Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement

2018-2019

HoloLens, Réalité Augmentée et Epaule

Tuteurs académiques
Jordan NICOT
Mohand SLIMANE

Étudiant
Grégoire MARCO (DI5)

3 avril 2019

Liste des intervenants

Nom	Email	Qualité
Grégoire MARCO	gregoire.marco@etu.univ-tours.fr	Étudiant DI5
Jordan NICOT	jordan.nicot@outlook.com	Tuteur académique, Département Informatique
Mohand SLIMANE	mohand.slimane@univ-tours.fr	Tuteur académique, Département Informatique

Avertissement

Ce document a été rédigé par Grégoire MARCO susnommé l'auteur.

L'Ecole Polytechnique de l'Université de Tours est représentée par Jordan NICOT et Mohand SLIMANE susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Grégoire MARCO, *HoloLens, Réalité Augmentée et Epaule*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2018-2019.

```
@mastersthesis{
    author={MARCO, Grégoire},
    title={HoloLens, Réalité Augmentée et Epaule},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université de Tours},
    address={Tours, France},
    year={2018-2019}
}
```

Document mis en forme sous LaTeX

grâce à la classe de document `polytech.cls` V2.2.5 (2017/05/02)

Table des matières

Liste des intervenants a

Avertissement b

Pour citer ce document c

Table des matières i

Table des figures v

Remerciements 1

1 Introduction 2

1 Acteurs, enjeux et contexte 2

1.1 Contexte du projet 2

1.2 Enjeux 2

1.3 Acteurs 2

2 Objectifs 3

2.1 Objectif général 3

2.2 Les états de l'art 3

2.3 Récupération d'un nuage de points 3

3 Hypothèses 3

4 Bases méthodologiques 4

4.1 Gestion de projet 4

4.2 Démarche 4

2 Description générale 6

1 Environnement du projet 6

2 Caractéristiques des utilisateurs 7
3 Fonctionnalités du système 7
4 Structure générale du système 8

3 État de l'art **9**
1 Les modèles 3D 9
1.1 Définition d'une image 3D 9
1.2 Définition d'une vidéo 3D 9
1.3 Obtention d'un nuage de points 9
1.4 Reconstruction des surfaces 9
1.5 Les fichiers de scène 3D 10
1.6 Les référentiels de visualisation 11
2 Les caméras de profondeur 12
2.1 Les caméras *actives* 12
2.1.1 Temps de vol (en anglais *time of flight*, abbrégé ToF) 12
2.1.2 Lumière structurée 13
2.2 Les caméras *passives* 14
2.2.1 Stéréoscopie 14
2.2.2 Les caméras plénoptiques 14
2.3 Récapitulatif et conclusion 15
3 Les lunettes HoloLens 16
3.1 Les composants matériels 16
3.2 Les caractéristiques techniques 16
3.3 Le Windows Device Portal 17
3.4 La gestuelle et la reconnaissance vocale 18

4 Analyse et conception **19**
1 Prise en main des lunettes HoloLens 19
2 Découverte du HoloLens Research mode 19
2.1 Accès aux caméras de localisation 20
2.2 Accès à la caméra d'enregistrement vidéo 20
3 Conception de l'application 22
3.1 Principe algorithmique 22
3.2 Modélisation de l'application 23
3.3 Première version : récupération et affichage du flux de profondeur 24
3.3.1 Récupération du flux 24
3.3.2 Présentation à l'utilisateur 25
3.4 Deuxième version : récupération des données de profondeur 25
3.5 Version finale 26
3.5.1 Changement du référentiel 26
3.5.2 Ecriture d'un fichier .obj 26

5 Bilan et Conclusion 27
1 Bilan du PRD1 ... 27
2 Bilan du PRD2 ... 27

Annexes 29

A Spécifications fonctionnelles 30
1 Récupérer un flux d'images Bitmap ... 30
2 Déterminer la profondeur des points de la cible ... 30
2.1 Version 1 ... 30
2.2 Version 2 ... 31
3 Obtention d'un fichier .obj ... 31
3.1 Changement de référentiel ... 31
3.2 Ecriture dans un fichier .obj ... 31
4 Suppression des doublons du fichier .obj ... 32
5 Démarrage et fin de l'application ... 32
5.1 Version 1 ... 32
5.2 Version 2 ... 32

B Spécifications non fonctionnelles 33
1 Contraintes de développement ... 33
1.1 Matériel ... 33
1.2 Logiciels ... 33
1.3 Langage de programmation ... 33
2 Contraintes de fonctionnement ... 33
2.1 Performances ... 33
2.2 Capacités ... 33
2.3 Maintenance et évolution du système ... 34

C Dossier de tests 35

D Gestion de projet 37
1 Méthodologie ... 37
2 Plannings ... 37

E Cahier du développeur 39
1 Mise en place de l'environnement ... 39
1.1 Installation des logiciels requis ... 39
1.2 Exportation d'un projet sous Unity ... 40
1.3 Compilation d'un projet sous Visual Studio ... 41

1.4 Autorisation de permissions ... 41
2 Créer un projet XAML ... 42
3 Génération de la documentation C# ... 42
4 Connexion au Windows Device Portal ... 43
5 Le mode Research ... 43
5.1 Activation/Désactivation ... 43
5.2 Exemples de codes ... 43
6 Description des versions du projet ... 43
6.1 ScannerGleneV0 ... 43
6.2 ScannerGleneColorPixels ... 44
6.3 ScannerGleneV2 ... 44
7 Données d'authentification ... 44

F Guide utilisateur **45**
1 Lancement de l'application ... 45
2 Utilisation de l'application ... 45
3 Visualisation du nuage de points ... 46

Comptes rendus hebdomadaires **47**

Webographie **53**

Bibliographie **54**

Table des figures

1 Introduction

1 Diagramme de Gantt 4
2 Base méthodologique 4

2 Description générale

1 Un SDK pour l'ensemble des applications Windows 10 6
2 Diagramme de cas d'utilisation 7
3 Déploiement de l'application 8

3 État de l'art

1 A gauche, une triangulation de Delaunay. A droite, chaque cercle contient un quatrième point 10
2 Un exemple de fichier OBJ et du modèle 3D résultant 11
3 Les différents référentiels sous Unity 11
4 Un exemple de caméra de profondeur 12
5 Principe des caméras ToF à décalage de phase 13
6 Principe des caméras à lumière structurée 14
7 Un exemple de lentilles plénoptiques 15
8 Les composants des lunettes HoloLens 16
9 Le Device Portal 17
10 Le geste de floraison (ou bloom gesture) pour afficher le menu 18
11 Le air tap pour sélectionner des applications et des hologrammes 18

4 Analyse et conception

1 Accès aux caméras de localisation des lunettes 20

2 Exécution de l'algorithme Canny transmis sur ordinateur 21
3 Exécution de l'algorithme Canny transmis directement sur lunettes 21
4 Fonctionnement de l'application 22
5 Fonctionnement de notre application 23
6 Diagramme de classes de notre application 23
7 Processus simplifié de récupération des images du flux vidéo 24
8 Balise XAML 25
9 Image de l'interface XAML accessible 25
10 Visualisation de la profondeur 25

D Gestion de projet

1 Le cycle en V 37
2 Diagramme de Gantt - PRD1 38
3 Diagramme de Gantt prévisionnel - PRD2 38
4 Diagramme de Gantt réellement effectué - PRD2 38

E Cahier du développeur

1 Détails de l'installation 39
2 Configuration 40
3 Inpector 40
4 Configuration des paramètres 41
5 Configuration des paramètres 41
6 Configuration de Visual Studio 41
7 Les packages à importer 41
8 Les permissions à accorder 42

F Guide utilisateur

1 Permission d'accès à la caméra 45

Remerciements

J'adresse mes remerciements à mes encadrants, messieurs Jordan NICOT, Christian PROUST et Mohand SLIMANE pour leur confiance, le temps qu'ils m'ont accordé et leur suivi tout au long des semestres. Leur investissement est une grande source de motivation.

Je remercie monsieur Jean-Yves RAMEL pour le temps qu'il a pris à m'aider dans la rédaction de mes spécifications.

