : A structure-aware convolutional network for human pose estimation," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[17] Microsoft, "Azure kinect dk documentation." [Online]. Available : https://docs.microsoft.com/en-us/azure/Kinect-dk/

[18] A. Roussos, S. Theodorakis, V. Pitsikalis, and P. Maragos, "Dynamic affine-invariant shape-appearance handshape features and classification in sign language videos," *Gesture Recognition*, p. 231–271, 2017.

[19] R. A. Tennant and M. G. Brown, *The American Sign Language handshape dictionary.* Clerc Books, Gallaudet University Press, 1998.

[20] S. Escalera, V. Athitsos, and I. Guyon, "Challenges in multi-modal gesture recognition," *Gesture Recognition*, p. 1–60, 2017.

[21] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*, 2009.

[22] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Markerless and efficient 26-dof hand pose recovery," *Computer Vision – ACCV 2010*, p. 744–757, 2011.

[23] ——, "Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints," *2011 International Conference on Computer Vision*, 2011.

[24] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3d hand pose estimation from monocular video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, p. 1793–1805, 2011.

[25] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object detection," *Proceedings of IEEE International Conference on Computer Vision*, 1995.

[26] J. Triesch and C. von der Malsburg, "Classification of hand postures against complex backgrounds using elastic graph matching," *Image and Vision Computing*, vol. 20, no. 13-14, p. 937–943, 2002.

[27] W. T. Freeman, K.-i. Tanaka, J. Ohta, and K. Kyuma, "Computer vision for computer games," in *Proceedings of the second international conference on automatic face and gesture recognition*. IEEE, 1996, pp. 100–105.

[28] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video : A new large-scale dataset and methods comparison," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2020, pp. 1459–1469.

[29] S. Liwicki and M. Everingham, "Automatic recognition of fingerspelled words in british sign language," *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2009.

[30] P. Buehler, A. Zisserman, and M. Everingham, "Learning sign language by watching tv (using weakly aligned subtitles)," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[31] H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden, "Sign language recognition using sub-units," *Gesture Recognition*, p. 89–118, 2017.

[32] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International workshop on automatic face and gesture recognition*, vol. 12. Zurich, Switzerland, 1995, pp. 296–301.

[33] Q. Yang, "Chinese sign language recognition based on video sequence appearance modeling," *2010 5th IEEE Conference on Industrial Electronics and Applications*, 2010.

[34] F. Yasir, P. Prasad, A. Alsadoon, and A. Elchouemi, "Sift based approach on bangla sign language recognition," *2015 IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA)*, 2015.

[35] A. Tharwat, T. Gaber, A. E. Hassanien, M. K. Shahin, and B. Refaat, "Sift-based arabic sign language recognition system," *Advances in Intelligent Systems and Computing*, p. 359–370, 2015.

[36] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision : a review of techniques," *journal of Imaging*, vol. 6, no. 8, p. 73, 2020.

[37] Q. Chen, N. D. Georganas, and E. M. Petriu, "Real-time vision-based hand gesture recognition using haar-like features," in *2007 IEEE instrumentation & measurement technology conference IMTC 2007*. IEEE, 2007, pp. 1–6.

[38] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131–153, 2019.

[39] J. Redmon and A. Farhadi, "Yolov3 : An incremental improvement," *arXiv preprint arXiv :1804.02767*, 2018.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[41] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets : Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv :1704.04861*, 2017.

[42] A. C. Carneiro, L. B. Silva, and D. P. Salvadeo, "Efficient sign language recognition system and dataset creation method based on deep learning and image processing," in *Thirteenth International Conference on Digital Image Processing (ICDIP 2021)*, vol. 11878. International Society for Optics and Photonics, 2021, p. 1187803.

[43] M. Asadi-Aghbolaghi, A. Clapés, M. Bellantonio, H. J. Escalante, V. Ponce-López, X. Baró, I. Guyon, S. Kasaei, and S. Escalera, "Deep learning for action and gesture recognition in image sequences : A survey," in *Gesture Recognition*. Springer, 2017, pp. 539–578.

[44] B. Kang, S. Tripathi, and T. 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)*. IEEE, 2015, pp. 136–140.

[45] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4597–4605.

[46] X. Wang, A. Farhadi, and A. Gupta, "Actions˜ transformations," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 2658–2667.

[47] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[48] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn : Surprisingly easy synthesis for instance detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1301–1310.

[49] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, "Synthesizing training data for object detection in indoor scenes," *arXiv preprint arXiv :1702.07836*, 2017.

[50] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn : Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2686–2694.

[51] M. Garon and J.-F. Lalonde, "Deep 6-dof tracking," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 11, pp. 2410–2418, 2017.

[52] Z. Wu, D. Hoang, S.-Y. Lin, Y. Xie, L. Chen, Y.-Y. Lin, Z. Wang, and W. Fan, "Mm-hand : 3d-aware multi-modal guided hand generative network for 3d hand pose synthesis," *arXiv preprint arXiv :2010.01158*, 2020.

[53] "Object detection," 2005. [Online]. Available : https://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[55] "Pytorch." [Online]. Available : https://pytorch.org/hub/pytorch_vision_resnet/

[56] Z. Dai, H. Liu, Q. Le, and M. Tan, "Coatnet : Marrying convolution and attention for all data sizes," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[57] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 3304–3311.

[58] F.-S. Chen, C.-M. Fu, and C.-L. Huang, "Hand gesture recognition using a real-time tracking method and hidden markov models," *Image and vision computing*, vol. 21, no. 8, pp. 745–758, 2003.

[59] J. Martin, V. Devin, and J. L. Crowley, "Active hand tracking," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 1998, pp. 573–578.

[60] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1. IEEE, 2005, pp. 166–173.

