

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

Elliot Vanegue

9 août 2016

Remerciement

Résumé

Table des matières

1	Introduction	5
1.1	Contexte	5
1.2	Sujet	5
1.3	Problématique	6
1.4	Organisation	6
2	Etat de l'art	7
2.1	Segmentation	7
2.1.1	Segmentation d'une scène intérieur	7
2.1.2	Corps humain	8
2.2	Reconnaissance d'objets	9
2.2.1	Descripteur	9
2.2.2	Apprentissage automatique	11
2.2.3	Sac de mot (bag of word)	11
2.3	Positionnement de modèle	12
2.4	Interaction utilisateur	12
3	Modification des membres du corps humain	13
3.1	Objectif	13
3.2	Délimitation du corps humain et minimisation de la quantité de donnée	14
3.3	Calcule de la distance géodésique	15
3.4	Segmentation du corps humain	15
3.5	SDK de la Kinect	17
3.6	Positionnement	18
4	Reconstruction d'un environnement intérieur	20
4.1	Objectif	20
4.2	Création de la base d'apprentissage	20
4.3	Interface utilisateur	21
5	Conclusion	21

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 le laboratoire 3D-SAM spécialisé dans l'acquisition et le traitement d'image 3D à partir de capteur 3D de type Microsoft Kinect. Leur principaux travaux porte 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 planifié et ce stage me permet de me décider sur l'environnement de travail qui me convient le mieux. Durant mes deux derniers stage 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 proposé par d'autre laboratoire de recherche.

Ces deux années de master nous ont beaucoup appris sur le traitement d'image et la reconnaissance de forme 2D. Ces sujets mon particulièrement intéressé et je souhaite, durant ce stage, approfondir ces notions sur des types de donnée plus complet 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 dirigé par M. Daoudi comprend cinq membres permanents, un post-doctorant et quatre doctorant. Mes encadrant 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 produit 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 approprié durant leur processus de création ou en générant automatiquement de nouvelles ressources à partir d'éléments existant.

1.2 Sujet

Comme nous pouvons le constater depuis quelques années, les approches 3D interactive 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 complexe, dans le loisir avec les jeux vidéo dont le revenu mondial en 2015 est de plus de 90 milliard de dollars ou encore dans l'industrie pour visualiser des produits. L'engouement pour cette technologie requiert des outils de plus en plus efficace et rapide 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 virtuelle 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 membre improbable comme un bras de robot. Le principe est le même dans une pièce, si nous souhaitons remplacer des meubles d'une pièce intérieur par d'autre plus étonnant. L'intérêt de cela est d'éviter à un designer de devoir remodeliser une personne ou une pièce existante qui serait déjà idéal 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 capteur optique basé sur des caméras

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

infrarouge avec des marqueurs réfléchissant. 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 point de l'environnement qu'elles enregistrent. Cependant, ce type de caméra est peu précise et elles sont souvent bruitées et comportent des valeurs qui n'existent pas dans le monde réel. Ce bruit dépend de plusieurs facteurs comme la luminosité de l'environnement dans lequel nous effectuons l'acquisition ou tout simplement la qualité et la précision du capteur utilisé. La première difficulté lors de ce projet est de réussir à filtrer les données de sorte à ce qu'il ne reste que les données réellement présentes. L'ensemble des données ne nous intéresse pas forcément. En effet, si nous travaillons sur le corps humain nous n'avons pas besoin de l'ensemble de la pièce intérieure, 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 un membre comme la main il faut d'abord savoir quel membre représente la main. La reconnaissance d'objet dépend fortement du type de donnée 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 est qu'il y a un écart entre deux membres, la scène perd toute sa crédibilité et son réalisme.

Nous avons donc trois problèmes majeurs 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 exacte des objets présents dans notre scène ?

1.4 Organisation

Pour répondre aux questions précédentes, je vais travailler sur deux applications mettant en avant des problèmes similaires, mais avec deux façons différentes d'aborder ces problèmes. Je vais travailler dans un premier temps sur une application qui se focalise sur le corps humain. Cette application aura 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. La seconde application se concentre sur une pièce intérieure comportant plusieurs objets. Ici le but est de segmenter la pièce afin de 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, je vais utiliser un ensemble de bibliothèques tel que la bibliothèque graphique de Microsoft pour la réalisation de l'interface, la bibliothèque « Point Cloud Library » [1] (PCL) pour les calculs sur les nuages de points et opencv pour travailler sur les images couleurs et les

images de profondeur fournit par la Kinect.

Je vais dans un premier temps faire l'état de l'art des solutions existantes sur les problématique de segmentation et de reconnaissance de forme à partir de nuage de point ou d'image de profondeur. Puis je vais décrire les solutions testé et appliqué au deux applications que j'ai développé durant ce projet.

