

Auteur :

— Elliot VANEGUE

Encadrants :

— Hazem WANNOUS

— Jean-Philippe VANDEBORRE

Présentation du stage de fin d'étude du master IVI

Approche sémantique de segmentation et de recherche interactive par le contenu issu d'une caméra de profondeur



Mars 2016 à Aout 2016

Remerciement

Je remercie d'abord M. Daoudi pour m'avoir pris dans son équipe.

Je remercie M Vandeborre et M Wannous pour leur soutien, leur conseil et leur suivi durant ce stage.

Je remercie l'ensemble des doctorants de l'équipe pour leur accueil et pour la bonne ambiance durant ce stage.

Je remercie ensuite M Aubert pour son soutien et son investissement durant cette année dans la spécialité IVI.

Enfin je remercie l'ensemble de ma famille pour leur soutien durant mes études.

Résumé

L'objectif de ce stage est de réaliser une méthode de recherche d'information et de reconnaissance de forme 3D. Le travail a été divisé en deux phases. La première concerne des objets non rigides comme les parties du corp et la seconde sur des objets rigides comme des meubles. Pour cela, je me sers de la caméra 3D Kinect v2 pour récupérer les informations du monde réel sous forme de nuage de points. Grâce à ces données, nous pouvons détecter des objets afin de les matérialiser dans un environnement virtuel. Ce projet vise à faciliter les outils de designer afin de leur permettre d'importer des objets du monde réel dans leur projet, pour éviter de remodeliser des objets existants.

La première parties du projet consiste à réaliser une application qui récupère le corps de l'utilisateur dans le but de pouvoir en modifier certaines parties. La seconde partie est une application qui permet d'apparailler des objets 3D rigide dans une scène 3D. Les deux applications passent par une étape de segmentation, puis de détection et de reconnaissance d'objet. Les parties les plus complexes à réaliser à l'heure actuelle sont prises en charge par l'utilisateur au travers d'une interface que j'ai construit pour ces deux projets.

Pour ces projets, j'ai utilisé les outils fournis dans le SDK de la Kinect, ainsi qu'une bibliothèque pour le traitement de nuage de points, appelée PCL. Mes travaux se sont surtout concentrés sur la reconnaissance d'objet, notamment dans la seconde application. Pour cette étape, j'ai utilisé un descripteur appelé FPFH dans un bag of words, pour calculer un descripteur uniformisé et invariant en translation, rotation et changement d'échelle. Afin de créer une base de connaissances, j'ai utilisé un algorithme d'apprentissage automatique appelé SVM.

Table des matières

1	Introduction	5
1.1	Contexte	5
1.2	Objectif du stage	5
1.3	Problématique	6
1.4	Organisation	6
2	Etude des approches existantes	7
2.1	Segmentation 3D	7
2.1.1	Segmentation d'un environnement intérieur	8
2.1.2	Segmentation des parties du corps humain	8
2.2	Reconnaissance d'objets	9
2.2.1	Calcul de descripteur	10
2.2.2	Apprentissage automatique	11
2.2.3	L'approche sac de mots (bag of word)	12
2.3	Positionnement de modèle	12
2.4	Interaction utilisateur	13
3	Appariement de modèles 3D complet et non-complet	14
3.1	Objectif	14
3.2	Délimitation du corps humain et minimisation de la quantité de donnée	14
3.3	Calcul de la distance géodésique	15
3.4	Segmentation du corps humain	17
3.5	SDK de la Kinect	18
3.6	Positionnement	19
3.7	Travaux futurs et applications	21
4	Reconstruction d'un environnement intérieur	21
4.1	Objectif	21
4.2	Création de la base d'apprentissage	22
4.3	Interface utilisateur	23
4.4	Reconnaissance d'objet	23
4.5	Travaux futurs et applications	24
5	Conclusion	24

1 Introduction

1.1 Contexte

Durant mon master IVI¹, nous avons l'occasion de réaliser un stage de fin d'étude. J'ai choisi de réaliser ce stage dans l'équipe 3D-SAM du laboratoire CRISTAL spécialisé dans l'acquisition et le traitement d'image 3D à partir de capteur 3D de type Microsoft Kinect. Leurs principaux travaux portent sur l'analyse de forme d'objet 3D et la modélisation des variations des formes dans des vidéos 3D. Réaliser mon stage dans un laboratoire était une priorité, car mes projets d'avenir ne sont pas encore parfaitement planifiés et ce stage me permet de me décider sur l'environnement de travail qui me convient le mieux. Durant mes deux derniers stages en IUT et en licence 3, j'ai pu observer les atouts et les contraintes de chaque environnement, cependant le stage que j'ai effectué durant mon année en IUT relevé plus de l'ingénierie que de la recherche. Durant ce stage de deuxième année d'étude, je n'ai pas eu à effectuer un état de l'art, ni à tester des méthodes proposées par d'autres laboratoires de recherche.

Ces deux années de master nous ont beaucoup appris sur le traitement d'image et la reconnaissance de forme 2D. Ces sujets m'ont particulièrement intéressé et je souhaite, durant ce stage, approfondir ces notions sur des types de données plus complexes comme sur des images 3D. C'est pourquoi je réalise mon stage dans l'équipe 3D-SAM avec qui j'avais déjà travaillé sur mon projet de fin d'étude. Cette équipe dirigée par M. Daoudi comprend cinq membres permanents, un post-doctorant et quatre doctorants. Mes encadrants durant ce stage sont M. Vandeborre et M. Wannous. Ce stage se déroule dans le cadre du projet CrABEx qui concerne la production et l'édition de produits 3D pour des applications de loisir. L'enjeu de ce projet est d'aider les designer 3D dans la création et l'édition de ressources graphiques, en suggérant des éléments appropriés durant leur processus de création ou en générant automatiquement de nouvelles ressources à partir d'éléments existants.

1.2 Objectif du stage

Comme nous pouvons le constater depuis quelques années, les approches 3D interactives sont de plus en plus présentes dans nos vies, que ce soit dans le domaine de la médecine avec l'utilisation de simulateur pour apprendre à réaliser des opérations complexes, dans le loisir avec les jeux vidéo dont le revenu mondial en 2015 est de plus de 90 milliards de dollars ou encore dans l'industrie pour visualiser des produits. L'engouement pour cette technologie requiert des outils de plus en plus efficaces et rapides permettant à des personnes de profession plutôt artistique de laisser place à leur imagination sans s'inquiéter de l'aspect technique.

Le fait de pouvoir modéliser dans un monde virtuel des objets du monde réel, et de pouvoir modifier ces objets facilement permettrait au designer de se soustraire de certaines tâches. On peut par exemple modéliser une scène ou une personne facilement grâce à une caméra Kinect, puis effectuer des modifications sur le modèle résultant. Il serait intéressant de récupérer le modèle 3D d'une personne et de modifier quelques unes des parties de son corps avec des membres improbables comme un bras de robot. Le principe est le même dans une pièce, si nous souhaitons remplacer des meubles d'une pièce intérieure par d'autres plus étonnants. L'intérêt de cela est d'éviter à un designer de devoir remodeler une personne ou une pièce existante qui serait déjà idéale pour ce que l'on souhaite réaliser.

Ce genre de technologie existe déjà pour modéliser des visages ou des gestes dans des jeux vidéo ou des films d'animation. Il y a par exemple des capteurs optiques basés sur des caméras

1. Le master Image Vision Interaction est une spécialité du master informatique de l'université de Lille 1

infrarouges avec des marqueurs réfléchissants. La Kinect fait également partie des outils de capture de mouvement, mais elle reste très peu utilisée dans le milieu professionnel.

1.3 Problématique

Les caméras 3D nous permettent d'obtenir un nuage de points de l'environnement qu'elles enregistrent. Cependant, ce type de caméra est peu précis et celles-ci sont souvent bruitées et comportent des valeurs qui n'existent pas dans le monde réel. La première difficulté lors de ce projet est de réussir à filtrer les données, de sorte qu'il ne reste que les données réellement présentes. L'ensemble des données ne nous intéresse pas forcément. Nous devons donc passer par une phase de segmentation des données afin de détecter les objets présents dans une scène 3D. En effet, si nous travaillons sur le corps humain nous n'avons pas besoin de l'environnement qu'il y a autour, ce surplus de données nous dérange lors de nos traitements. Il faut donc trouver une solution permettant de segmenter les données que nous recevons, afin de ne garder que certains objets.

Une fois que les données ont été triées, nous avons besoin de reconnaître les objets présents dans notre scène. Par exemple, dans le cas du corps humain, si nous souhaitons modifier une partie du corps comme la main, il faut d'abord savoir quel partie du corps représente la main. La reconnaissance d'objet dépend fortement du type de données et nécessite généralement d'avoir une base d'apprentissage assez volumineuse. Lors de la modification d'une partie de la scène, il est nécessaire de connaître la position de l'objet remplacé, afin de pouvoir placer le nouvel objet au même endroit et dans le même sens. Cette partie est très délicate surtout pour des membres humains, car si le repositionnement n'est pas parfait et qu'il y a un écart entre deux membres, la scène perd toute sa crédibilité et son réalisme.

Les données que nous récupérons via la Kinect sont considérés comme des données 2.5D. En effet, dans ce type de données nous récupérons des informations de profondeur, mais nous ne pouvons récupérer les informations occultées comme le dos de la personne qui est face à la caméra. La solution que nous souhaitons mettre en place doit remplacer des données en 2.5D par des données 3D de synthèse. Nous devons donc trouver une solution permettant de mettre en correspondance les caractéristiques de ces deux types de données.

Nous avons donc trois problématiques majeures dans ce projet :

- Comment filtrer les données de manière à ne traiter que celles qui nous intéressent ?
- Comment reconnaître automatiquement un objet dans une scène 3D ?
- Comment connaître la position et l'orientation exactes des objets présents dans notre scène ?
- Comment mettre en correspondance des informations 2.5D et 3D ?