Enfin, je remercie toutes les personnes qui ont contribué de quelque façon à la réalisation de ce projet.

1 Introduction

1 Acteurs, enjeux et contexte

1.1 Contexte du projet

Depuis près de cinq ans, plusieurs étapes ont été abordées afin de tendre vers l'objectif final d'une assistance 3D lors d'une opération chirurgicale avec des lunettes de réalité augmentée. Dans un premier temps il y a eu la reconstitution du modèle 3D de la scapula pathologique à partir d'images de scanner. Ensuite il y a eu un travail d'analyse de données, puis un travail de reconstruction 3D à partir d'un nuage de points, afin de reconstruire, en 3D, la partie disparue de la glène de la scapula pathologique, en fonction d'autres parties intactes de la scapula. Concernant la dernière étape avec des lunettes 3D, un premier travail exploratoire, en 2014, a dressé un état de l'art des lunettes électroniques 3D. Un premier prototype a été réalisé (en 2014-2015, 2015-2016) sur des lunettes BT200 de Epson, avec le logiciel Creator de Metaio.[4] [3] [2] [5] [1]

1.2 Enjeux

Ce projet de recherche fait partie du grand projet *Chirurgie de l'épaule* initié par Polytech Tours en collaboration avec le CHU Trousseau et plus particulièrement avec le Docteur Julien Berhouet. L'enjeu principal est de pouvoir démontrer la faisabilité du projet et de réussir à assister le chirurgien dans son opération.

1.3 Acteurs

Le client est le CHRU Trousseau, précisément l'équipe dirigée par le Docteur Julien Berhouet. Le maître d'oeuvre est formé de Polytech Tours et de Grégoire MARCO. Ce dernier est également l'auteur de ce rapport.

2 Objectifs

2.1 Objectif général

Le chirurgien dispose des lunettes connectées HoloLens auxquelles est transmis un fichier 3D contenant l'intégralité de l'image de l'omoplate (la partie encore existante - de couleur verte - et celle manquante reconstituée mathématiquement - de couleur rouge). Le projet consiste à permettre une superposition "parfaite" entre cette image et l'omoplate réelle, dont seule une extrémité, appelée glène, est visible lors de l'intervention chirurgicale. Nous intervenons donc dans le domaine de la Réalité Augmentée et non pas dans celui de la Réalité Virtuelle.[WWW2] [6]

2.2 Les états de l'art

Pour atteindre l'objectif général, je vais dans un premier temps fournir plusieurs états de l'art. Les premiers concerneront les lunettes HoloLens (ses composants matériels et ses caractéristiques techniques) ainsi que les modèles 3D de façon générale et comment ils sont utilisés avec les lunettes.

Enfin, un deuxième état de l'art concernera les caméras de profondeur. Une caméra de ce type permet d'évaluer la distance la séparant d'un objet et est donc en fait utilisée par les lunettes HoloLens.

2.3 Récupération d'un nuage de points

Dans un second temps, mon objectif sera d'étudier comment récupérer le nuage de points d'un objet 3D à travers les lunettes HoloLens. Je dois pour cela prendre en main les lunettes, et plus particulièrement le mode *Research*. Son fonctionnement est expliqué plus loin dans ce rapport mais, brièvement, ce mode permet d'accéder aux données brutes des caméras des lunettes.

Pour réaliser cet objectif, j'ai à ma disposition un modèle d'omoplate. Je commencerai toutefois mes tests sur un objet un peu moins complexe.

Tout ceci afin de répondre à la question fondamentale suivante : nous souhaitons que le matching entre la glène réelle et la glène virtuelle soit réalisée de façon extrêmement précise, et ce même lorsque le chirurgien tourne autour du patient. L'image récupérée de la glène faisant office de point de repère pour placer les éléments virtuels, nous ne devons donc pas avoir plus d'un millimètre de décalage.

3 Hypothèses

Nous souhaitons savoir s'il est possible de récupérer avec les lunettes HoloLens le nuage de points correspondant à un objet visualisé. Il semblerait que Microsoft ait récemment ouvert aux développeurs un mode *Research* permettant d'accéder aux flux de données brutes de chaque caméra. Nous faisons donc pour le moment l'hypothèse que cela réponde aux objectifs souhaités.

S'il s'avère que ce n'est pas le cas, il faudra étudier la possibilité de coupler les lunettes avec un autre matériel pour le faire.

Nous émettons également l'hypothèse que la scène (l'épaule du patient à opérer) n'est pas en mouvement. C'est un fait important à prendre en compte pour la reconnaissance des points caractéristiques de la scène.

4 Bases méthodologiques

4.1 Gestion de projet

Pour gérer efficacement mon projet et ne pas me perdre dans mes différents travaux, j'ai réalisé un diagramme de Gantt grâce au logiciel MindView. Cela me permet de d'avoir des objectifs bien définis, de visualiser l'intégralité du projet mais aussi de réviser facilement mon planning si nécessaire.

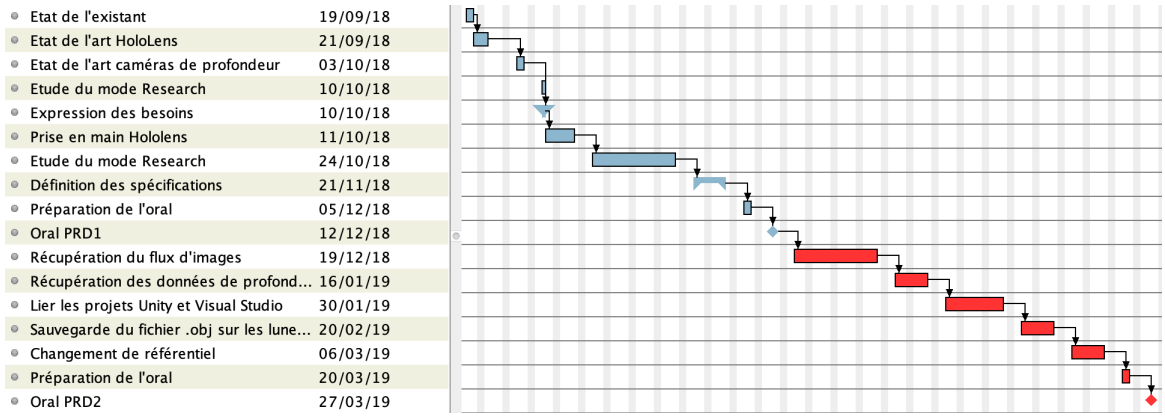


Figure 1 – *Diagramme de Gantt*

Des comptes rendus et mon rapport sont à remettre chaque semaine à mes encadrants qui peuvent évaluer en conséquence mon avancement.

Je n'utiliserai pas d'outils de gestion de versions comme *Git* par exemple, puisque je travaille seul sur ce projet. En revanche, et sur les conseils de mon tuteur Jordan NICOT, je peux mettre en place une gestion de versions en mettant à jour le nom du projet lors de mises à jour importantes (il ne sera ainsi pas écrasé à chaque fois).

4.2 Démarche

Le projet se centre autour de l'ouverture du mode *Research* aux développeurs et des applications possibles. L'objectif fixé est donc de pouvoir récupérer l'image de la glène réelle sous la forme d'un nuage de points grâce aux lunettes. Cette démarche se caractérise en trois étapes :

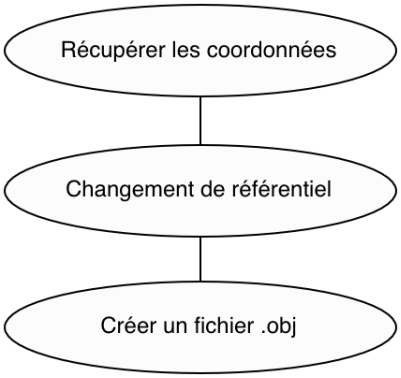


Figure 2 – *Base méthodologique*

1. D'abord, nous devons récupérer les coordonnées (x, y, z) des points de l'image obtenue par les lunettes. La valeur de la profondeur se calcule grâce aux données fournies par la caméra de profondeur,

2. Ces coordonnées sont cependant relatives à la caméra, alors que nous souhaitons obtenir les points dans le référentiel de la scène,

3. Enfin, nous créerons un fichier 3D qui nous permettra de visualiser le modèle obtenu. Nous pourrons, par exemple, utiliser le logiciel Blender.