2 Etat de l'art

2.1 Segmentation

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 important 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 supprimer pour gagner du temps lors de nos traitements. Une second segmentation est nécessaire pour le traitement du corps humain. Pour cette étape du projet, nous devons segmenter le corps en plusieurs partie pour pouvoir par la suite les reconnaitres. Si cette seconde segmentation n'est pas réalisé il ne nous sera pas possible de reconnaître les mains ou encore la tête si nous ne savons pas délimité les parties du corps.



FIGURE 1 – Résultat attendu pour la segmentation d'une pièce intérieur et du corps humain

2.1.1 Segmentation d'une scène intérieur

De nombreux travaux ont été réalisé dans la segmentation d'image 2D avant que les caméras 3D ne soit ouvert 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 utilise un simple seuillage en utilisant par exemple la méthode de N. Otsu[3] pour binariser l'image et ainsi la segmenter.

Avec l'arrivé des caméras 3D de nombreuses recherche ont été effectué 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 point, 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 point sont différentes et des méthodes de segmentation se sont développées autour de ces informations.

T. Rabbani et al[5] utilise les informations obtenus dans un nuage de point 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 ses voisins sont inférieurs au seuil alors les points appartiennent à la même région.

Nous pouvons voir que les méthodes citées précédemment sont efficaces 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 importante et est trop instable pour que cela marche sur le corps humain qui peut adopter de nombreuses postures.

2.1.2 Corps humain

La segmentation du corps humain est un sujet très complexe, car contrairement à l'objet 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 labélisation 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 labélisées avec les différentes parties du corps. J. Shotton et al[6] ont d'abord créé une base d'apprentissage en calculant un descripteur et une technique d'apprentissage automatique appelé forêt d'arbres décisionnels[7]. Le descripteur de J. Shotton et al[6] utilise les informations de l'image de profondeur. Afin de déterminer quelle posture a l'utilisateur, il utilise une caractéristique reposant sur la valeur de deux pixels, un pixel x et d'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 à la posture de la base d'apprentissage.



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.

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 le centre de la main. Mais cette méthode permet tout de même de déterminer approximativement les parties du corps occultées.

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 complexe 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 outils 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 arrête de ce modèle. Etant donné que ce n'est pas un modèle 3D que fournit la Kinect, mais un nuage de point, les auteurs forment un maillage en créant n arrêtes avec les n points les plus proches pour un point donné. Cette distance géodésique permet de connaître l'orientation des points. Cette orientation est appliqué sur d'autres caratéristiques sensible à la rotation, ce qui permet à ces caractéristique d'être invariant en rotation.

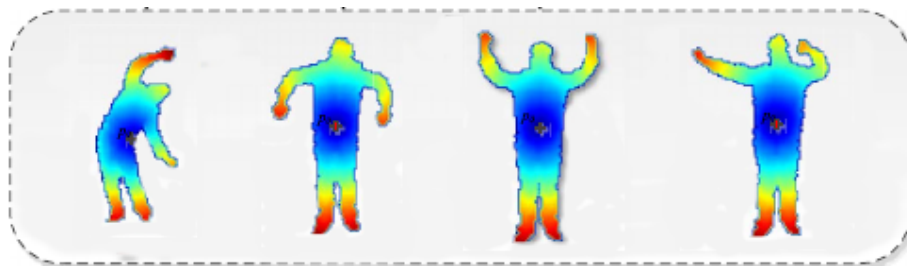


FIGURE 3 – Distance géodésique d'un corps humain

2.2 Reconnaissance d'objets

La reconnaissance d'objet 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 la méthode la plus utilisé est le calcul de descripteur, ce qui correspond à un ensemble de caractéristiques représentant un objet spécifique.

2.2.1 Descripteur

Le nombre de descripteur qui existe 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 utilisées. 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 histogramme afin de reconnaître un objet. Ce descripteur fournit de bon résultat 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 point en utilisant un histogramme multi-dimensionnel. Le calcul de la courbure et le fait que ce soit une généralisation permet d'être invariant en translation, en rotation et en densité de point, et permet également d'être moins sensible au bruit présent dans le nuage de point. Le voisinage dépend de la distance des points avec le point centrale et il ne peut excéder un certain nombre de voisin (voir Fig. 4).

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 normals entre deux points.