1.4 Organisation

Pour répondre aux questions précédentes, j'ai travaillé sur deux applications mettant en avant des problématiques similaires, mais avec deux façons différentes d'aborder ces problèmes. J'ai travaillé dans un premier temps sur une application qui se focalise sur le corps humain. Cette application a pour but d'identifier les membres du corps, afin que l'utilisateur puisse les modifier grâce à des modèles 3D existants. Les seules interactions de l'utilisateur sont donc de cliquer sur le membre à modifier et de sélectionner un modèle parmi ceux que l'application lui proposera afin de l'apparailler au reste du corps humain. La seconde application se concentre sur un environnement intérieur comportant plusieurs objets. Ici le but est de segmenter la pièce pour reconnaître les objets présents afin de les ajouter dans une scène. Encore une fois, l'utilisateur doit

sélectionner un meuble et le modèle 3D qui lui convient dans une liste proposée par l'application.

Pour ce projet, j'utilise la caméra Microsoft Kinect v2, qui est l'une des caméras les plus utilisées dans la littérature. Le SDK fourni avec cet outil comporte une segmentation du corps humain, la position du squelette de l'utilisateur et l'algorithme Kinect fusion qui permet de construire un modèle 3D à partir du nuage de points fourni par la caméra. Pour réaliser les deux applications, j'ai utilisé un ensemble de bibliothèques telles que la bibliothèque graphique de Microsoft pour la réalisation de l'interface, la bibliothèque « Point Cloud Librairie » [1] (PCL) pour les calculs sur les nuages de points et opencv pour travailler sur les images couleurs et les images de profondeur fournies par la Kinect.

J'ai dans un premier temps réalisé un état de l'art des solutions existantes sur les problématiques de segmentation et de reconnaissance de forme à partir de nuage de points ou d'images de profondeur. Puis je vais décrire les solutions testées et appliquées aux deux applications que j'ai développées durant ce projet.

2 Etude des approches existantes

2.1 Segmentation 3D

La première étape lors de ce projet va être de segmenter les images que nous recevons de la caméra. Les informations contenues dans une image 3D sont nombreuses et nous devons déterminer les éléments importants pour nos traitements. Dans notre scène, nous avons besoin des objets proches ou du corps de la personne en face de la caméra, mais l'environnement autour des ces objets clés n'est pas important et doit être supprimé pour gagner du temps lors de nos traitements. Une seconde segmentation est nécessaire pour le traitement du corps humain. Pour cette étape du projet, nous devons segmenter le corps en plusieurs parties pour pouvoir, par la suite, les reconnaître. Si cette seconde segmentation n'est pas réalisée, il ne nous sera pas possible de reconnaître les mains ou encore la tête si nous ne savons pas délimiter les parties du corps.



FIGURE 1 – Résultat attendu pour la segmentation d'un environnement intérieur³et du corps humain

3. source : <http://kos.informatik.uni-osnabrueck.de/icar2013/>

2.1.1 Segmentation d'un environnement intérieur

De nombreux travaux ont été réalisés dans la segmentation d'image 2D avant que les caméras 3D ne soient ouvertes au grand public. Les premières méthodes de segmentation reposaient sur la détection de contour comme pour la méthode de P. Arbelaez et al[2]. Leur méthode repose sur le détecteur de contour gPb qui est composé d'un seuillage sur la luminance et sur la couleur, et d'une détection de texture. La fermeture des contours se fait ensuite en utilisant les superpixels. D'autres méthodes 2D utilisent un simple seuillage en utilisant par exemple la méthode de N. Otsu[3] pour binariser l'image et ainsi la segmenter.

Avec l'arrivée des caméras 3D, de nombreuses recherches ont été effectuées sur la segmentation d'image à partir des informations extraites de ce type de caméra. S.A.A Shah et al[4] utilisent les informations de l'image de profondeur afin de calculer un vecteur sur chaque pixel. Ce vecteur s'obtient en calculant la divergence entre les pixels. En appliquant un seuillage sur les vecteurs, ils obtiennent une segmentation de l'environnement qui leur permet de détecter des objets dans une pièce. Cette méthode est efficace lorsque l'objet que l'on cherche à détecter est proche de la caméra. Il est possible à partir de l'image de profondeur, de créer un nuage de points, ce qui permet d'obtenir les coordonnées 3D des points présents dans l'image de profondeur. Les informations qu'il est possible d'extraire d'un nuage de points sont différentes, et des méthodes de segmentation se sont développées autour de ces informations.

T. Rabbani et al[5] utilisent les informations obtenues dans un nuage de points afin de calculer les normales de chaque point. Ils segmentent ensuite l'image en comparant les normales et en appliquant un seuillage sur cette comparaison. Si les angles formés par les normales du point courant et de ces voisins sont inférieurs au seuil alors les points appartiennent à la même région.

Nous pouvons voir que les méthodes citées précédemment sont efficace pour segmenter une scène comportant des objets, mais elles ne sont pas applicables à un corps humain. Le principal défaut de ces méthodes pour le corps humain est que celui-ci est trop lisse. La différence entre les normales ou entre les vecteurs de pixel n'est pas assez important et celle-ci est trop instable pour que cela s'adapte sur le corps humain qui peut adopter de nombreuses postures.

2.1.2 Segmentation des parties du corps humain

La segmentation du corps humain est un sujet très complexe, car contrairement aux objets, celui-ci bouge et adopte des postures différentes. La méthode la plus souvent utilisée pour résoudre cette problématique est de déterminer la posture de l'utilisateur, puis de cette posture, déterminer les différentes parties du corps à partir de la labellisation qui doit être réalisée sur la base de connaissance. Ces méthodes nécessitent d'avoir une base de connaissance contenant de nombreuses postures qui doivent être segmentées et labellisées avec les différentes parties du corps. J. Shotton et al[6] ont créé une base d'apprentissage en calculant un descripteur et ils utilisent une technique d'apprentissage automatique appelée forêt d'arbres décisionnels[7]. Le descripteur de J. Shotton et al[6] utilise les informations de l'image de profondeur pour déterminer quelle posture a l'utilisateur. Ils utilisent une caractéristique reposant sur la valeur de deux pixels, un pixel x et un pixel dont l'offset par rapport au pixel x a été défini. Lorsque l'utilisateur bouge, le descripteur utilisé précédemment est recalculé sur l'image courante et le résultat est comparé au posture de la base d'apprentissage.

La méthode de J. Shotton et al[6] est efficace pour de nombreuses parties du corps, mais reste instable sur les parties de la main comme le poignet et son centre. Mais cette méthode permet tout de même de déterminer approximativement les parties du corps occultées.



FIGURE 2 – Deux exemples de caractéristique du descripteur de J. Shotton et al[6]. La croix jaune correspond au pixel à classifier et le cercle rouge correspond au pixel décalé.

Comme pour la méthode précédente B. Yoo et al[8] utilisent les images de profondeur pour déterminer la posture de l'utilisateur. Cependant, leur descripteur repose sur des caractéristiques plus complexes comme l'élongation 3D de la forme du corps, le centre de gravité, la rectangularité de la forme ou encore la dissymétrie. Grâce à ces caractéristiques, ils forment un descripteur qu'il passe dans leur propre outil d'apprentissage automatique appelé « Randomized Decision Bush ».

Y. Liu et al[9] ont développé un autre descripteur spécifique à la reconnaissance de la posture du corps humain appelé « Geodesic Invariant Feature » (GIF). Ce descripteur se base sur le calcul de la distance géodésique. La distance géodésique est la distance entre deux points d'un modèle 3D en ne passant que par les arêtes de ce modèle. Etant donné que ce n'est pas un modèle 3D que fournit la Kinect, mais un nuage de points, les auteurs forment un maillage en créant n arêtes avec les n points les plus proches pour un point donné. Cette distance géodésique permet de connaître l'orientation des points. Cette orientation est appliquée sur d'autres caractéristiques sensibles à la rotation, ce qui permet à ces caractéristiques d'être invariantes en rotation.

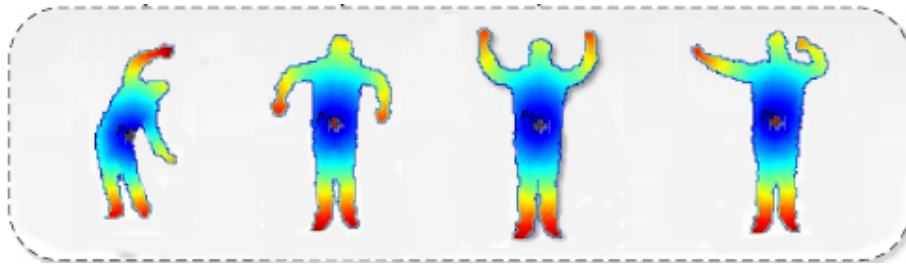


FIGURE 3 – Distance géodésique d'un corps humain

2.2 Reconnaissance d'objets

La reconnaissance d'objets est un sujet assez vaste dans le monde de l'imagerie et il existe de nombreuses méthodes dans le domaine, que ce soit pour des images 2D ou 3D. Dans le cas d'image 3D, l'approche de reconnaissance d'objet commence par le calcul de descripteur, ce qui correspond à un ensemble de caractéristiques représentant un objet spécifique. Ce descripteur va ensuite être utilisé dans un classifieur afin de réaliser une base comportant les caractéristiques de l'ensemble des objets que nous souhaitons reconnaître par la suite.