2 Description générale

1 Environnement du projet

Depuis maintenant deux ans, le projet est axé sur la partie réalité augmentée, à savoir superposer l'image virtuelle d'une omoplate à la réalité.

Pour cela, nous utilisons les lunettes HoloLens, dont les caractéristiques sont décrites dans la suite de ce rapport. Nous utilisons les mêmes logiciels que ceux des années précédentes, à savoir :

— **Unity 2017.4.12f1 LTS**, un moteur de jeu multiplateforme gérant la 3D. Nous utilisons une version **LTS** (*Long Term Support* ou en français *Support sur le Long Terme*) qui assure une stabilité du logiciel pendant 2 ans.

— **Visual Studio 2017**, un logiciel de développement Windows. Pour notre projet, nous importons le kit de développement de jeux avec Unity.

Windows 10 offre la possibilité de créer une application pouvant être exécutée sur tous les appareils exécutant le système d'exploitation. On parle d'applications de Plateforme Windows Universel, abbrégé UWP pour *Universal Windows Platform* en anglais.

Il faut préalablement télécharger le SDK disponible, la version 10.0.17134.0 dans le cas de notre projet, qui contient toutes les bibliothèques et outils indispensables pour le développement des applications.

Figure 1 – *Un SDK pour l'ensemble des applications Windows 10*

Nous pouvons aussi développer notre application grâce à la librairie OpenCV qui permet le traitement d'images en temps réel. Elle est disponible sous Unity à travers une extension payante, d'environ 85€. En revanche, il s'avère qu'elle est compatible pour les applications UWP et est proposée sur de nombreux forums.

Sinon, il faut développer notre propre librairie. En effet, OpenCV utilise les langages C et C++, tandis que Unity fonctionne avec des scripts C#. Il faudrait donc développer notre propre surcouche (ou *wrapper*) C#, qui reprendrait les fonctionnalités d'OpenCV. Cela serait en revanche une perte de temps non nécessaire. C'est pourquoi j'ai proposé qu'il serait préférable d'investir dans un module payant, mais stable et maintenu par des développeurs. C'est la décision que nous avons prise.

Un point fort intéressant est qu'il existe de nombreux exemples d'application de la librairie, aussi bien sur Unity que sur les lunettes HoloLens, que l'on peut trouver gratuitement sur différents dépôts Github.

En terme de matériel, une maquette d'épaule ainsi qu'un modèle 3D d'omoplate sont mis à disposition pour réaliser des tests.

2 Caractéristiques des utilisateurs

On distingue deux utilisateurs différents :

— Le **chirurgien** qui doit être totalement autonome avec l'application.

— Le **technicien** possédant les compétences informatiques nécessaires pour maintenir l'application et/ou la mettre à jour.

3 Fonctionnalités du système

Voici le diagramme de cas d'utilisation :

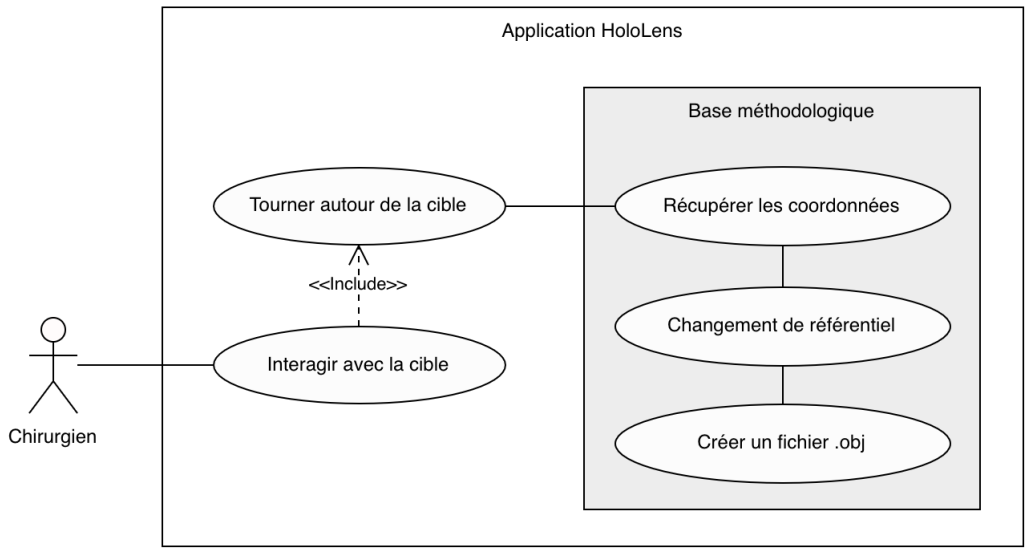


Figure 2 – *Diagramme de cas d'utilisation*

4 Structure générale du système

Au final, notre application pourra être compilée sous Unity pour UWP. Visual Studio nous permettra d'importer la solution obtenue dans les lunettes HoloLens par câble USB ou par WiFi.

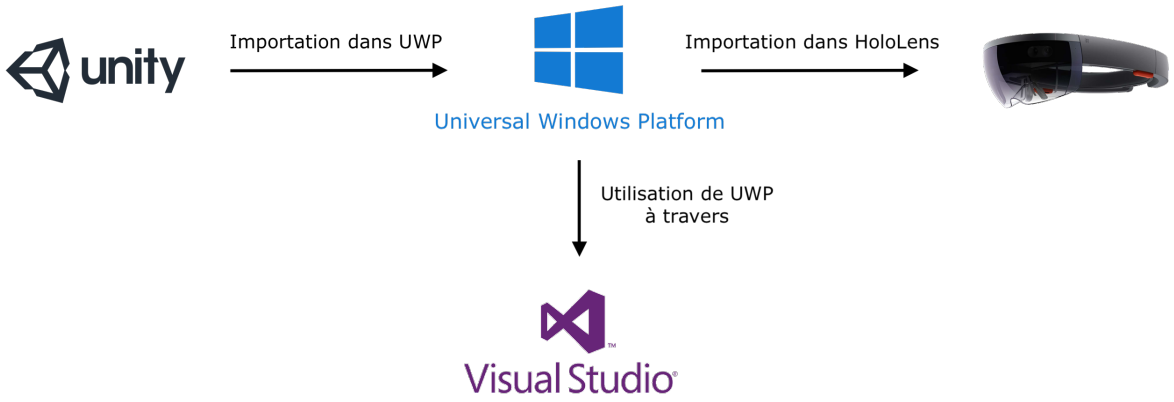


Figure 3 – *Déploiement de l'application*

3 État de l'art

1 Les modèles 3D

1.1 Définition d'une image 3D

Un image 3D est un modèle permettant de simuler une représentation d'un objet dans un espace tridimensionnel. Cette image peut être composée de polygones simples ou de formes simples (cube, sphère,...) eux-même définis par des points, des arêtes et des faces, ou plus généralement seulement de points. On parle alors d'un nuage de points.

Les surfaces des modèles 3D sont définies à l'aide de *shaders*, des composants qui caractérisent la couleur, l'émission de la lumière ou encore la texture globale du modèle.

1.2 Définition d'une vidéo 3D

Une vidéo est une succession d'images respectant une certaine cadence. Une vidéo 3D est donc, par définition, une succession d'images 3D.

1.3 Obtention d'un nuage de points

Pour récupérer un objet sous forme d'un nuage de points, on utilise un ensemble de techniques appelées *range imaging*, que l'on pourrait traduire par *imagerie de profondeur*. Elles ont toutes pour objectif de mesurer en temps réel une scène dans l'espace et de récupérer la distance qui sépare un utilisateur, ou plus généralement un point de référence, d'un objet, sans utiliser d'algorithme de reconnaissance de formes. Les caméras utilisant ces techniques sont appelées *caméras de profondeur*.