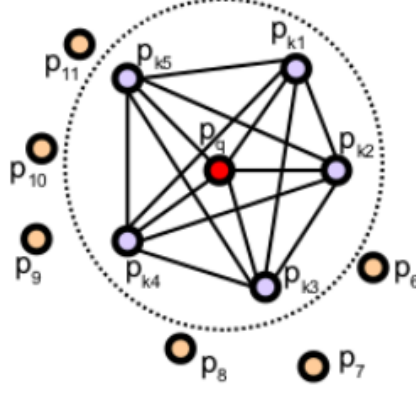


FIGURE 4 – Exemple de voisinage pris en compte dans le calcul de la courbure du descripteur PFH

Chaque classe de l'histogramme est composé de l'ensemble des points du voisinage dont la relation avec le point centrale est similaire. Une version amélioré du descripteur a été proposé par R. B. Rusu et al[12] appelé FPFH (Fast Point Feature Histograms). Cette version est plus rapide, car elle calcul un descripteur PFH simplifié, puis elle construit de nouveaux histogrammes à partir des histogramme simplifié précédent (voir Fig. 5).

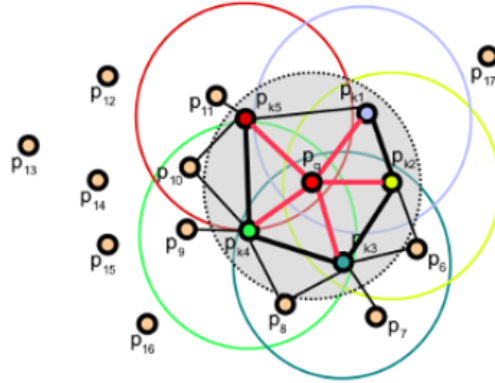


FIGURE 5 – Exemple de voisinage pris en compte dans le calcul de la courbure du descripteur FPFH

D. G. Lowe[13] a créé un descripteur appelé « scale-invariant feature transform » (SIFT) qui crée des points clés dans une image 2D et calcul 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 petit. Les points clés de ce descripteur sont des zones circulaires positionné sur les extrema 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 clé, il faut déterminer leur orientation en calculant le gradient dans le voisinage du point clé. Grâce au orientation des voisins des point clé 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 invariant 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 cette outil à besoin de donnée déjà traité sur un domaine précis. Il existe de nombreuses techniques d'apprentissage automatique. Lors de ce projets, je me suis intéressé à deux techniques en particulier qui sont parmi les plus utilisé 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'arbre. 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 calculés. L'algorithme test 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 rechercher nous fournit la solution à notre problème.

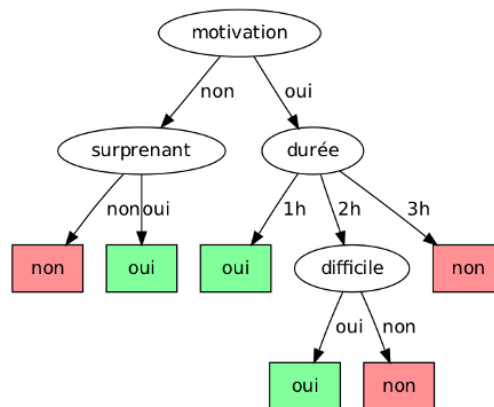


FIGURE 6 – Exemple d'arbre de décision

Les machines à vecteurs de support[14] quant à eux sont une classe d'algorithme d'apprentissage supervisé permettant la résolution de problème de discrimination non linéaire. Le 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 recherche l'hyperplan le plus approprié à la séparation des données dans l'espace vectoriel.

2.2.3 Sac de mot (bag of word)

Le sac de mot est une représentation dont la première utilisation était de décrire un document texte en fonction d'un dictionnaire. Le dictionnaire est un ensemble de mot 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 à ensuite été utilisé dans le domaine de la vision par ordinateur[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 du dictionnaire. 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.

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). Cette algorithm recherche l'ensemble des translations et rotations nécessaire 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.
- 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 library PCL[1], S. Ushakov implémente une classe basé sur le moment d'inertie² permettant de déterminer les axes principaux d'un nuage de point. Pour cela cette classe calcule la matrice de covariance du nuage de point et en extrait les vecteurs propres. Le plus grand vecteurs propre est considéré comme étant l'axe X et le plus petit devient l'axe Z, ce qui permet d'obtenir un système de coordonnée dans un référentiel local à l'objet.

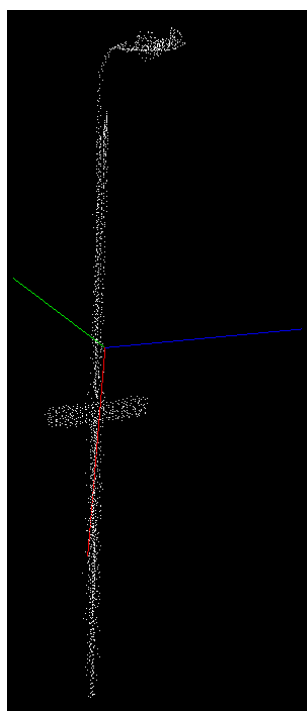


FIGURE 7 – Résultat du calcul des axes sur un nuage de point grâce à la méthode utilisant le moment d'inertie

2.4 Interaction utilisateur

T. Shao et al[17] propose, 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

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

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 et l'application change la couleur et le label de l'objet sélectionné.