2.2.1 Calcul de descripteur

Le nombre de descripteurs qui existent dans le domaine de l'image 3D est assez important, c'est pourquoi pour ce rapport, nous allons nous contenter de décrire seulement les plus utilisés. Le descripteur D2[10] est un des outils de comparaison de forme 3D les plus simple à réaliser. Il se repose sur le calcul de la distance euclidienne entre chaque point du modèle 3D. L'ensemble de ces distances permet de créer un histogramme 1D et de comparer ces histogrammes afin de reconnaître un objet. Ce descripteur fournit de bons résultats lorsque les objets à reconnaître sont très différents.

Le descripteur PFH[11] (Point Feature Histograms) est un outil permettant de calculer la courbure moyenne d'un voisinage de points en utilisant un histogramme multi-dimensionnel. Le calcul de la courbure et le fait que ce soit une généralisation permettent d'être invariant en translation, en rotation et en densité de points, et permet également d'être moins sensible au bruit présent dans le nuage de points. Le voisinage dépend de la distance des points avec le point central et il ne peut excéder un certain nombre de voisin s (voir Fig. 4).

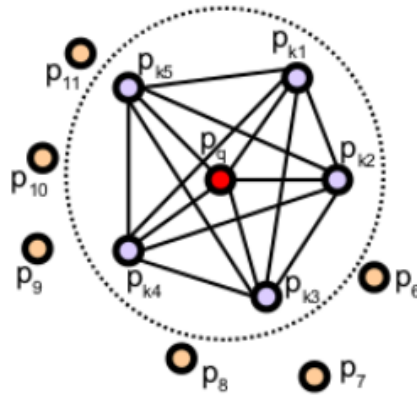


FIGURE 4 – Exemple de voisinage pris en compte dans le calcul de la courbure du descripteur PFH

Le descripteur PFH calculé en un point correspond à la relation que ce point a avec l'ensemble des points de son voisinage. Cette relation est la différence des normales entre deux points. Chaque classe de l'histogramme est composée de l'ensemble des points du voisinage dont la relation avec le point central est similaire. Une version améliorée du descripteur a été proposée par R. B. Rusu et al[12] appelé FPFH (Fast Point Feature Histograms). Cette version est plus rapide, car elle calcule un descripteur PFH simplifié, puis elle construit de nouveaux histogrammes à partir des histogrammes simplifiés précédents (voir Fig. 5).

D. G. Lowe[13] a créé un descripteur appelé « scale-invariant feature transform » (SIFT) qui crée des points clés dans une image 2D et calcule des descripteurs sur ces points. Ce descripteur permet entre autre de détecter des points clés similaires dans deux images différentes d'une même scène, tant que les angles de vue sont suffisamment petits. Les points clés de ce descripteur sont des zones circulaires positionnées sur les extremas dans l'espace des échelles, le facteur d'échelle étant proportionnel à la taille de la zone d'intérêt. Une fois que l'on a trouvé la position des points clés, il faut déterminer leur orientation en calculant le gradient dans le voisinage du point clé. Grâce aux orientations des voisins des points clés, il est possible de créer un histogramme des orientations. Ainsi l'orientation du point clé correspond aux pics les plus importants de l'histogramme. La construction de ces points clés permet à ceux-ci d'être

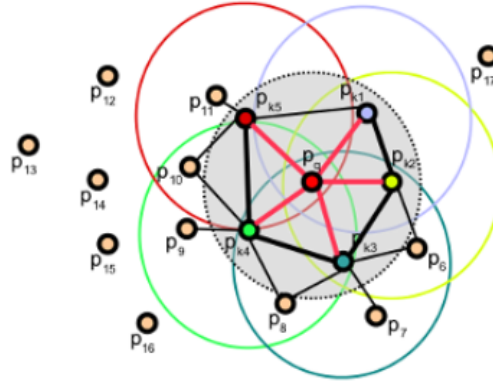


FIGURE 5 – Exemple de voisinage pris en compte dans le calcul de la courbure du descripteur FPFH

invariants aux changements d'échelle et aux rotations.

2.2.2 Apprentissage automatique

L'apprentissage automatique est un outil permettant à une machine de prendre des décisions rapidement. Pour fonctionner, cet outil a besoin de données déjà traitées sur un domaine précis. Il existe de nombreuses techniques d'apprentissage automatique. Lors de ce projet, je me suis intéressé à deux techniques en particulier qui sont parmi les plus utilisées dans le domaine de l'image : la « forêt d'arbres décisionnels » et les « machines à vecteurs de support » (SVM).

Le principe de la forêt d'arbres décisionnels[7] est de créer un ensemble d'arbres. Dans le cas de la méthode de J. Shotton et al[6], chaque arbre correspond à une posture. Les noeuds des arbres correspondent aux caractéristiques calculées. L'algorithme teste des noeuds lui permettant ainsi de tracer un chemin vers une feuille qui donne un résultat. L'arbre dont la feuille nous donne le résultat le plus proche de la valeur recherchée nous fournit la solution à notre problème.

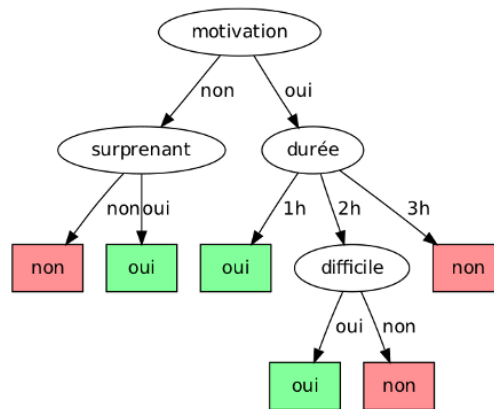


FIGURE 6 – Exemple d'arbre de décision ⁴

Les machines à vecteurs de support[14], quant à eux, sont une classe d'algorithmes d'apprentissages supervisés permettant la résolution de problèmes de discrimination non linéaire. Le

4. source : <https://scaron.info/doc/intro-arbres-decision/>

principe des SVM est d'apprendre un séparateur dans le but de classer les données à partir d'une base d'apprentissage. Dans le cas d'un problème linéaire, les SVM recherchent l'hyperplan le plus approprié à la séparation des données dans l'espace vectoriel, en minimisant les pertes de points dans la base d'apprentissage. Lorsque le problème est non linéaire, les SVM changent l'espace dans lequel les points sont projetés à l'aide d'une fonction noyau.

2.2.3 L'approche sac de mots (bag of word)

Le sac de mots est une représentation dont la première utilisation était de décrire un document texte en fonction d'un dictionnaire. Le dictionnaire est un ensemble de mots capable de décrire tous les textes. Le principe est de compter le nombre d'occurrence d'un mot dans un texte, et ce pour chaque mot du dictionnaire. Cette représentation a ensuite été utilisée dans le domaine de la vision par ordinateur appelé « bag of visual world » [15] en remplaçant le dictionnaire de mot par un dictionnaire de caractéristiques. Grâce à cette représentation, nous obtenons un vecteur de la taille du dictionnaire, composé du nombre d'occurrence de chaque caractéristique. L'intérêt de cette représentation est d'uniformiser la structure des données afin d'avoir un vecteur de même taille pour toutes les images, mais cela nécessite une première phase de création du dictionnaire avec l'ensemble des caractéristiques des images traitées. Nous obtenons ainsi un nouveau descripteur sous la forme d'un histogramme dont la taille est égale au nombre de caractéristique différente de chaque classe.

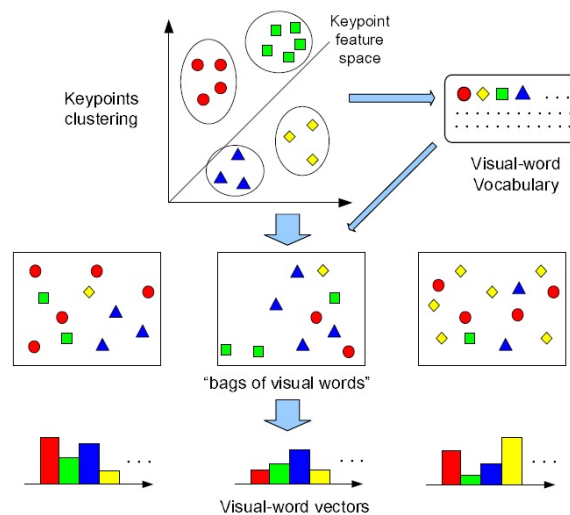


FIGURE 7 – Schéma du fonctionnement de l'algorithme du bag of visual words ⁵

2.3 Positionnement de modèle

Afin de réaliser la correspondance entre deux modèles P. J. Besl et al [16] développe l'algorithme « iterative closest point » (ICP). Cet algorithme recherche l'ensemble des translations et rotations nécessaires à la mise en correspondance de deux modèles similaires. Pour cela, celui-ci fonctionne en quatre étapes :

- Associer les points grâce aux critères du plus proche voisin. Pour cela, il suffit de calculer la distance euclidienne d'un point avec tous les autres points qui font partis du balayage que nous voulons comparer et de prendre la distance la plus petite.
- Estimer la transformation des points grâce à une fonction d'erreur quadratique moyenne, permettant ainsi de trouver la meilleure transformation possible.

5. source : <http://www.ifp.illinois.edu/~yuhuang/sceneclassification.html>

- Effectuer la transformation du nuage de points ayant la plus petite erreur.
- Itérer jusqu'à ce qu'on ait atteint la condition de fin fixée par l'utilisateur.

Dans la bibliothèque PCL[1], S. Ushakov implémente une classe basée sur le moment d'inertie⁶ permettant de déterminer les axes principaux d'un nuage de points. Pour cela, cette classe calcule la matrice de covariance du nuage de points et en extrait les vecteurs propres. Le plus grand vecteur propre est considéré comme étant l'axe X et le plus petit devient l'axe Z, ce qui permet d'obtenir un système de coordonnées dans un référentiel local à l'objet.