1.4 Reconstruction des surfaces

A partir du nuage de points obtenu, il faut désormais réussir à reconstruire virtuellement l'objet que l'on souhaite reconnaître ultérieurement. Pour cela, la plupart des algorithmes utilisent

la triangulation de Delaunay [WWW1], nommée d'après le mathématicien russe Boris Delone (1890-1980) dont le nom a été francisé en Delaunay. Comme son nom l'indique, elle consiste simplement à lier les points entre eux pour former des triangles.

La triangulation de Delaunay se définit comme tel : pour un ensemble de n points, elle est l'unique triangulation telle qu'un cercle circonscrit à un triangle x ne contienne aucun autre sommet que ceux de ce triangle. Une extension de ce principe, dans l'espace à 3 dimensions, a conduit à l'algorithme dit "Point Cloud Skinner", extension du logiciel Blender (utilisé pour la modélisation 3D).[WWW4] [WWW3]

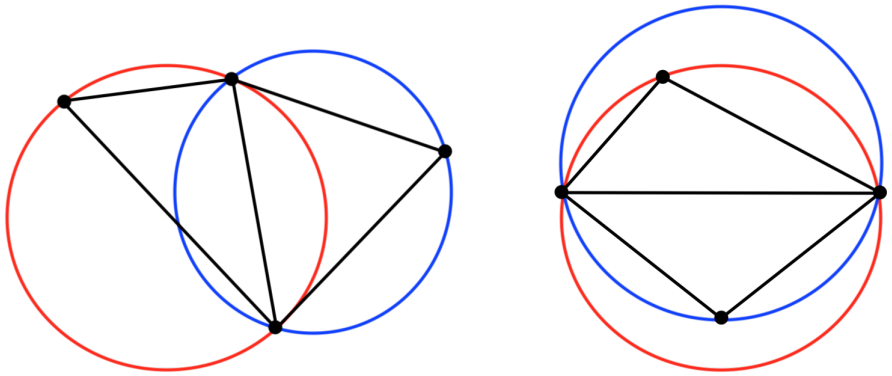


Figure 1 – *A gauche, une triangulation de Delaunay. A droite, chaque cercle contient un quatrième point*

1.5 Les fichiers de scène 3D

La représentation des objets créés par les logiciels de modélisation 3D nécessite un format de données particulier. En effet, il faut pouvoir stocker les informations relatives à sa forme, sa taille, ses textures, etc.

Il existe de nombreux formats de fichiers permettant de sauvegarder ces informations, mais plusieurs avantages tendent en faveur des fichiers OBJ.

Premièrement, ils sont très communs, ce qui signifie qu'ils sont supportés par beaucoup de logiciels, si ce n'est tous. Ils sont également très facile d'utilisation : ils sont notamment écrits au format ASCII et sont relativement compréhensibles puisqu'ils ne gèrent que des polygones. Ces deux spécificités font que les fichiers OBJ sont totalement lisibles par un être humain,

Il est enfin possible d'y sauvegarder des données relatives aux couleurs et/ou à la lumière et autorisent toutes sortes de calcul et de manipulations des données de base.

Chaque ligne définissant l'objet est identifiée par un mot-clé suivi des données associées. On retrouve principalement :

— Les points (ou *vertices)*, identifiés par ***v*** *x y z* (x, y, et z étant les coordonnées du point dans l'espace),

— Les faces, identifiées par ***f*** *i j k* (i, j et k sont les indices des points décrits précédemment). On retrouve ici la définition des faces sous forme triangulaire.

— Les normales, identifiées par ***vn*** *x y z*, qui permettent de définir l'orientation de l'objet dans l'espace.

— Les textures associées aux points, identifiées par ***vt*** *x y z*

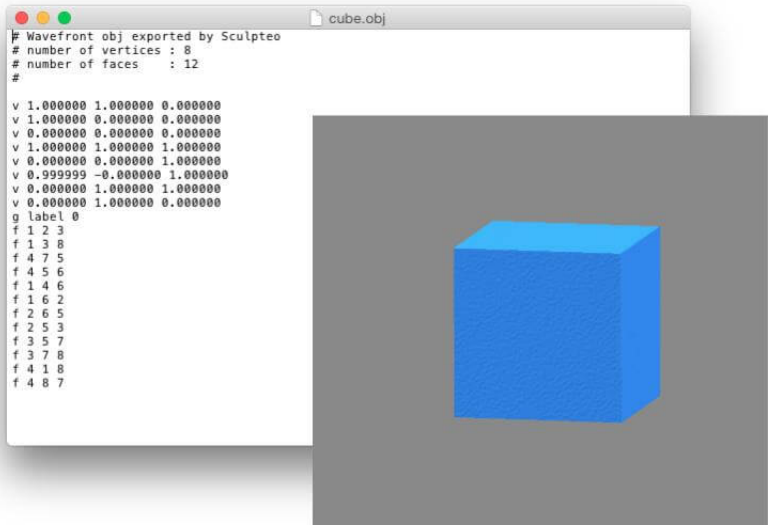


Figure 2 – *Un exemple de fichier OBJ et du modèle 3D résultant*

1.6 Les référentiels de visualisation

Unity utilise trois types de référentiels pour représenter un objet dans l'espace.

— Le référentiel de la scène (ou world space) forme l'espace dans lequel les cibles *vivent*. Il s'agit d'un système de coordonnées en trois dimensions possiblement infini en terme de taille. Dans le cadre de notre projet, il s'agit de la salle d'opération,

— Le référentiel de l'écran (ou screen space) forme l'espace délimité par la taille de l'écran. Il possède donc une hauteur et une largeur qui sont définies par un nombre de pixels. C'est un référentiel en deux dimensions. Dans notre projet, il s'agit de la caméra des lunettes HoloLens,

— Le référentiel local (ou viewport space) est relatif à la caméra, c'est-à-dire que la largeur et la hauteur de l'écran sont d'unité 1.

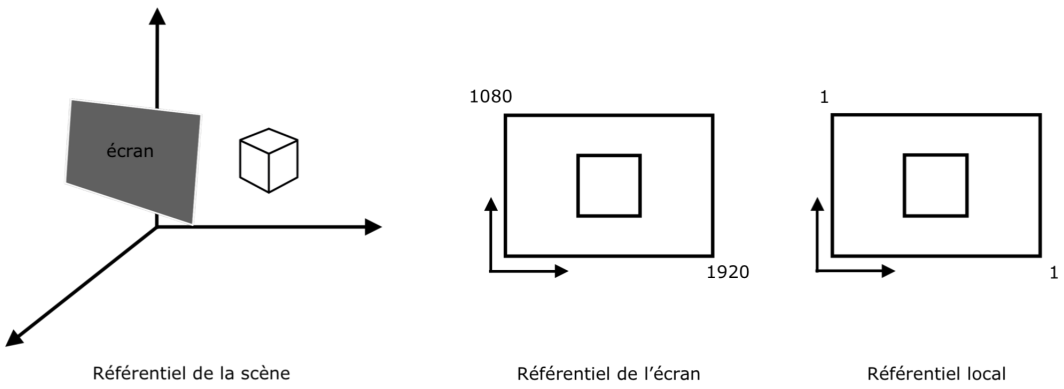


Figure 3 – *Les différents référentiels sous Unity*

Nous nous concentrerons sur les deux premiers référentiels. Nous aurons peut-être besoin de mettre en place des marqueurs afin de reconnaître correctement la cible.

2 Les caméras de profondeur

Elles sont réparties en deux catégories : les caméras *actives* qui émettent volontairement un signal lumineux, et les caméras *passives* qui utilisent l'intensité lumineuse déjà présente.

2.1 Les caméras *actives*

Ces caméras possèdent deux composantes fondamentales : un système permettant d'illuminer l'espace ainsi qu'un système analysant l'image de la scène éclairée.

L'utilisation de ce type de caméra suppose tout d'abord que chaque point illuminé réfléchisse une partie de la lumière reçue vers le centre optique de la caméra. Le type de diffusion joue donc un rôle important dans l'estimation de cette profondeur.