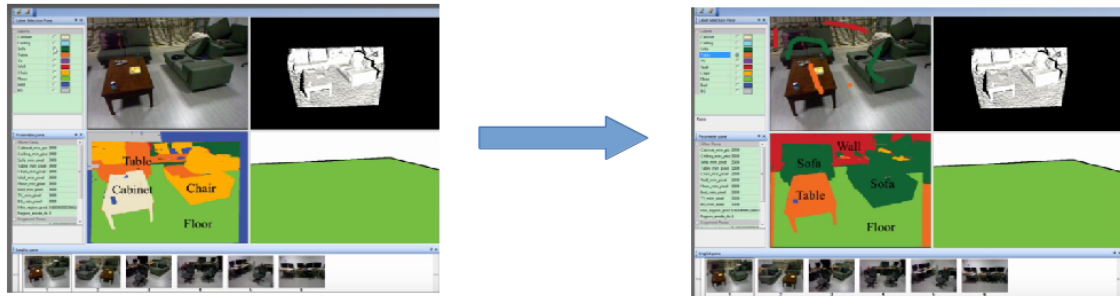


FIGURE 8 – Résultat de la reconnaissance interactive de T. Shao et al[17]

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.

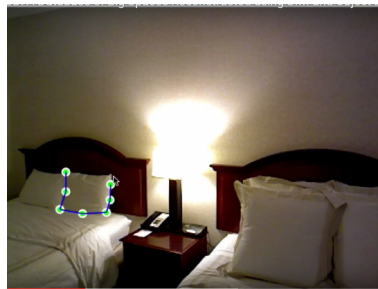


FIGURE 9 – Résultat de la sélection interactive de J. Xiao et al[18]

3 Modification des membres du corps humain

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 devenu plus stable et plus précise. Le second objectif est de remplacer un nuage de point 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 second 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 point 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 rapporte pas au corps humain. La première solution que nous développons consiste à utiliser deux seuils. Les points qui sont plus éloigné que le premier seuil ainsi que les points en dessous du second seuil sont supprimé. Ces seuils sont les distances, en mètre, dans lequel l'utilisateur doit se trouver. Cette méthode est utilisé dans l'application ReconstructMe³ qui est une application permettant de construire un modèle 3D complet à partir de données fournit par plusieurs images provenant d'une caméra 3D. Comme pour cette application, j'ai décidé de laisser la possibilité à l'utilisateur de changer les seuils en fonction de sont besoin.



FIGURE 10 – Résultat d'une segmentation par seuillage

Nous pouvons voir sur la Fig. 10 que le seuil permet effectivement de supprimer beaucoup d'information correspondant à l'environnement, mais qu'il reste beaucoup de bruit du à la qualité de l'acquisition de la caméra ainsi que des objets qui se trouve à la même distance que l'utilisateur. La librairie PCL[1] nous fournit beaucoup d'outil pour ce genre de problématique. Il y a une classe appelé `StatisticalOutlierRemoval` qui permet de supprimer les points supposé être du bruit. Pour cela cette classe calcul la distance moyenne d'un point avec son voisinage et si cette moyenne est trop élevé, 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 point de ces objets leur permet d'avoir suffisamment de voisin proche pour avoir une moyenne de distance très petite. Pour pallier à ce problème, nous avons ajouter un système de sélection permettant à l'utilisteur de sélectionner les données ne faisant pas partie du corps humain.

Pour la suite des traitements que nous souhaitons réalisé, il est préférable d'avoir un minimum d'information et de ne garder que ce qui est pertinent. Le corps humain que nous avons réussi à délimité comporte encore beaucoup trop de données. Le nombre de point fournit par la Kinect est très important et très concentré, il est possible de supprimer des points qui sont trop proche les uns les autres. La encore PCL[1] peut nous aider avec la classe `VoxelGrid`. Cette classe crée une grille de voxel sur le nuage de point dont la taille est définit par l'utilisateur. L'ensemble des points à l'intérieur d'un voxel sont approximé en un point qui correspond au centroïde du voxel. Grâce à l'ensemble de ces traitements nous ne gardons que l'information essentiel à nos

3. <http://reconstructme.net>

traitements.

3.3 Calcule de la distance géodésique

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

Avant de calculer la distance géodésique du corps humain, nous avons besoin de créer un maillage sur le nuage de point. 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 point avec tous les autres. Pour la création du voisinage nous considérons non seulement le nombre maximum de point 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 second condition, car lorsque la main de l'utilisateur était trop proche de sa jambe certain point de la main avait 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 du maillage. Pour améliorer la qualité de ce maillage nous avons créé deux paramètres que l'utilisateur peut modifier. Le premier paramètre est le nombre de voisin d'un point et le second est la distance euclidienne maximal 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 nage de point 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.