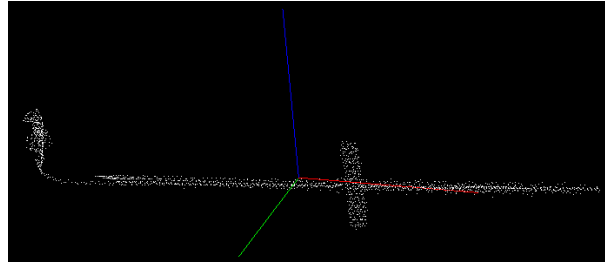


FIGURE 8 – Résultat du calcul des axes sur un nuage de points grâce à la méthode utilisant le moment d'inertie

2.4 Interaction utilisateur

T. Shao et al[17] proposent, en plus d'avoir une méthode de segmentation automatique, d'améliorer leur segmentation en impliquant l'utilisateur dans le processus. Ainsi lorsque la segmentation d'une pièce comporte des erreurs, l'utilisateur est amené à rectifier les erreurs de la machine au travers d'une interface. Pour cela les auteurs proposent à l'utilisateur de dessiner sur l'objet dont le label est erroné avec la couleur du label qui correspond, ainsi l'application change la couleur et le label de l'objet sélectionné.

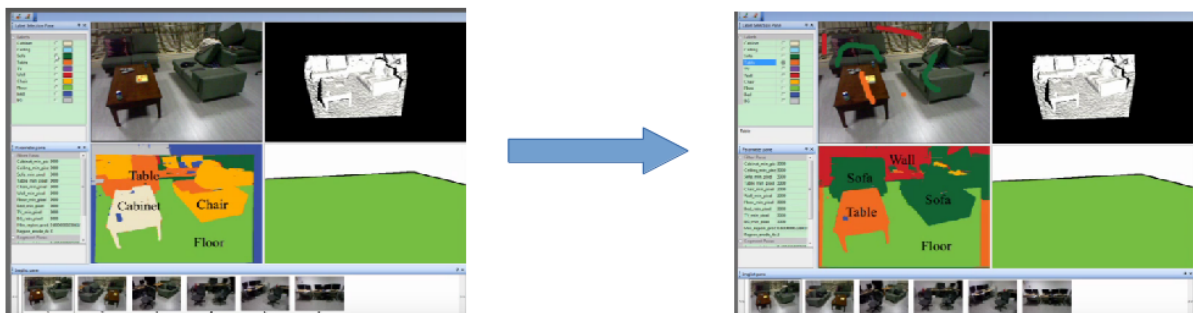


FIGURE 9 – Résultat de la reconnaissance interactive de T. Shao et al[17]

6. http://pointclouds.org/documentation/tutorials/moment_of_inertia.php#moment-of-inertia

Sur le même principe, J. Xiao et al[18] ont créé un système d'interaction afin de faciliter la création de leur base d'apprentissage. Pour cela, lorsque l'utilisateur clique sur l'image, il place un marqueur. Cela permet, en plaçant plusieurs marqueurs, de délimiter un objet. Lorsque le premier marqueur a la même position que le dernier, le contour est terminé, et l'utilisateur est invité à donner un label à l'objet qu'il vient de délimiter.

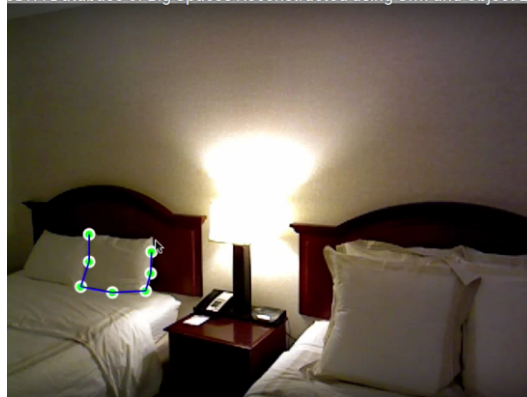


FIGURE 10 – Résultat de la sélection interactive de J. Xiao et al[18]

3 Appariement de modèles 3D complet et non-complet

3.1 Objectif

Cette première application a plusieurs objectifs. Dans un premier temps, nous souhaitons réaliser une segmentation sans utiliser les outils fournis par la Kinect, dans le but d'obtenir une méthode plus précise ou, tout du moins, permettant d'améliorer une partie du procédé de segmentation. Comme nous l'avons vu dans l'état de l'art, les méthodes utilisées par la Kinect ont évolué et sont devenues plus stables et plus précises. Le second objectif est de remplacer un nuage de points représentant un membre par un modèle 3D de ce même membre mais ayant une forme différente. Cette première partie du stage me permet surtout d'aborder des concepts qui me serviront dans la seconde application qui est le but premier de ce stage.

3.2 Délimitation du corps humain et minimisation de la quantité de donnée

Durant cette première phase, nous travaillons sur un nuage de points et non sur les informations de l'image de profondeur. Comme nous l'avons dit précédemment, l'ensemble des informations n'est pas pertinente pour les traitements que nous souhaitons réaliser. La première étape dans la réalisation de cette application est de supprimer l'ensemble des informations qui ne se rapportent pas au corps humain. La première solution que nous développons consiste à utiliser deux seuils. Les points qui sont plus éloignés que le premier seuil, ainsi que les points en dessous du second seuil sont supprimés. Ces seuils sont les distances, en mètre, dans lequel l'utilisateur doit se trouver. Cette méthode est utilisée dans l'application ReconstructMe⁷ qui est une application permettant de construire un modèle 3D complet à partir de données fournies par plusieurs images provenant d'une caméra 3D. Comme pour cette application, nous avons décidé de laisser la possibilité à l'utilisateur de changer les seuils en fonction de son besoin.

Nous pouvons voir sur la Fig. 11 que le seuil permet effectivement de supprimer beaucoup d'informations correspondant à l'environnement, mais qu'il reste beaucoup de bruit dû à la qua-

7. <http://reconstructme.net>



FIGURE 11 – Résultat d’une segmentation par seuillage

lité de l’acquisition de la caméra ainsi que des objets qui se trouvent à la même distance que l’utilisateur. La librairie PCL[1] nous fournit beaucoup d’outils pour ce genre de problématique. Il y a une classe appelée *StatisticalOutlierRemoval* qui permet de supprimer les points supposés être du bruit. Pour cela, cette classe calcule la distance moyenne d’un point avec son voisinage et si cette moyenne est trop élevée, elle supprime le point en question. Cependant, l’utilisation de cette classe ne suffit pas à supprimer les objets à la même distance que l’utilisateur, car la densité de points de ces objets leur permet d’avoir suffisamment de voisins proches pour avoir une moyenne de distances très petites. Pour pallier à ce problème, nous avons ajouté un système de sélection permettant à l’utilisateur de sélectionner les données ne faisant pas partie du corps humain.

Pour la suite des traitements que nous souhaitons réaliser, il est préférable d’avoir un minimum d’informations et de ne garder que ce qui est pertinent. Le corps humain que nous avons réussi à délimiter comporte encore beaucoup trop de données. Le nombre de points fourni par la Kinect est très important et très concentré, il est possible de supprimer des points qui sont trop proches les uns des autres. Là encore PCL[1] peut nous aider avec la classe *VoxelGrid*. Cette classe crée une grille de voxels sur le nuage de points, dont la taille est définie par l’utilisateur. L’ensemble des points, à l’intérieur d’un voxel, sont approximés en un point qui correspond au centroïde du voxel. Grâce à l’ensemble de ces traitements, nous ne gardons que l’information essentielle à nos traitements.

3.3 Calcul de la distance géodésique

Notre première idée pour segmenter le corps humain est d’utiliser la distance géodésique. Y. Liu et al[9] montre que la distance géodésique pour une même personne, quelque soit sa posture, est toujours la même pour les points de ses membres. Le seul problème est que plusieurs membres ont la même distance géodésique, on ne peut donc pas associer un membre directement à une distance. Cependant, il est possible de seuiller le corps et de calculer des descripteurs, afin de déterminer quel nuage de points correspond à quel membre.

Avant de calculer la distance géodésique du corps humain, nous avons besoin de créer un maillage sur le nuage de points. Pour cela, j’ai repris le principe utilisé pour le descripteur FPFH[12] pour la sélection des voisins. Dans un premier temps, je calcule la distance euclidienne de chacun des points du nuage de points avec tous les autres. Pour la création du voisinage, nous considérons non seulement le nombre maximum de points à prendre en compte, mais aussi la distance maximum d’un point avec les autres. Nous avons eu de nombreux problèmes

lorsque nous n'avions pas mis la seconde condition, car lorsque la main de l'utilisateur était trop proche de sa jambe, certains points de la main avaient des voisins dans la jambe. Cette partie de l'algorithme est la plus délicate, car nous ne pourrions pas avoir un aussi bon maillage que sur un modèle 3D et la distance géodésique repose sur la qualité de celui-ci. Pour l'améliorer, nous avons créé deux paramètres que l'utilisateur peut modifier. Le premier paramètre est le nombre de voisins d'un point et le second est la distance euclidienne maximale entre les voisins.

A cette étape de l'algorithme, nous avons donc un maillage et les distances euclidiennes de chaque point avec son voisinage. Pour calculer la distance géodésique du centroïde du nuage de points avec chaque point, nous allons utiliser l'algorithme de Dijkstra[19]. Il faut donc trouver le plus court chemin du centroïde jusqu'au point dont on veut calculer la distance géodésique, en passant par le maillage que nous avons construit précédemment. Les valeurs prises en compte entre chaque point dans l'algorithme de Dijkstra seront les distances euclidiennes entre les points. A chaque fois qu'un point est ajouté au chemin emprunté par l'algorithme il faut additionner la distance euclidienne entre ce point et le précédent, pour obtenir un résultat final correspondant à la distance géodésique entre le centroïde et le point dont on cherche la distance.