2.1.1 Temps de vol (en anglais *time of flight*, abbrégé ToF)

Nous nous intéresserons, dans notre cas, plus particulièrement aux caméras ToF, puisqu'il s'agit de la technologie embarquée par les lunettes HoloLens. Il en existe deux types : à impulsion et à décalage de phase.

ToF à impulsion

Le principe consiste à utiliser la vitesse de la lumière pour calculer la distance entre la caméra et un objet.

Figure 4 – *Un exemple de caméra de profondeur*

1. D'abord, une impulsion lumineuse infrarouge est émise par une LED. Les rayons infrarouges sont utilisés seulement pour ne pas gêner l'utilisateur.

2. La lumière réfléchie est ensuite captée par une optique, qui la focalise vers le capteur ToF (3) contenu à l'intérieur de la caméra.

3. Chaque pixel de ce capteur réalise indépendamment le calcul de cette distance, qui sera donc différent pour chacun. Étant donné que la célérité de la lumière est connu, il suffit simplement de calculer le temps de trajet de l'impulsion pour obtenir la distance.

Cette technique est extrêmement intéressante en terme de coût de temps puisqu'il n'y a pas besoin de mettre en place un système de balayage.

En revanche, on peut noter quelques inconvénients :

— Le premier point négatif est l'influence de la lumière ambiante, dans notre cas, il s'agirait de celle de la salle d'opération, sur le calcul de la profondeur. Elle peut en effet interférer avec le signal réfléchi et fausser la mesure.

— La quantité de rayons infrarouges est limitée et la qualité du signal réfléchi peut être faible. Pour l'améliorer, les capteurs possèdent des pixels de taille plus importante, ce qui diminue la résolution de l'image obtenue.

ToF par décalage de phase

Ces caméras s'appuient cette fois-ci sur le décalage de phase (noté $\Delta\phi$ sur l'image) entre le rayon émis et le rayon réfléchi. Sur le schéma, f est la fréquence du signal.

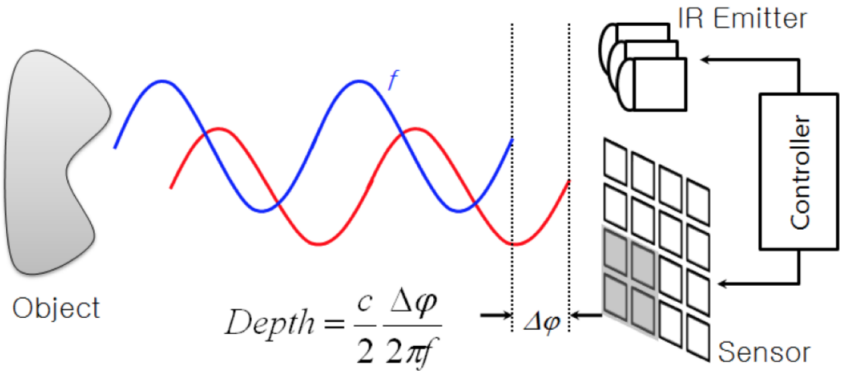


Figure 5 – *Principe des caméras ToF à décalage de phase*

Une caméra de profondeur basée sur le principe ToF expliqué ici nécessite une résolution de l'ordre de la picoseconde (soit 10^{-12} secondes) dans ses mesures de temps pour atteindre une résolution de profondeur de l'ordre du millimètre. C'est pour cette raison que beaucoup de technologies utilisent le principe ToF par décalage de phase.

Elles ont pour autres avantages d'être relativement plus rapides et précises que les caméras ToF. En revanche, elles ont une portée plus réduite.

2.1.2 Lumière structurée

Ces caméras projettent un motif lumineux (un point, une ligne ou une image généralement faite de bandes) sur l'objet cible et en observe sa déformation grâce à un algorithme qui calcule la profondeur.

Le point fort de cette technique est sa rapidité, puisque l'intégralité du champ de vision est analysée. Cela limite ainsi, voire élimine, les problèmes de distorsion liés aux mouvements que l'on peut retrouver sur les caméras ToF.

En revanche, des problèmes d'ambiguïté peuvent apparaître. En effet, selon la complexité des discontinuités de l'objet cible, rien ne certifie que les différentes primitives (point, ligne, etc...) du motif projeté conserveront leur position initiale. Cela entraîne une identification plus difficile. Pour cela, il faut mettre en place un codage qui, pour chaque primitive, associe un code unique permettant une mise en correspondance.

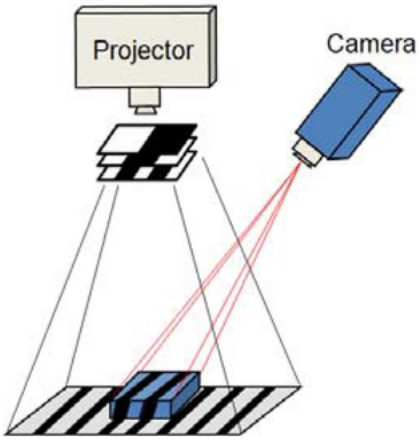


Figure 6 – *Principe des caméras à lumière structurée*

2.2 Les caméras *passives*

2.2.1 Stéréoscopie

Cette technique consiste à reproduire la vision humaine grâce à l'utilisation de deux caméras. Deux images sont donc récupérées simultanément et il est possible d'en extraire les informations de profondeur grâce aux couleurs, aux contours, etc. Cela nécessite donc un algorithme de reconnaissance d'images supplémentaire.

On peut également ajouter trois autres contraintes importantes :

— En général, l'estimation de profondeur est plus précise pour un objet proche que l'inverse. Cependant, cette précision est perdue si sa distance par rapport aux caméras est trop petite. En effet, cela aurait pour conséquence qu'il ne soit visible que par une seule des deux caméras.

— Selon la façon dont l'objet se situe dans l'espace, un point peut être présent sur la première caméra mais pas sur la seconde. Dans ce cas là, aucune correspondance ne peut être réalisée.

— La scène doit posséder des points caractéristiques fortement reconnaissables pour que la correspondance soit réalisée.

2.2.2 Les caméras plénoptiques

Une caméra plénoptique est composée de microlentilles, situées entre une lentille principale et un capteur, qui sont constituées d'un certain nombre de pixels. Quand la scène est récupérée par chacune des microlentilles, les pixels de l'image plénoptique obtenue sont donc la scène elle-même sous des angles de vue différents.

L'un des principaux inconvénients concerne la perte de résolution lors de la création d'une image plénoptique. En effet, les pixels de chaque microlentille sont pris ensemble pour déterminer l'intensité de cette partie de l'image.

Il est toutefois possible de contourner ce problème en modifiant la taille de l'ouverture de l'objectif : plus la lumière sera captée et plus l'image sera de meilleure qualité. Mais cela

Figure 7 – *Un exemple de lentilles plénoptiques*

provoque un nouveau problème de temps d'acquisition et donc, par conséquent, un problème pour effectuer des mouvements.

2.3 Récapitulatif et conclusion

	ToF à impulsion	ToF à décalage de phase	Lumière structurée	Stéréoscopie	Caméra plénoptique
Type	Actif	Actif	Actif	Passif	Passif
Coût d'acquisition de la scène	Très faible	Très faible	Très faible	Information non trouvée	Dépend de la taille de l'ouverture de l'objectif
Avantages	Rapidité, précision	Avantages ToF, précision accrue	Peu de problèmes de distorsion	Utilisation de la lumière ambiante = pas d'interférences	Qualité des images
Inconvénients	Interférence de la lumière ambiante, résolution faible	Portée faible	Correspondance des points, taille	Contrainte de distance, correspondance des points	Perte de résolution
Principaux domaines d'application	Cartographie, médical, architecture, archéologie	Cartographie, médical, architecture, archéologie	Archéologie, reconnaissance faciale	Information non trouvée	Photographie, Cinéma
Fourchette de prix	86€ - 10400€	86€ - 10400€	2300€ - 12000€	350€ - 680€	150€ - 780€

Cet état de l'art nous a permis de comprendre pourquoi les lunettes HoloLens embarquent une caméra de profondeur de type ToF dite *active*. En effet, les technologies *actives* apportent un avantage considérable en terme d'acquisition de la scène. Les caméras *passives* manquent de précision pour l'objectif souhaité et ne sont clairement pas adaptées à nos besoins. Pour autant, nous ignorons s'il s'agit de TOF à impulsion ou à décalage de phase.