Sur les images de la Fig. 12 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 traverse dans le vide du ventre au bras, ce qui réduit la distance géodésique au niveau des mains. Dans la second image, on voit que la position du centroïde est très importante. La position du centroïde 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.

3.4 Segmentation du corps humain

Malgré un léger manque de précision de la par 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.

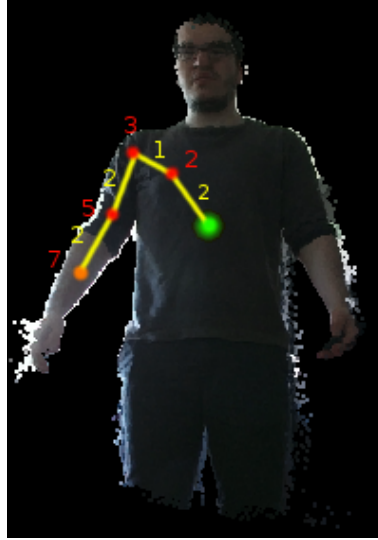


FIGURE 11 – Exemple de chemin parcouru par l’algorithme de calcul de la distance géodésique pour un point. Le point vert correspond au centre de gravité du corps, les points rouge sont les noeuds du chemin et le point orange est le point d’arrivée. Les chiffres jaune sont les distances pour aller d’un noeud à l’autre et les chiffres rouge est la distance géodésique au noeud correspondant.

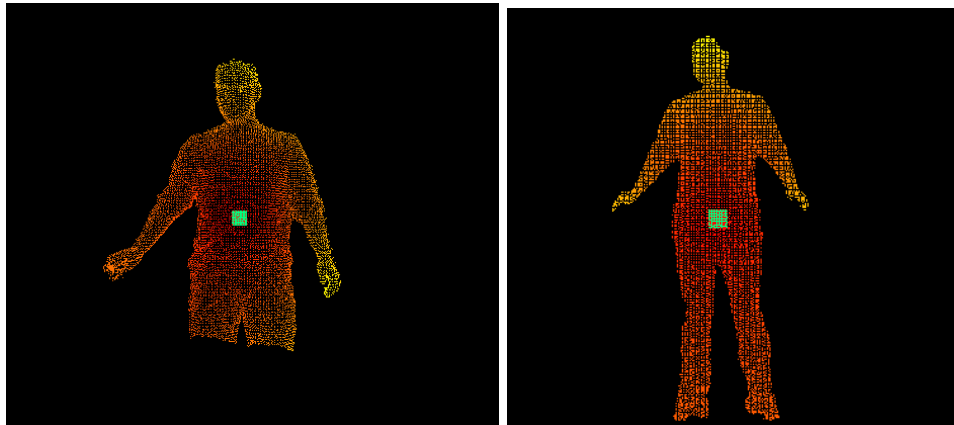


FIGURE 12 – Résultat du calcul de la distance géodésique sur l’ensemble du nuage de point. Le point vert correspond au centroïde du nuage de point. Plus la couleur des points est proche de rouge, plus les points sont proches du centroïde.

Cette zone a un rayon prédéfini qui sera le même pour chaque membre. Les points à l’intérieur de cette zone seront enregistré et sont considéré comme faisant partie d’un même membre. Ils seront ensuite supprimer du nuage de point du corps humain, afin de ne pas être pris en compte dans les étape suivante.

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

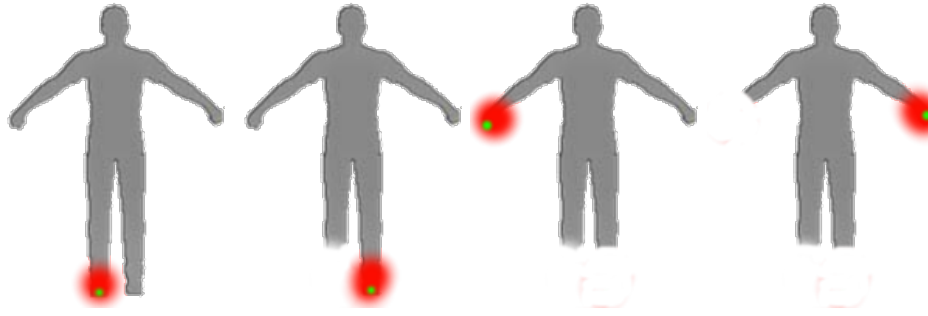


FIGURE 13 – Premières étape de l’algorithme de segmentation du corps humain. Le point vert correspond au point le plus éloigné et la zone rouge l’ensemble des points considérés comme faisant partie du membre.

ment 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 humaine comme les mains, les pieds et la tête, mais 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 labelisé avec le nom de la partie du corps humain à laquel elle appartient (voir Fig. 14.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. 14.b).