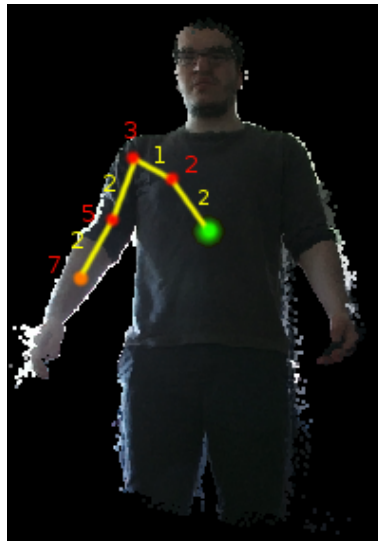


FIGURE 12 – Exemple de chemin parcouru par l'algorithme de calcul de la distance géodésique pour un point. Le point vert correspond au centre de gravité du corps, les points rouges sont les noeuds du chemin et le point orange est le point d'arrivée. Les chiffres jaunes sont les distances pour aller d'un noeud à l'autre, et les chiffres rouges sont la distance géodésique au noeud correspondant.

Sur les images de la Fig. 13, on peut voir que la distance géodésique n'est pas aussi précise qu'on le souhaiterait. Dans la première image, on voit que la distance géodésique n'est pas la même sur le bras gauche et sur le bras droit. Cette imprécision vient de la qualité du maillage. Certains points ne passent pas par le corps et traversent dans le vide du ventre au bras, ce qui réduit la distance géodésique au niveau des mains. Dans la seconde image, on voit que la position du centroïde est très importante. Celle-ci change en fonction de ce que l'on voit du corps humain, ce qui modifie également la distance géodésique. Un simple seuillage n'est donc pas suffisant pour segmenter le corps humain, car de trop nombreux paramètres interviennent dans le calcul de la distance géodésique, y compris la physiologie de l'utilisateur.

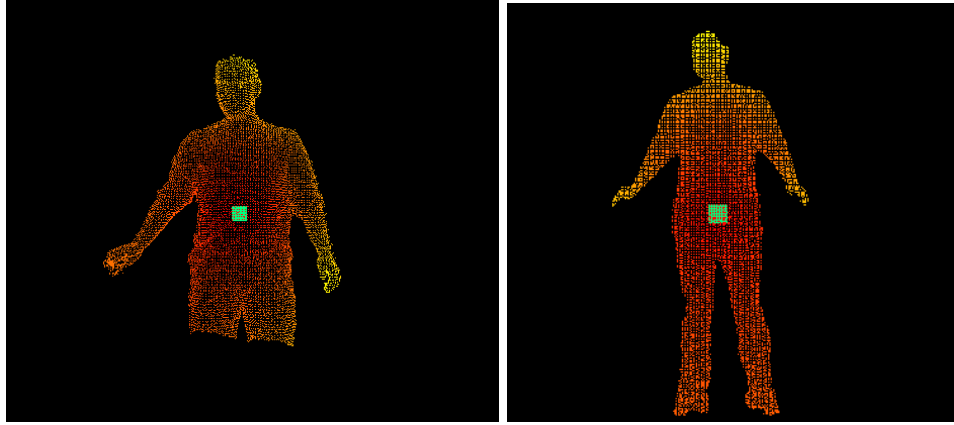


FIGURE 13 – Résultat du calcul de la distance géodésique sur l’ensemble du nuage de points. Le point vert correspond au centroïde du nuage de points. Plus la couleur des points est proche du rouge, plus les points sont proches du centroïde.

3.4 Segmentation du corps humain

Malgré un léger manque de précision de la part du calcul de la distance géodésique sur le corps humain, nous avons testé une méthode de segmentation qui ne prend pas en compte cette imprécision. Le principe de cette segmentation est de détecter le membre le plus éloigné, puis de le supprimer dans la recherche des autres membres. Pour cela, notre algorithme recherche le point dont la distance géodésique est la plus grande et forme une zone autour de ce point. Cette zone a un rayon prédéfini qui sera le même pour chaque membre. Les points à l’intérieur de cette zone seront enregistrés et sont considérés comme faisant partie d’un même membre. Ils seront ensuite supprimés du nuage de points du corps humain, afin de ne pas être pris en compte dans les étapes suivantes.

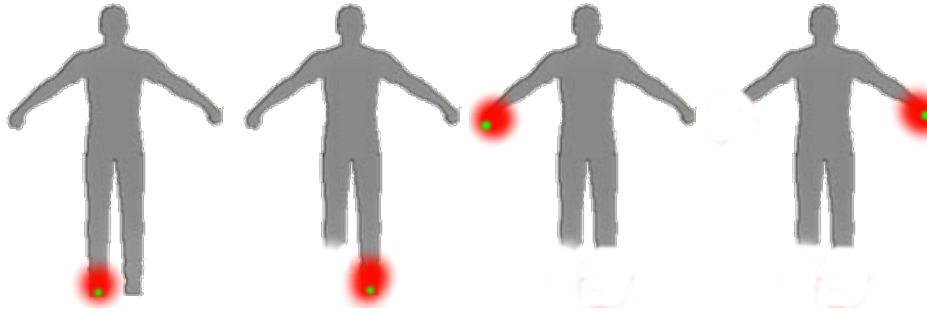


FIGURE 14 – Premières étapes de l’algorithme de segmentation du corps humain. Le point vert correspond au point le plus éloigné et la zone rouge correspond à l’ensemble des points considérés comme faisant partie du membre.

Y. Liu et al[9] proposent une amélioration de leur méthode basée sur le superpixel SLIC[20]. Le but de l’utilisation du superpixel SLIC est de diminuer le nombre de points pris en compte lors des calculs du plus court chemin de l’algorithme de Dijkstra afin de diminuer le temps de calcul. Cela permet également d’améliorer la segmentation du corps humain. Le seul paramètre de l’algorithme SLIC est le nombre de classe à retrouver dans l’image. Cet algorithme s’effectue en général dans l’espace colorimétrique LAB. La méthode de Y. Liu et al[9] quant à elle, utilise les coordonnées x , y et z . Après avoir testé l’utilisation des superpixels, nous remarquons que

nous obtenons plus rapidement le résultat, mais que celui-ci n'est pas meilleur que celui que nous obtenions précédemment.

Cette méthode est efficace pour reconnaître les extrémités du corps humain comme les mains, les pieds et la tête. Le reste des parties du corps est moins précis, notamment au niveau des épaules où le point de la zone est très instable et peut se retrouver au niveau du torse. De plus, la taille des zones dépend de la physionomie de la personne devant la Kinect.

3.5 SDK de la Kinect

Etant donné que nos résultats pour la segmentation du corps ne sont pas suffisamment précis pour la suite de nos traitements et par faute de temps, nous avons décidé d'utiliser les outils fournis avec la Kinect pour continuer le projet. Grâce à la caméra de Microsoft, nous pouvons récupérer le squelette de l'utilisateur dont les articulations sont labellisées avec le nom de la partie du corps humain à laquelle elle appartient (voir Fig. 15.a). De plus, la Kinect nous permet de séparer les points qui appartiennent à l'utilisateur de ceux qui appartiennent à l'environnement (voir Fig. 15.b).

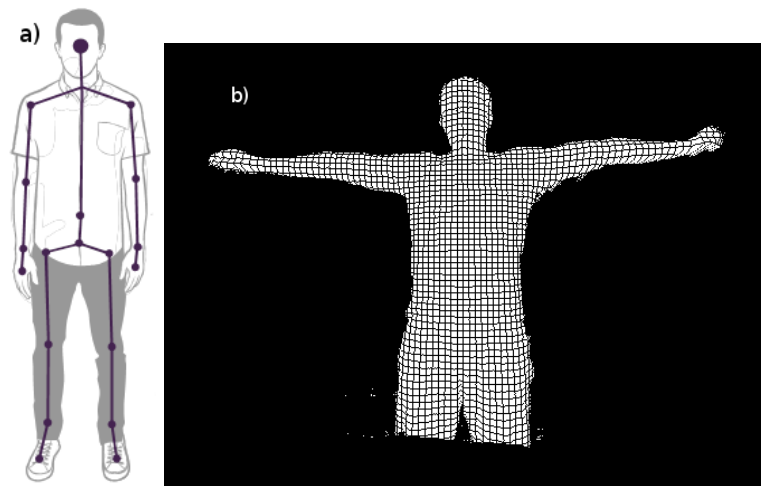


FIGURE 15 – a) Squelette fourni par la caméra Kinect⁹ et b) séparation du corps humain de l'environnement

Grâce à ce squelette et au nuage de points, nous pouvons segmenter le corps humain. Pour cela, pour chaque articulation nous définissons une zone dont la taille est variable en fonction de l'articulation. Les points du nuage sont labellisés en fonction de leur distance avec les articulations. Donc, un point prend le label de l'articulation dont il est le plus proche.

Nous pouvons voir que comme nous l'attendions, le résultat de cette segmentation est bien meilleur puisque nous avons à notre disposition beaucoup plus d'informations qu'auparavant grâce au squelette de l'utilisateur. Même si cette segmentation n'est pas parfaite, comme on peut le voir dans la première image, elle reste suffisante pour nos futurs traitements. En plus de la segmentation, nous connaissons déjà le nom des membres grâce aux labels des articulations. Nous n'avons donc pas besoin de passer par une étape de reconnaissance des parties du corps, car celle-ci a déjà été réalisée par les outils de la Kinect. Il ne reste plus qu'à supprimer le bruit présent dans les membres, les points mal labellisés, grâce à la classe `StatisticalOutlierRemoval` que nous avons vu précédemment.