[61] H. Ershaed, I. Al-Alali, N. Khasawneh, and M. Fraiwan, "An arabic sign language computer interface using the xbox kinect," in *Annual Undergraduate Research Conf. on Applied Computing*, vol. 3, 2011.

[62] K. Lyons, H. Brashear, T. Westeyn, J. S. Kim, and T. Starner, "Gart : The gesture and activity recognition toolkit," in *International Conference on Human-Computer Interaction*. Springer, 2007, pp. 718–727.

[63] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti, "American sign language recognition with the kinect," in *Proceedings of the 13th international conference on multimodal interfaces*, 2011, pp. 279–286.

[64] K. Kahol, P. Tripathi, and S. Panchanathan, "Automated gesture segmentation from dance sequences," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.* IEEE, 2004, pp. 883–888.

[65] H. Kang, C. W. Lee, and K. Jung, "Recognition-based gesture spotting in video games," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1701–1714, 2004.

[66] H.-K. Lee and J.-H. Kim, "An hmm-based threshold model approach for gesture recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 10, pp. 961–973, 1999.

[67] T. J. Darrell, I. A. Essa, and A. P. Pentland, "Task-specific gesture analysis in real-time using interpolated views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1236–1242, 1996.

[68] J. Kruskall and M. Liberman, "The symmetric time warping algorithm : From continuous to discrete. time warps, string edits and macromolecules," 1983.

[69] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997, pp. 994–999.

[70] N. Stefanov, A. Galata, and R. Hubbold, "Real-time hand tracking with variable-length markov models of behaviour," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*. IEEE, 2005, pp. 73–73.

[71] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in *Proceedings IEEE ICCV workshop on recognition, analysis, and tracking of faces and gestures in real-time systems*. IEEE, 2001, pp. 82–89.

[72] R. Cutler and M. Turk, "View-based interpretation of real-time optical flow for gesture recognition," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 1998, pp. 416–421.

[73] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *Third workshop on mining temporal and sequential data*, vol. 32. Citeseer, 2004.

[74] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler, "Learning human pose estimation features with convolutional networks," *arXiv preprint arXiv :1312.7302*, 2013.

[75] S. Li, Z.-Q. Liu, and A. B. Chan, "Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 482–489.

[76] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "Modeep : A deep learning framework using motion features for human pose estimation," in *Asian conference on computer vision*. Springer, 2014, pp. 302–315.

[77] J. Huang, W. Zhou, H. Li, and W. Li, "Sign language recognition using 3d convolutional neural networks," in *2015 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2015, pp. 1–6.

[78] Y. Li, Q. Miao, K. Tian, Y. Fan, X. Xu, R. Li, and J. Song, "Large-scale gesture recognition with a fusion of rgb-d data based on saliency theory and c3d model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2956–2964, 2017.

[79] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[80] X. Chai, Z. Liu, F. Yin, Z. Liu, and X. Chen, "Two streams recurrent neural networks for large-scale continuous gesture recognition," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 31–36.

[81] T. Eleni, "Gesture recognition with a convolutional long short term memory recurrent neural network," Ph.D. dissertation, 2015.

[82] N. Nishida and H. Nakayama, "Multimodal gesture recognition using multi-stream recurrent neural network," in *Image and Video Technology*. Springer, 2015, pp. 682–694.

[83] C. K. Lee, K. K. Ng, C.-H. Chen, H. C. Lau, S. Chung, and T. Tsoi, "American sign language recognition and training method with recurrent neural network," *Expert Systems with Applications*, vol. 167, p. 114403, 2021.

[84] R. Ibanez, A. Soria, A. Teyseyre, and M. Campo, "Easy gesture recognition for kinect," *Advances in Engineering Software*, vol. 76, pp. 171–180, 2014.

[85] K. Iglesias, "3d character dummy : Characters." [Online]. Available : https://assetstore.unity.com/packages/3d/characters/humanoids/humans/3d-character-dummy-178395

[86] A. Mueen and E. Keogh, "Extracting optimal performance from dynamic time warping," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 2129–2130.

[87] Z. Li, Y. Hu, M. Salzmann, and X. Ji, "Robust rgb-based 6-dof pose estimation without real pose annotations," *arXiv preprint arXiv :2008.08391*, 2020.

[88] R. M. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1426–1446, 1989.

[89] C. Ünsalan, "A model based approach for pose estimation and rotation invariant object matching," *Pattern Recognition Letters*, vol. 28, no. 1, pp. 49–57, 2007.

[90] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn : A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv :1711.00199*, 2017.

[91] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d : Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

[92] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 699–715.

[93] A. Kendall, M. Grimes, and R. Cipolla, "Posenet : A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[94] Z. Li, G. Wang, and X. Ji, "Cdpn : Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7678–7687.

[95] E. Brachmann and C. Rother, "Learning less is more-6d camera localization via 3d surface regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4654–4662.

[96] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod : 6d pose object detector and refiner," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1941–1950.

[97] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.

[98] A. Wetzler, R. Slossberg, and R. Kimmel, "Rule of thumb : Deep derotation for improved fingertip detection," *arXiv preprint arXiv :1507.05726*, 2015.

[99] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 409–419.

[100] M. Garon, D. Laurendeau, and J.-F. Lalonde, "A framework for evaluating 6-dof object trackers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 582–597.

[101] J. Marvin, "Calibrating the handyscan 3d," Dec 2021. [Online]. Available : https://www.cati.com/blog/calibrating-handyscan-3d/

[102] pywinauto, "pywinauto documentation." [Online]. Available : https://pywinauto.readthedocs.io/en/latest/

[103] K. Lauzon, "Jivago – the highly-reflective object-oriented python web framework." [Online]. Available : https://jivago.readthedocs.io/en/stable/