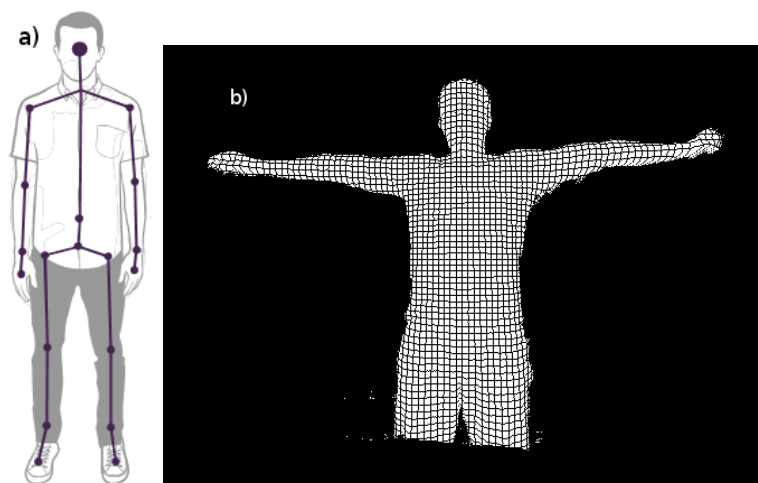


FIGURE 14 – a) Squelette fournit par la caméra Kinect et b) séparation du corps humain de l’environnement

Grâce à ce squelette et au nuage de point 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 labelisé en fonction de leur distance avec les articulations. Donc un point prend le label de l'articulation dont il est le plus proche.

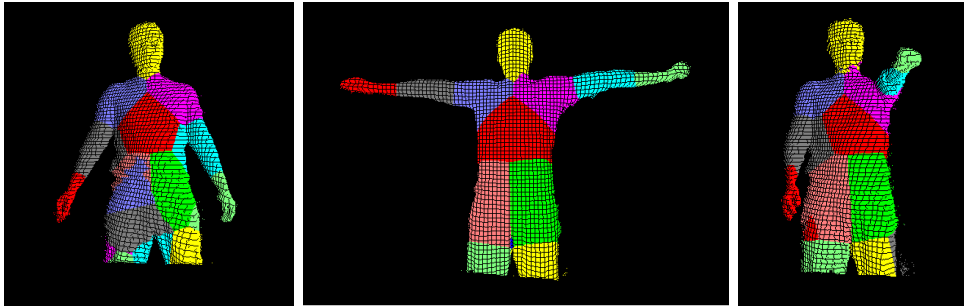


FIGURE 15 – Résultat obtenu avec la segmentation du corps humain via les outils de la Kinect

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'information 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 futures traitement. 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 réalisé par les outils de la Kinect. Il ne reste plus qu'à supprimer le bruit présent dans les membres, les points mal labeliser, grâce à la classe `StatisticalOutlierRemoval` que nous avons vu précédemment.

3.6 Positionnement

La dernière étape de cette première application est d'échanger le nuage de point d'un membre du corps par un modèle 3D représentant cette même partie du corps. Cette partie 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 point dont les points sont très concentrés, mais avec des parties occultés. De l'autre côté, nous avons un modèle 3D avec un nombre de point inconnu et donc une concentration de point inconnu, mais avec aucune zone occulté 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 différentes.

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 partie 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. Malgré que les données soit différentes, la forme général d'une partie du corps reste la même que ce soit un modèle ou un nuage de point, 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 point à partir de la matrice de covariance (voir Fig. 16).

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 certain modèle 3D. Dans la ma-