9. source : <http://hdimagelib.com/standing+person+sideways?image=356490844>

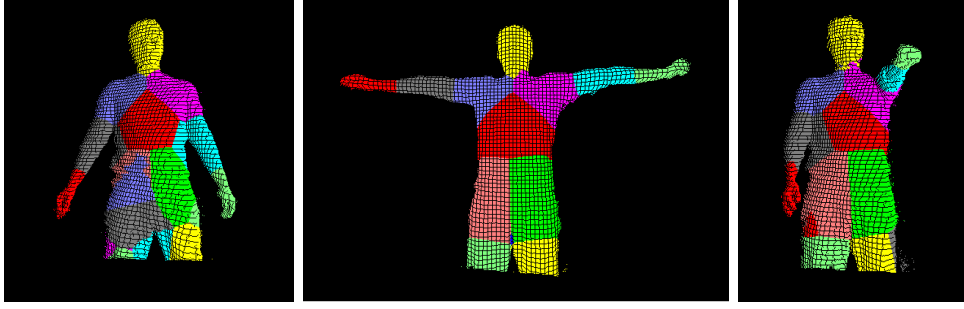


FIGURE 16 – Résultat obtenu avec la segmentation du corps humain via les outils de la Kinect

3.6 Positionnement

La dernière étape de cette première application est d'échanger le nuage de points d'un membre du corps par un modèle 3D représentant cette même partie du corps. Cette étape est très complexe à cause de deux problèmes majeurs. Tout d'abord, le type d'information n'est pas le même. Dans un cas, nous avons un nuage de points dont les points sont très concentrés, mais avec des parties occultées. De l'autre côté, nous avons un modèle 3D avec un nombre de points inconnus, donc une concentration de points inconnu, mais sans zone occultée si nous prenons en considération le maillage du modèle. Le second problème est que le but recherché est de remplacer la partie du corps humain par quelque chose de plus improbable et étonnant comme une partie du corps d'un monstre, d'un robot ou d'un corps dont la musculature est différente.

Nous testons dans un premier temps l'algorithme le plus utilisé dans la littérature en terme de correspondance de modèle 3D. L'algorithme ICP[16] est disponible dans la librairie PCL[1]. Après plusieurs tests sur différentes parties du corps, nous pouvons voir que l'algorithme ICP arrive à déterminer la bonne translation, si on compare les centroïdes des modèle 3D avec leur partie du corps correspondant. Cependant, ICP ne parvient pas à déterminer l'orientation du modèle. Le problème vient du fait que les données ne sont pas les mêmes. Bien que celles-ci soient différentes, la forme générale d'une partie du corps reste la même, que ce soit un modèle ou un nuage de points, donc sa boîte englobante reste similaire également. Nous utilisons la méthode de moment d'inertie de S. Ushakov disponible dans PCL pour déterminer l'orientation du nuage de points à partir de la matrice de covariance (voir Fig. 17).

Nous pouvons voir sur les résultats de la correspondance de modèle, qu'il y a deux problèmes importants. Le premier problème est celui de l'orientation de certains modèles 3D. Dans la majorité des tests que nous avons effectué, il y a au moins deux axes sur lesquels l'orientation est la bonne. La méthode permet de déterminer l'orientation, mais pas le sens, ce qui implique que dans certains cas, l'un des axes soit visuellement tourné du mauvais côté. Le second problème provient du non raccord du modèle avec le reste du nuage de points, qui est très bien illustré dans l'exemple de la pince de crabe. Pourtant dans chacun des cas, la translation est juste. Nous le vérifions en comparant la position du centroïde entre le modèle 3D et le nuage de points (il existe juste un petit décalage sur la profondeur à cause des zones occultées dans le nuage de points). Ce décalage est dû à une différence de ce qui est pris en compte comme étant une main (dans les exemples que nous avons vu). Dans le nuage de points, la zone considérée comme étant la main comporte la main et le poignet, alors que dans le modèle 3D de la main de monstre le poignet n'est pas présent.

Notre méthode serait suffisamment efficace pour remplacer deux nuages de points de membre du corps provenant d'une caméra 3D. On pourrait facilement remplacer la tête de deux per-

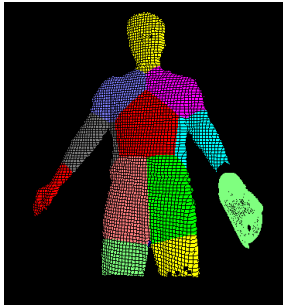
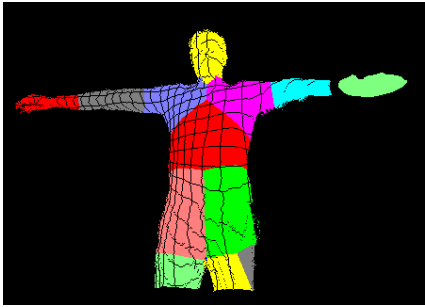
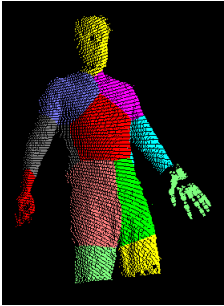
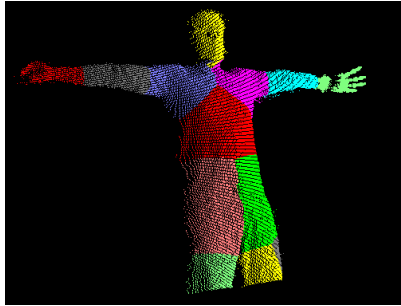
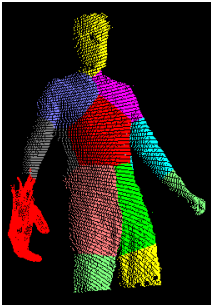
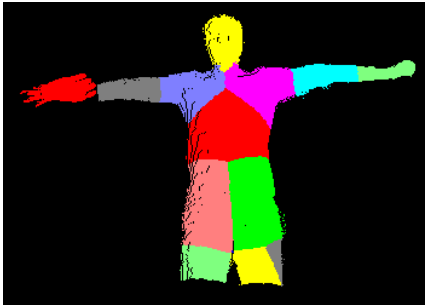
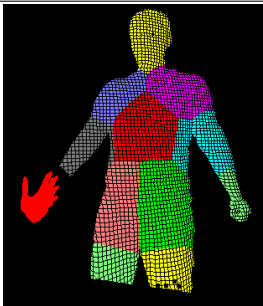
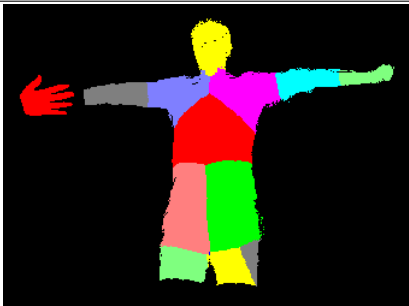
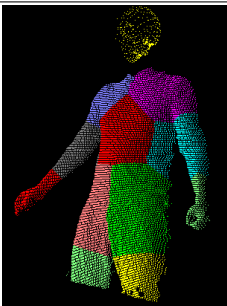
modèle	position 1	position 2
pince de crabe (main gauche)		
main robot (main gauche)		
main cybord (main droite)		
main de monstre (main droite)		
tête cyborg		

FIGURE 17 – Résultat de remplacement de parties du corps par des modèles 3D

sonnes. Mais la mise en place d'une correspondance entre un nuage de points et un modèle 3D d'une partie du corps n'est pas suffisamment précise. Pour pallier aux erreurs de l'application,

nous avons envisagé de laisser la possibilité à l'utilisateur d'interagir avec celle-ci. Le plus important est l'orientation du modèle, et nous savons qu'en cas d'erreur, il faut effectuer une rotation du modèle de 180° sur l'un des trois axes. L'utilisateur devrait juste sélectionner l'axe qui ne convient pas et l'application effectuerait la rotation seule. Maintenant que nous avons vu un cas particulier, nous allons pouvoir partir sur une application plus généraliste et plus simple en réutilisant ce qui a été vu pour l'application sur le corps humain.

3.7 Travaux futurs et applications

L'application que nous avons développée est vraiment la base, et de nombreuses extensions peuvent être ajoutées. L'une d'entre elles est la possibilité de bouger la partie du corps modifiée en temps réel. Pour l'instant, nous filmons la scène et l'utilisateur doit mettre en pause l'acquisition lorsqu'il souhaite modifier une partie du corps. Lorsque le processus est terminé, il serait intéressant de laisser la possibilité à l'utilisateur de relancer l'acquisition afin de voir la nouvelle partie du corps se déplacer. La possibilité de changer des parties du corps en temps réel durant l'acquisition, serait un véritable plus, mais la difficulté à optimiser les calculs serait trop importante. Sur l'application actuelle, nous sommes à 5 à 10s de calcul pour le remplacement d'une partie du corps.

Nous n'avons pas non plus eu le temps de développer l'interface de l'application. Pour l'instant, les modèles de remplacement sont prédéfinis pour chaque membre, donc l'utilisateur ne peut pas sélectionner le modèle qu'il souhaite.

L'un des aspects sur lequel nous avons réfléchi était la construction d'un modèle complet, sans partie occultée, du corps humain. Cette étape est réalisable avec un algorithme fourni dans le SDK de la Kinect qui est le « kinect fusion »[21]. Actuellement, nous mélangeons un nuage de points fourni par la Kinect, avec le dos de la personne occulté, et des modèles 3D dont on ne prend que les points. Même si le tout reste visuellement cohérent, on peut voir la différence entre le nuage de points et le modèle 3D. Cette extension permettrait, en plus d'une simple visualisation de l'utilisateur avec une partie du corps différente, de reconstruire un maillage complet avec le corps de l'utilisateur. Ce nouveau modèle 3D pourrait alors être directement utilisé dans un film d'animation ou un jeu vidéo.