3 Les lunettes HoloLens

3.1 Les composants matériels

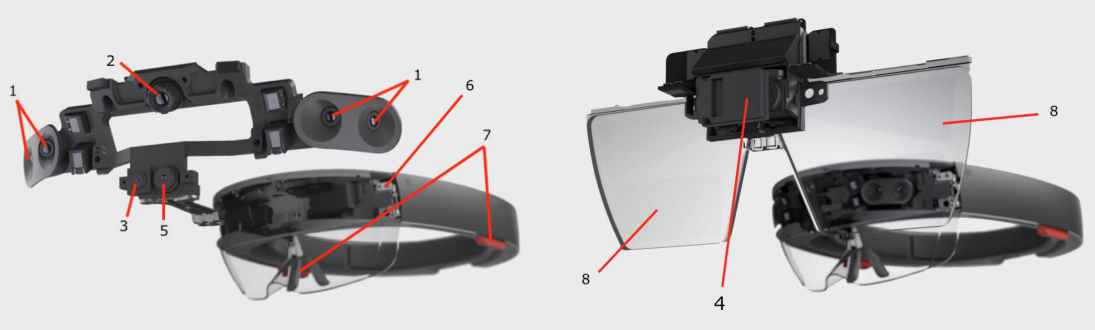


Figure 8 – *Les composants des lunettes HoloLens*

1. **4 caméras** (deux de chaque côté des lunettes) utilisées pour la localisation. Elles permettent ainsi un suivi de l'environnement ainsi qu'une détection de l'orientation de l'appareil.

2. **1 caméra de profondeur** de type *temps de vol*.

3. **1 capteur de lumière ambiante** permettant de capturer l'intensité lumineuse de l'environnement.

4. **1 caméra centrale inertielle** composée de multiple capteurs tels un gyroscope, un accéléromètre ou encore un magnétomètre.

5. **1 caméra vidéo** de deux mégapixels pour effectuer des enregistrements, hologrammes inclus.

6. **4 microphones internes**.

7. 2 **hauts-parleurs** situés au-dessus des oreilles et offrant un son spatial.

8. **Des guides d'ondes**, ou *waveguides*, permettant de combiner les images virtuelles au réel. Ils forment un système permettant de guider les ondes électromagnétiques, comme la lumière, dans un milieu spécifique.

Les lunettes possède enfin **1 CPU, 1 GPU et 1 HPU**. Les deux premiers gèrent l'interface générale des lunettes tandis que le HPU (Holographic Processing Unit) gère quasiment tous les traitements de l'environnement et les gestes de l'utilisateur.

3.2 Les caractéristiques techniques

L'HoloLens dispose du logiciel Windows Holographic, intégré à Windows 10. Les lunettes disposent d'une autonomie d'environ 2-3 heures et, par conséquent, n'ont pas besoin de connexion

filaire à un ordinateur pour fonctionner. En terme de connectivité, on peut noter la présence du Wifi, du Bluetooth et d'un port micro USB 2.0.

En revanche, les lunettes possèdent un champ de vision assez restreint de 35° (contre environ 180° pour un oeil humain).

3.3 Le Windows Device Portal

Le *Windows Device Portal* est un outil qui permet de gérer un appareil à distance, par l'intermédiaire d'un câble USB ou du réseau. Il s'agit d'un serveur web sur l'appareil auquel on peut accéder grâce à un navigateur web. Dans notre cas, il offre une liste de différentes fonctionnalités spécifiques qui nous permettent de gérer les lunettes.

En plus d'une page d'accueil qui présente un récapitulatif de l'état des lunettes (l'état, le nom, la version de Windows installée), on trouve :

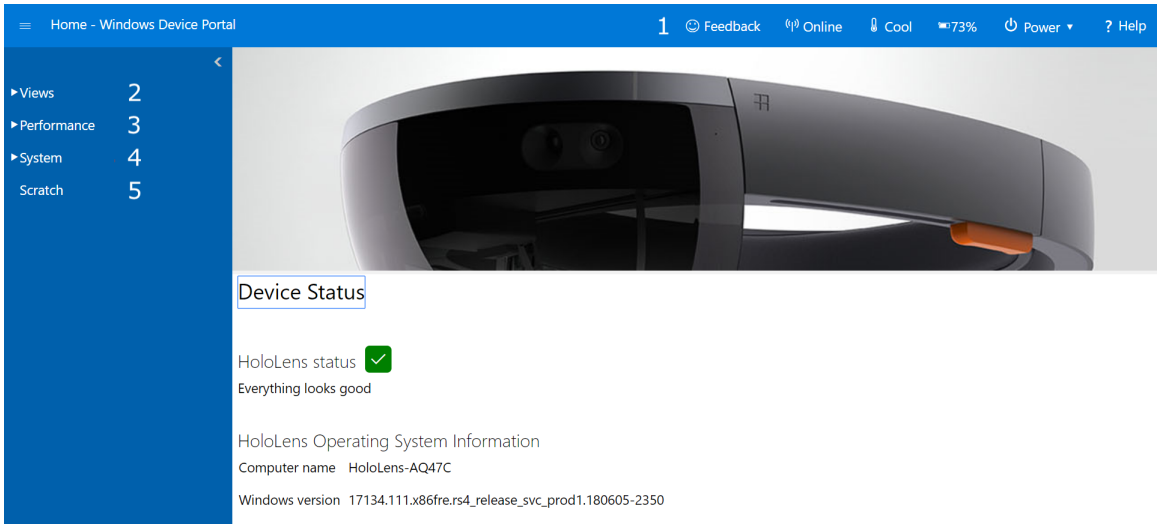


Figure 9 – *Le Device Portal*

1. **Une barre d'outils** affichant les différents états des lunettes (niveau de batterie, température, ...) et des actions rapides comme *Eteindre* ou *Redémarrer*,

2. La page **Views** concerne les différents flux des lunettes : on peut ainsi cartographier l'espace dans lequel on évolue, visualiser en direct ce que l'on peut voir à travers les lunettes ou encore faire des enregistrements vidéos,

3. La Page **Performances** affiche les performances mémoire, CPU, réseau ou encore les processus en cours d'exécution,

4. La Page **System** permet de gérer les fonctionnalités du système (le choix d'un réseau, visualiser les applications installées ou activer le fameux mode *Research* dont nous parlerons plus tard),

5. La page **Scratch** permet de créer un ou plusieurs espaces de travail personnalisés auquel on peut ajouter les fonctionnalités que l'on souhaite.

3.4 La gestuelle et la reconnaissance vocale

Les capteurs des lunettes HoloLens offrent à l'utilisateur la possibilité d'utiliser les applications Windows à travers des gestes et la voix. Cependant, cela demande quelques contraintes.

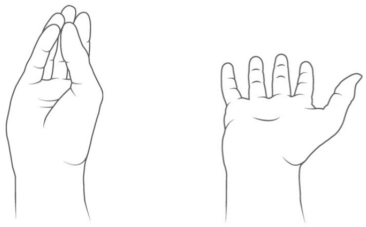


Figure 10 – *Le geste de floraison (ou bloom gesture) pour afficher le menu*

La gestuelle, tout d'abord, n'est prise en compte que si les mains se situent dans une zone spécifique, dû au champ de vision restreint. Si l'on souhaite sélectionner une application, il faut préalablement fixer celle qui nous intéresse avant d'effectuer le geste requis.

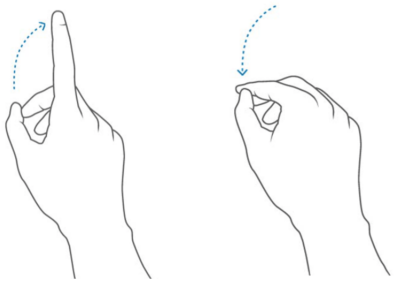


Figure 11 – *Le air tap pour sélectionner des applications et des hologrammes*

Ces gestes peuvent être remplacés par des commandes vocales simples (Select, Place, Bigger, ...), toutefois seulement en anglais pour le moment. Puisqu'il s'agit d'un produit Microsoft, l'HoloLens embarque l'assistant Cortana à qui l'utilisateur peut demander d'effectuer certaines actions.