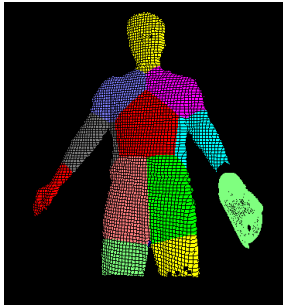
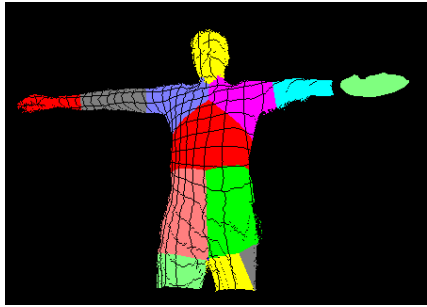
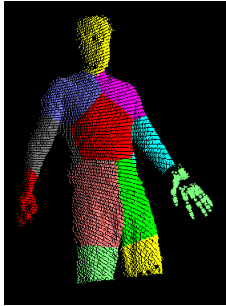
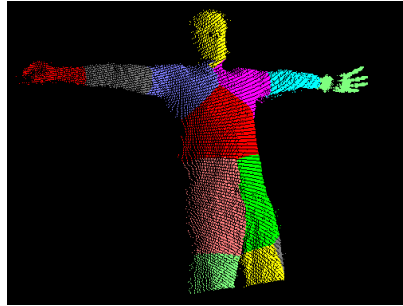
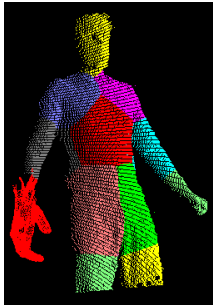
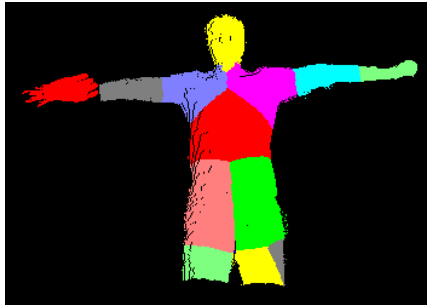
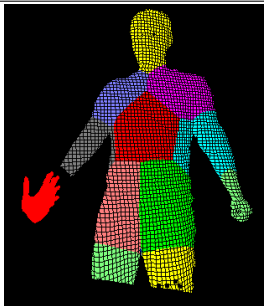
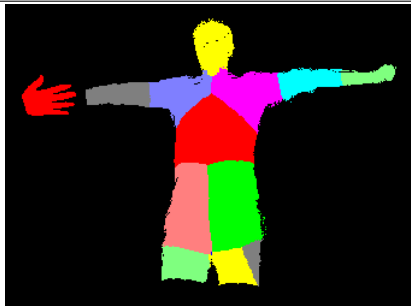
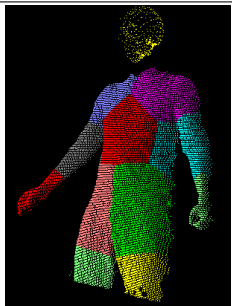
modèle	position 1	position 2
pince de crabe (main gauche)		
main robot (main gauche)		
main cybord (main droite)		
main de monstre (main droite)		
tête cyborg		

FIGURE 16 – Résultat de remplacement de partie du corps par des modèles 3D

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 tourné visuellement du mauvais côté. Le second problème provient du non raccord du modèle avec le reste du nuage de point 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 point (il existe juste un petit décalage sur la profondeur à cause des zones occultées dans le nuage de point). Ce décalage est dû à une différence de ce qui est pris en compte comme étant une main par exemple. Dans le nuage de point, 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 point de membre du corps provenant d'une caméra 3D. On pourrait facilement remplacer la tête de deux personnes. Mais la mise en place d'une correspondance entre un nuage de point 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 aura juste à sélectionner l'axe qui ne convient pas et l'application effectuera la rotation seul. 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.

4 Reconstruction d'un environnement intérieur

4.1 Objectif

Le but de cette seconde application est de pouvoir modifier un environnement d'une pièce intérieure au travers d'une interface simple et intuitive. Pour cela, nous allons dans un premier temps récupérer un nuage de point d'une pièce intérieure. Puis à partir de ce nuage de point, 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'objet équivalant à celui sélectionné. L'utilisateur n'a plus qu'à sélectionner 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 point de la pièce dans le but de détecter des objets. Puis il faut reconnaître les objets détectés afin d'afficher la bonne liste d'objet 3D. Nous avons décidé, pour faciliter le développement, de laisser l'utilisateur sélectionner l'objet qu'il souhaite dans le nuage de point afin de ne pas avoir à développer la détection d'objet dans le nuage de point. 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 soixantaine d'image 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[21] et de M. Firman[22] qui a regroupé un ensemble de base d'objet provenant de caméra 3D.

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 point. Nous calculons ce descripteur sur l'ensemble des images de notre base afin de créer le dictionnaire de notre bag of word. 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 chaque objet. Cela nous permet d'obtenir un histogramme pour chaque objet comportant les caractéristique qu'il contient et le nombre de fois qu'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.