4 Reconstruction d'un environnement intérieur

4.1 Objectif

Le but de cette seconde application est de pouvoir modifier un environnement intérieur au travers d'une interface simple et intuitive. Pour cela, nous allons dans un premier temps récupérer un nuage de points d'une pièce intérieure. Puis à partir de ce nuage de points, nous allons segmenter la pièce afin de pouvoir détecter des objets à l'intérieur de celle-ci. L'utilisateur doit ensuite sélectionner un objet, ce qui déclenchera l'apparition d'une liste contenant un ensemble de modèle 3D d'objets équivalents à celui sélectionné. L'utilisateur n'a plus qu'à choisir l'objet qui lui convient dans la liste, afin de l'ajouter dans un autre environnement 3D présent dans l'application.

Lors de ce scénario, nous pouvons voir qu'il y a deux grandes étapes qu'il faudra réaliser dans l'application. Tout d'abord, il faut segmenter le nuage de points de la pièce, dans le but de détecter des objets. Puis il faut reconnaître les objets détectés afin d'afficher la bonne liste d'objets 3D. Nous avons décidé, pour faciliter le développement, de laisser l'utilisateur sélectionner l'objet qu'il souhaite dans le nuage de points, afin de ne pas avoir à développer la détection

d'objet dans le nuage de points. Pour cela, nous nous inspirons des travaux de T. Shao et al[17] et de J. Xiao et al[18]. Il nous reste donc à reconnaître l'objet sélectionné par l'utilisateur.

4.2 Création de la base d'apprentissage

Afin de pouvoir reconnaître un objet, nous avons besoin d'une base d'apprentissage contenant les descripteurs d'un ensemble d'objets. Pour créer cette base, nous utilisons la représentation du « bag of word » ainsi que la méthode d'apprentissage automatique SVM[14] disponible dans la librairie opencv. Pour la création de notre base, nous utilisons une vingtaine d'images de six objets différents. Les nuages de points dont nous nous sommes servis pour la création de notre base, viennent de K. Lai et al[22] issue de M. Firman[23] qui a regroupé un ensemble de bases d'objets provenant de caméra 3D. La base que nous utilisons est composée de plusieurs exemplaires de chaque objet, et chaque exemplaire d'objet est représenté par plusieurs images représentant plusieurs angles de vue de l'objet. L'acquisition des objets a été réalisée avec une caméra Kinect v1.

Le descripteur que nous utilisons pour cette application est le descripteur FPFH[12] qui est l'un des plus efficaces dans la reconnaissance d'objet à partir de nuage de points. Nous calculons ce descripteur sur l'ensemble des images de notre base afin de créer le dictionnaire de notre *bag of words*. A cette étape du développement, nous avons créé le dictionnaire, il faut alors recalculer le descripteur sur chacun des objets afin de déterminer les caractéristiques de chacun d'entre eux. Cela nous permet d'obtenir un histogramme pour chaque objet comportant les caractéristiques qu'il contient et le nombre de fois où il apparaît. Il ne reste plus qu'à utiliser le descripteur du *bag of word* dans SVM afin de finaliser la création de notre base d'apprentissage. Celle-ci est donc un vecteur d'une taille égale au nombre de classes, donc six, composé d'histogramme dont la taille est égale au nombre de caractéristiques.

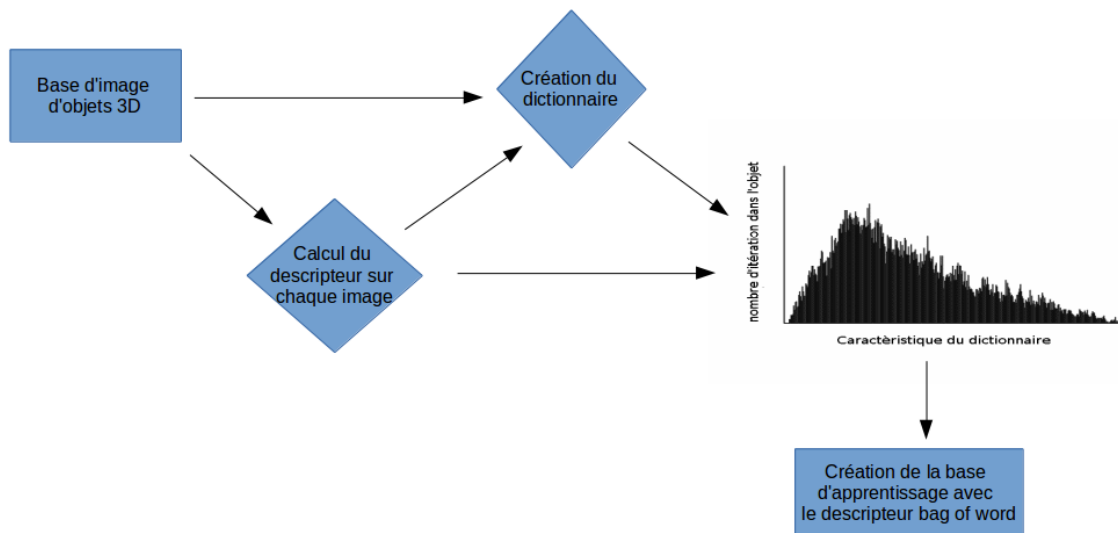


FIGURE 18 – Schéma de la création de la base d'apprentissage

Nous créons une seconde base d'apprentissage utilisant le même principe que précédemment, mais qui utilise des images que nous produisons nous même avec notre caméra Kinect v2. Cette base comporte quatre objets, chacun représenté par dix à quinze angles de vue différent. L'intérêt de cette seconde base est de vérifier que la qualité de l'acquisition ne fait pas varier

significativement les résultats de la reconnaissance d'objet. Cependant, le fait de créer nous même notre base ne nous permet pas d'avoir autant de données que dans la base de K. Lai et al[22] à cause du temps que cela prend de créer une telle base.

4.3 Interface utilisateur

L'interface utilisateur comprend deux fenêtres. La première comporte une vue avec le nuage de points récupéré à partir de la Kinect et un ensemble d'options. Parmi ces options, nous avons la navigation dans le nuage de points et le choix de la technique de sélection. La première sélection est une simple sélection rectangulaire où l'ensemble des données à l'intérieur d'un rectangle formé par deux points est sélectionné. La seconde est une sorte de pinceau sélectionnant les données en dessous de la souris, ainsi que celles dans un voisinage prédéfini autour de celle-ci. Lorsque l'utilisateur a sélectionné un objet et que l'application l'a reconnu, un nombre n de cases avec des objets apparaît. Les objets dans ces cases sont les mêmes que l'objet sélectionné. Lorsque l'utilisateur clique sur l'une de ces cases, l'objet correspondant est ajouté dans la seconde fenêtre. La seconde fenêtre contient l'environnement final dans lequel il y a déjà une pièce modélisée avec des objets 3D. L'objet sélectionné est ajouté dans cet environnement. Il est ensuite possible de déplacer l'objet dans l'environnement virtuel en cliquant dessus afin de le positionner à l'endroit voulu dans la pièce.

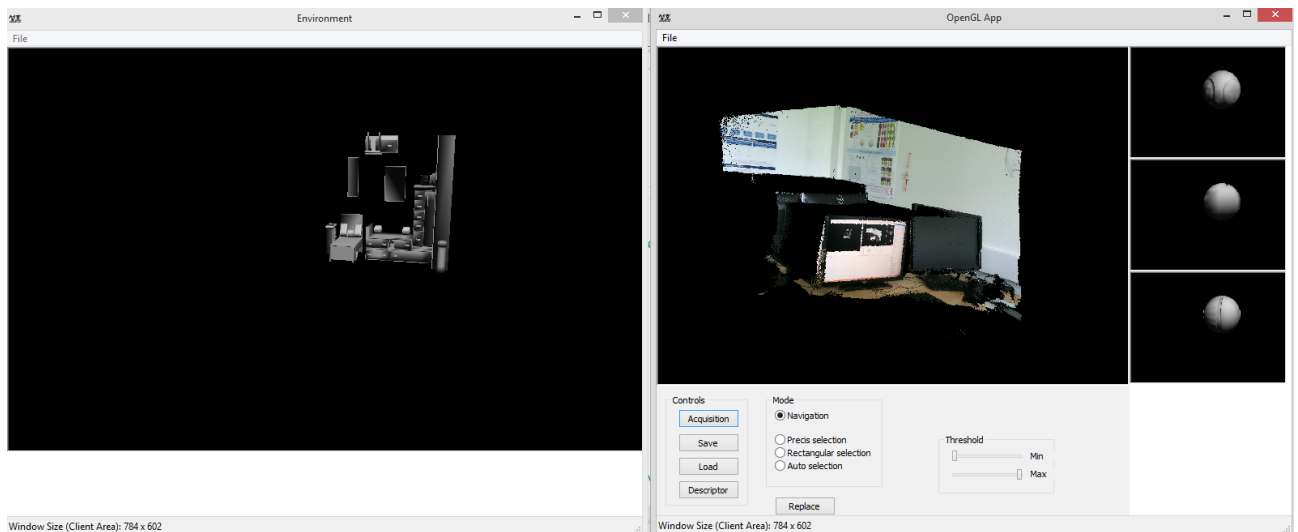


FIGURE 19 – Interface de la seconde application

4.4 Reconnaissance d'objet

Une fois la base d'apprentissage et l'interface créée, nous ajoutons le code de la reconnaissance de forme dans l'application. Pour cela, nous filtrons les données que l'utilisateur a sélectionné. Nos méthodes de sélection n'étant pas très précises, il arrive couramment que l'utilisateur sélectionne du bruit ou des parties de l'environnement. Pour filtrer les données, nous utilisons les deux mêmes classes que dans l'application précédente, c'est-à-dire *StatisticalOutlierRemoval* et *VoxelGrid*. Lorsque le filtrage est réalisé, il nous faut calculer le même descripteur que dans notre base sur les données sélectionnées par l'utilisateur, puis envoyer le résultat dans notre SVM afin qu'il détermine la classe de l'objet.