4 Analyse et conception

1 Prise en main des lunettes HoloLens

La tout premier travail que j'ai réalisé a été de prendre en main les lunettes. J'ai pour cela suivi plusieurs tutoriels, disponibles sur le site de Microsoft, qui proposent de mettre en oeuvre plusieurs applications couvrant une grande partie des possibilités offertes par les lunettes.[WWW5]

L'environnement est très différent de ce qu'on a l'habitude de manipuler, étant donné que l'interaction avec les applications se fait principalement grâce à des gestes, ou éventuellement grâce à la voix. Cela demande donc un certain temps d'adaptation.

En plus du matériel, j'ai également dû prendre en main le logiciel Unity et la façon de programmer qui s'effectue à travers des scripts. Au lieu de simplement récupérer les scripts proposés par les tutoriels, j'ai choisi d'y apporter des modifications afin, d'une part, de comprendre le code écrit, et d'autre part de comprendre quelles actions exécutaient certaines parties des scripts.

Au total, j'ai consacré deux séances entières à la prise en main du matériel. J'ai fait le choix de me former sûrement plus longtemps que nécessaire, mais je souhaitais être à l'aise avec l'environnement pour la réalisation de l'objectif suivant.

2 Découverte du HoloLens Research mode

Le *Research mode* offre depuis Avril 2018 la possibilité aux développeurs d'accéder aux données de différents composants des lunettes HoloLens, notamment :

— Les 4 caméras qui permettent la localisation dans l'espace,

— Les données de la caméra de profondeur sous deux versions différentes : une qui est utilisée plutôt pour le calcul de profondeur d'éléments proches (la détection des mains par exemple), l'autre pour déterminer la profondeur d'éléments plus éloignés (telle que la cartographie de l'espace),

— Deux versions du flux vidéo de réflectivité du rayon infrarouge, utilisé par la caméra pour calculer la profondeur.

En revanche, ce mode ne peut être exploité qu'à des fins de recherches académiques ou industrielles. Il ne peut donc pas être utilisé par des applications qui seront déployées en entreprise ou sur le Microsoft Store. Cela s'explique à la fois parce que la sécurité de l'appareil est diminuée, mais également parce que la consommation de la batterie augmente fortement.

De plus, Microsoft ne s'engage pas à le rendre utilisable sur d'autres appareils. Une application développée n'est donc pas assurée d'être encore opérationnelle sur de futurs produits. Dans notre cas, cela n'a pas d'importance.

L'activation du mode *Research* n'a pas été aussi facile qu'espéré. En effet, il a d'abord fallu mettre à jour les lunettes, dont la dernière mise à jour datée de 2017. Lors du redémarrage des lunettes, le compte Microsoft associé a été déconnecté, ce qui a eu pour conséquence une perte de temps pour retrouver les mots de passe.

Pour voir les effets exacts du mode, j'ai utilisé la fonctionnalité de visualisation en direct, accessible par le *Device Portal*. Les premiers essais n'ont pas été concluants puisque les hologrammes n'étaient visibles que sur un fond vert, l'environnement réel n'apparaissant donc plus.

Microsoft propose un dépôt Github contenant plusieurs projets qui permettent d'utiliser les lunettes comme outil d'analyse d'images. On peut donc coupler ces programmes à la librairie OpenCV.

2.1 Accès aux caméras de localisation

Un premier programme nous permet d'accéder aux 4 caméras permettant aux lunettes de connaître la localisation de l'utilisateur (celles situées de chaque côté) ainsi qu'à la caméra de profondeur.

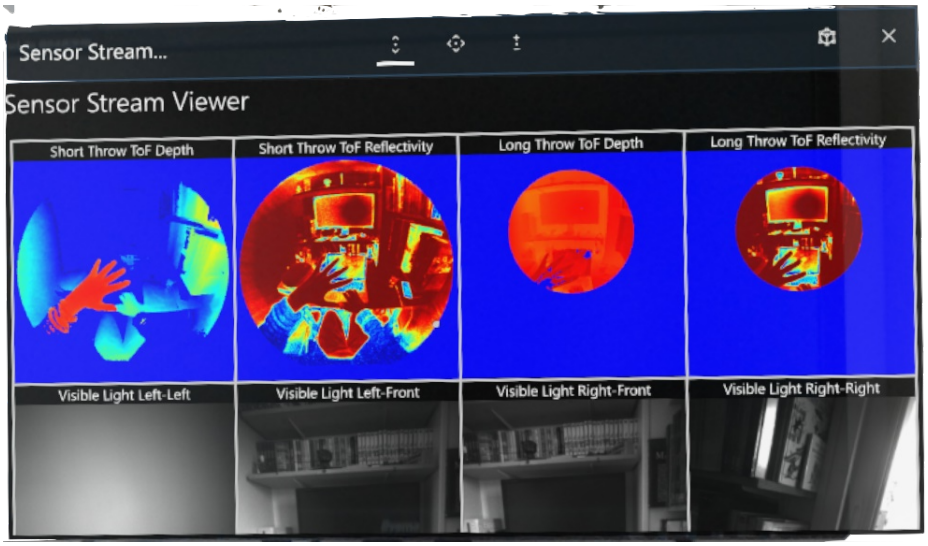


Figure 1 – *Accès aux caméras de localisation des lunettes*

Sur la première ligne, nous pouvons voir le traitement de profondeur (d'éléments proches et éloignés, comme expliqué précédemment, images 1 et 3) ainsi que la réflectivité du rayon infrarouge (sur les éléments proches et éloignés, images 2 et 4).

Sur la seconde ligne, il s'agit des flux bruts des caméras.

2.2 Accès à la caméra d'enregistrement vidéo

Un second programme permet d'accéder à la caméra gérant l'enregistrement de photos et de vidéos. Comme dit précédemment, nous pourrions coupler un programme OpenCV qui

s'exécuterait soit directement sur les lunettes, soit sur un ordinateur après avoir récupéré le flux.

Les deux images ci-dessous présentent l'exécution de l'algorithme Canny, permettant la détection de contours, implémenté dans le programme d'exemple.

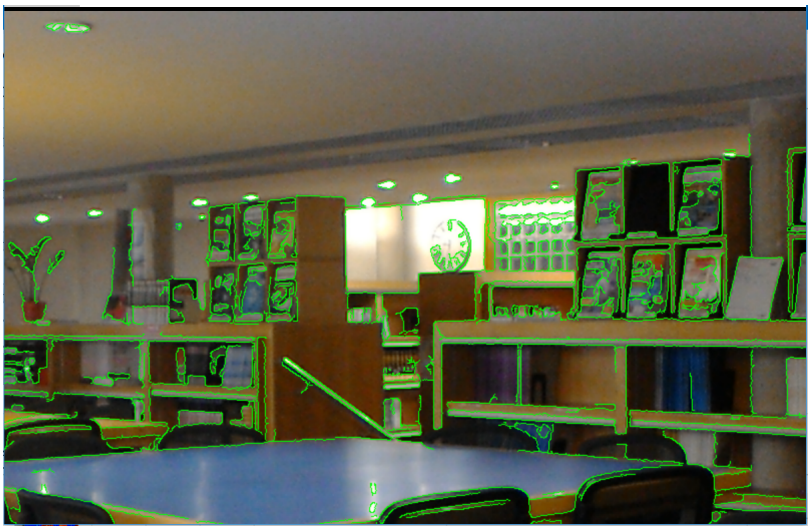


Figure 2 – *Exécution de l'algorithme Canny transmis sur ordinateur*

Les tests du traitement effectué sur ordinateur ne sont toutefois pas aussi bons qu'espéré. Le nombre d'images par seconde est réduit et la fluidité est perdue, voire quasiment inexistante.