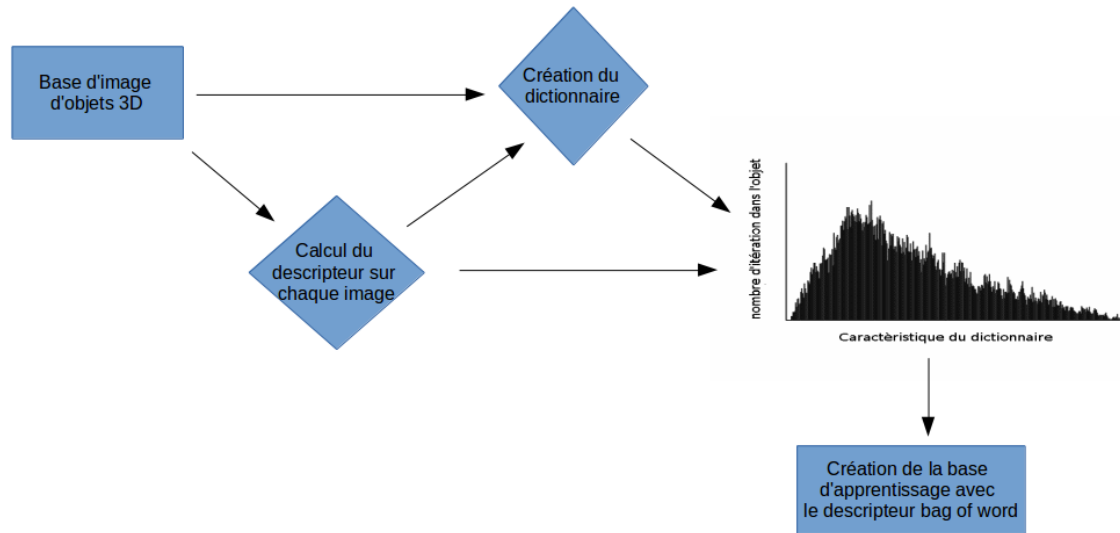


FIGURE 17 – Schéma de la création de la base d'apprentissage

4.3 Interface utilisateur

L'interface utilisateur comprend deux fenêtres. La première comporte une vue avec le nuage de point récupérer à partir de la Kinect et un ensemble d'option. Parmi ces options, nous avons la navigation dans le nuage de point 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 sont sélectionné. La second est une sorte de pinceau sélectionnant les données en dessous de la souris et celles dans un voisinage d'une taille variable. Lorsque l'utilisateur a sélectionné un objet et que l'application l'a reconnu, un nombre n de case 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'objets correspondant est ajouté dans la seconde fenêtre. La second fenêtre contient l'environnement final dans lequel il y a déjà une pièce modélisé avec des objets 3D. L'objet sélectionné est ajouté dans cette environnement.

5 Conclusion

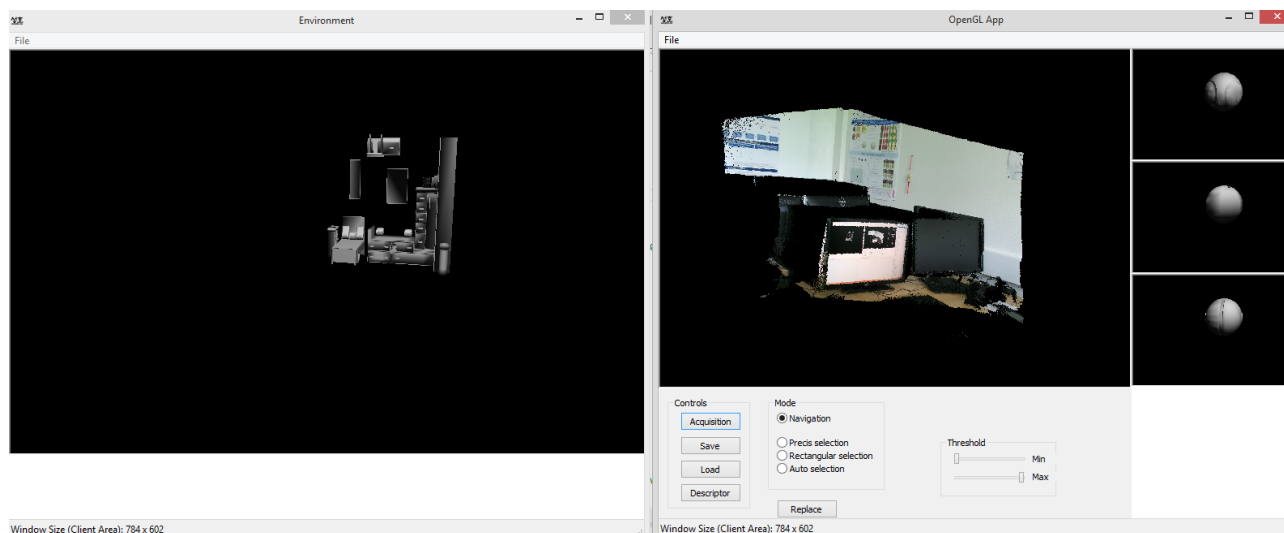


FIGURE 18 – Interface de la seconde application

Références

- [1] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlking, 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] J. Sivic and A. Zisserman, “Video google : a text retrieval approach to object matching in videos,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1470–1477 vol.2, Oct 2003.
- [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. Süsstrunk, “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] 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.
- [22] M. Firman, “RGBD Datasets : Past, Present and Future,” in *CVPR Workshop on Large Scale 3D Data : Acquisition, Modelling and Analysis*, 2016.