En utilisant les objets de la base de K. Lai et al[22] nous obtenons des résultats très insuffisants de la reconnaissance des objets provenant de notre caméra Kinect v2. Les différences entre

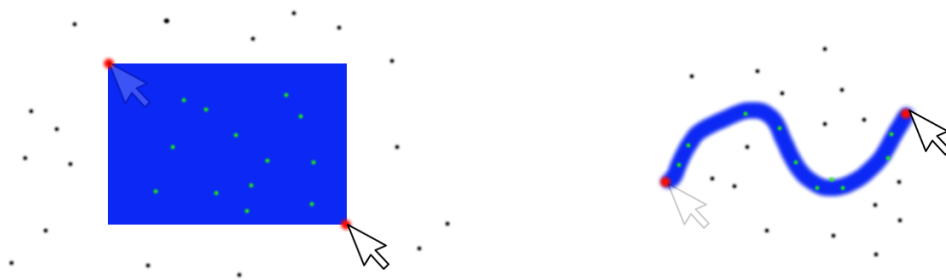


FIGURE 20 – Schéma des deux type de sélection disponible dans l’appllication. Les points verts sont les points sélectionnés et en noire les autres.

nos données et celle provenant de cette base viennent de notre sélection et de la caméra. Dans les données de la base, le bruit est presque null tandis que dans notre sélection il y en a un peu plus, malgré que nous en supprimons un maximum grâce à la librairie PCL. La principal différence entre la caméra de K. Lai et al[22] et la notre est la résolution de la caméra qui est plus grande dans notre cas. Avec ces paramètres, ne parvient à reconnaître qu’un objet sur les six testés et cela dépend du nombre de mot que nous mettons dans le dictionnaire. Cette première base nous à permis de déterminer que le nombre de mot idéal qu’il faut mettre dans le dictionnaire pour notre application est de deux milles.

4.5 Travaux futurs et applications

L’application finale de ce projet est assez complète et très fonctionnelle, cependant il existe plusieurs extensions qu’il est possible d’ajouter au projet. Par exemple, notre liste d’objets proposés à l’utilisateur pour ajouter à la scène 3D n’est pas assez conséquente. Nous pourrions ajouter plus d’objets en récupérant une base venant de grands magasins de meubles par exemple. Ainsi l’utilisateur pourrait potentiellement retrouver le meuble qu’il a scanné. Cela pourrait déboucher sur une application beaucoup plus importante, où l’environnement dans lequel nous souhaitons ajouter des meubles représente la future maison d’une personne. Si cette personne souhaite voir comment elle pourrait agencer ses meubles dans son prochain investissement, elle n’aurait qu’à récupérer un modèle 3D des lieux et scanner l’ensemble de ses meubles pour les ajouter dans la scène.

5 Conclusion

Durant ce stage, j’ai effectué deux projets permettant de réaliser la segmentation d’objets complexes comme le corps humain ou une scène intérieure à partir d’image 3D provenant d’une caméra Kinect. L’objectif était de créer des interfaces permettant de faciliter la tâche des professions artistiques en leur permettant d’utiliser des objets du monde réel dans un environnement virtuel. Dans la première application sur le corps humain, nous avons cherché à récupérer les informations du corps de la personne en face de la caméra et segmenter ces informations. Les informations ainsi segmentées pouvaient ensuite être remplacées par l’utilisateur afin de pouvoir mettre des éléments virtuels sur un corps venant du monde réel. Pour réaliser ce projet, nous avons utilisé les algorithmes fournis dans le SDK de la Kinect pour filtrer les données. Au vu de la différence des données entre le nuage de points que nous récupérons de notre caméra, et les modèles 3D que nous utilisons, nous avons privilégié une méthode basée sur l’utilisation des caractéristiques de la boîte englobante pour déterminer la position et l’orientation du nuage de

points de la partie du corps à remplacer. Ces informations nous permettent ainsi de transformer le modèle 3D pour que celui-ci soit exactement à la même position que l'ancien nuage de points. Le problème de notre méthode est qu'une boîte englobante est une représentation beaucoup trop générale pour une forme, et donc l'orientation du nouveau modèle possède souvent une erreur de 90° sur l'un des trois axes de la boîte. Pour l'application finale, nous avons pensé proposer à l'utilisateur de corriger ce problème en sélectionnant l'axe qui fait défaut ainsi l'application se chargerait d'appliquer la bonne rotation au modèle.

La seconde application avait pour but de prendre un environnement virtuel dans lequel nous pouvons rajouter des éléments du monde réel. Ces objets du monde réel viennent de l'acquisition d'une pièce via la caméra Kinect. Là encore, le processus nécessitait la segmentation des données afin de détecter les objets présents dans la pièce réelle. La segmentation est réalisée par l'utilisateur qui doit sélectionner l'objet qu'il veut importer dans le monde virtuel. Pour que l'application reconnaisse l'objet qui a été sélectionné, nous avons utilisé un algorithme d'apprentissage automatique, les SVM, avec le descripteur du *bag of words* formé à partir du descripteur FPFH.

Lors de ce stage, j'ai eu l'occasion de perfectionner les connaissances que j'avais acquises lors de mon master. Les notions que j'avais vues de reconnaissance d'objet m'ont évidemment été très utiles durant le stage, mais beaucoup de concepts n'ont pas été approfondis durant les cours par manque de temps. J'ai pu perfectionner mes connaissances concernant les algorithmes d'apprentissage automatique et j'ai pu voir le fonctionnement de plusieurs descripteurs.

Le plus compliqué lors de ce stage a été de comprendre et tester des méthodes proposées dans des publications scientifiques. Certaines méthodes n'étaient pas très complexes et faciles à tester, mais de nombreuses publications se reposaient sur des connaissances mathématiques qui n'ont pas été vues durant mon cursus scolaire. Cependant, cette expérience m'a appris à réaliser un état de l'art des solutions existantes sur un sujet très spécifique et à tester les méthodes proposées dans la littérature. Ma seconde difficulté, lors de ce stage, est que le sujet était assez vaste et qu'il ne fallait pas se perdre dans des méthodes qui s'éloignaient du sujet. Cela m'a appris à rester concentré sur l'objectif principal en prévoyant et en organisant les étapes du projet.

Ce stage m'a fait redécouvrir l'environnement de la recherche dans lequel je souhaitais m'orienter. Bien que j'ai réussi à surmonter les difficultés que j'ai rencontrées durant ce stage, il m'a permis de constater que l'environnement ne me convenait pas. La partie « état de l'art », bien que très intéressante, a été assez complexe à réaliser. De plus, j'ai ressenti une certaine frustration concernant les résultats du projet. J'ai passé énormément de temps à tester des méthodes difficiles à implémenter et je n'obtenais que rarement des résultats intéressants pour mon projet. Ce stage m'a permis de comprendre que le monde de l'entreprise me convenait mieux, grâce à son organisation et à sa structure qui permet de fournir des résultats rapides. Je pense qu'à l'heure actuelle, je ne suis pas encore capable de structurer un projet pour le mener à bien avec autant d'autonomie. Même si la recherche est un travail d'équipe, les projets s'organisent souvent seuls, et c'est sur ce point qu'il me reste encore à progresser.

Références

- [1] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Point cloud library," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/12, 2012.

- [2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 898–916, May 2011.
- [3] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, Jan 1979.
- [4] S. A. A. Shah, M. Bennamoun, and F. Boussaid, "A novel algorithm for efficient depth segmentation using low resolution (kinect) images," in *Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on*, pp. 603–607, June 2015.
- [5] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, pp. 248–253, 2006.
- [6] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.
- [7] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [8] B. Yoo, W. Kim, J. J. Han, C. Choi, D. Park, and J. Kim, "Randomized decision bush : Combining global shape parameters and local scalable descriptors for human body parts recognition," in *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 1560–1564, Oct 2014.
- [9] Y. Liu, P. Lasang, M. Siegel, and Q. Sun, "Geodesic invariant feature : A local descriptor in depth," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 236–248, 2015.
- [10] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Matching 3d models with shape distributions," in *Shape Modeling and Applications, SMI 2001 International Conference on*, pp. 154–166, IEEE, 2001.
- [11] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, *Persistent point feature histograms for 3D point clouds*. 2008.
- [12] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 3212–3217, IEEE, 2009.
- [13] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1150–1157 vol.2, 1999.
- [14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, (New York, NY, USA), pp. 144–152, ACM, 1992.
- [15] V. Garg, S. Vempati, and C. Jawahar, "Bag of visual words : A soft clustering based exposition," in *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCV-PRIPG), 2011 Third National Conference on*, pp. 37–40, IEEE, 2011.
- [16] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 239–256, Feb 1992.
- [17] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an rgbd camera," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 136, 2012.
- [18] J. Xiao, A. Owens, and A. Torralba, "Sun3d : A database of big spaces reconstructed using sfm and object labels," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1625–1632, 2013.
- [19] E. W. Dijkstra *et al.*, "On the cruelty of really teaching computing science," *Communications of the ACM*, vol. 32, no. 12, pp. 1398–1404, 1989.

- [20] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2274–2282, Nov 2012.
- [21] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion : Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, (New York, NY, USA), pp. 559–568, ACM, 2011.
- [22] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1817–1824, IEEE, 2011.
- [23] M. Firman, "RGBD Datasets : Past, Present and Future," in *CVPR Workshop on Large Scale 3D Data : Acquisition, Modelling and Analysis*, 2016.