Figure 3 – *Exécution de l'algorithme Canny transmis directement sur lunettes*

Dans le cas du traitement réalisé sur les HoloLens, le résultat est beaucoup plus fluide. En plus de cela, il est plus intéressant de traiter notre flux directement sur les lunettes.

3 Conception de l'application

3.1 Principe algorithmique

L'accès à la caméra de profondeur correspond exactement à ce que nous recherchons pour récupérer un nuage de points. Seules les données du traitement des objets en distance courte sont intéressantes puisque le chirurgien travaille près de l'épaule du patient. Nous allons donc nous intéresser au code correspondant et déterminer de quelle façon la profondeur est stockée.

Voici le principe de l'algorithme :

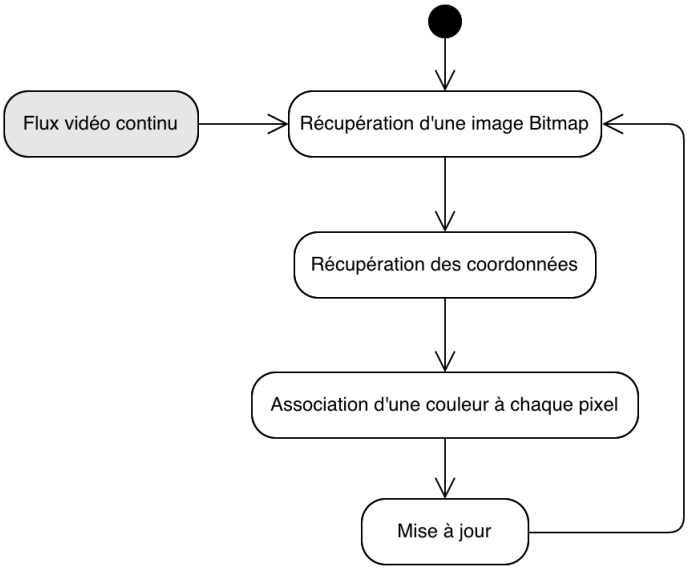


Figure 4 – *Fonctionnement de l'application*

Dans la même logique, nous allons élaborer la même approche pour notre application. Cependant, à l'inverse de cet exemple qui permet d'obtenir une image par point de vue, nous souhaitons dans ce projet obtenir une seule image composée des différents points de vue de l'utilisateur, à la manière d'un scanner. Cela signifie donc que les coordonnées des points obtenus correspondent à celles du référentiel de l'écran et que nous devons les convertir dans le référentiel de la scène.

L'application sera séparée en quatre grandes étapes :

1. Récupération d'images Bitmap du flux vidéo,
2. Récupération des coordonnées *(x, y, z)* de chaque pixel de l'image,
3. Changement du référentiel,
4. Création d'un fichier 3D *.obj*,

Lorsque nous débuterons la phase de développement, il est possible que ce modèle soit retravaillé. En effet :

— D'une part, deux possibilités de traitement nous sont offertes : soit réaliser un traitement des images de façon incrémentale (c'est-à-dire que nous changeons de référentiel à chaque image capturée), soit réaliser le traitement une fois que toutes les images auront été capturées,

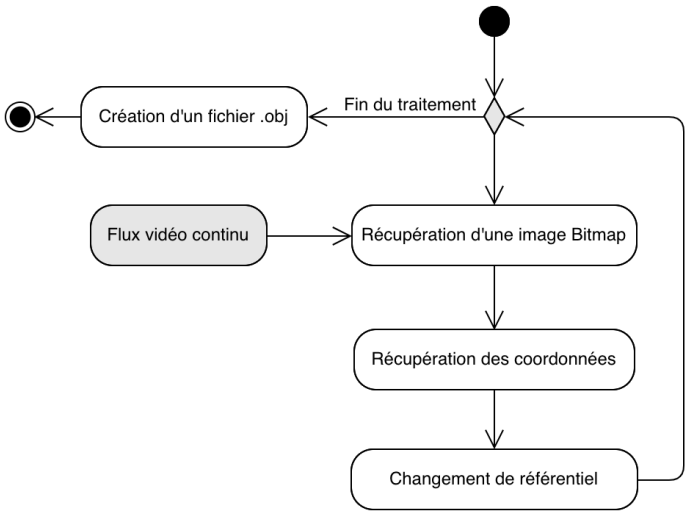


Figure 5 – *Fonctionnement de notre application*

— D'autre part, nous devons définir la condition d'arrêt du traitement (un nombre d'itérations, un temps, etc).

3.2 Modélisation de l'application

Voici le diagramme de classes qui découle de l'analyse précédente :

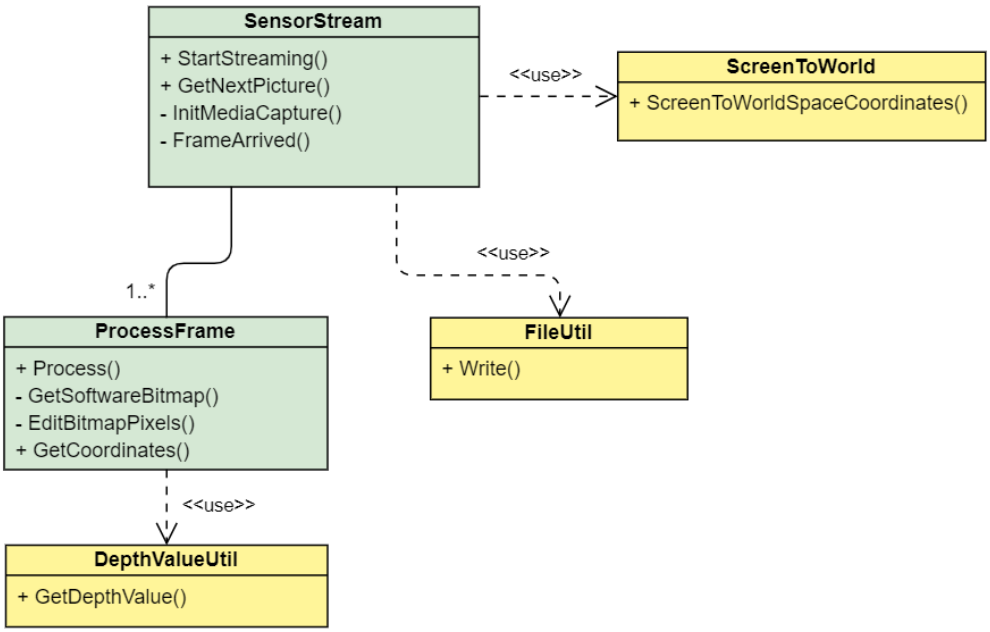


Figure 6 – *Diagramme de classes de notre application*

La signification des classes sera expliquée tout au long de cette partie.

3.3 Première version : récupération et affichage du flux de profondeur

3.3.1 Récupération du flux

Dans un premier temps, le diagramme UML était restreint aux deux classes *SensorStream* et *ProcessFrame* qui permettent respectivement de récupérer le flux de profondeur et de le traiter (un simple affichage du flux dans cette version).

Pour récupérer le flux vidéo de profondeur, il faut suivre les étapes suivantes :

1. Dans un premier temps, il faut récupérer les différents groupes de sources d'images. En effet, les lunettes HoloLens embarquent à la fois un appareil vidéo couleur et une caméra de profondeur. On récupère donc la liste des groupes de sources d'images multimédia pris en charge grâce à l'objet *MediaFrameSourceGroup* et on ne conserve que celui correspondant à la profondeur.

2. Ensuite, il est nécessaire de créer un objet *MediaCapture* qui va permettre d'utiliser le groupe de sources d'images de profondeur. Pour l'initialiser, il faut spécifier plusieurs paramètres :

   — Le groupe de source correspondant,
   — Les images récupérées doivent être disponibles en tant qu'objet pour les manipuler. Il s'agit d'objets *SofwareBitmap*,
   — Le flux audio n'est pas utile, il faut préciser que seule la vidéo est nécessaire,
   — Le flux est partagé avec d'autres applications.

3. Il faut vérifier ensuite que l'encodage de la source d'images correspond bien à celle des images de profondeur,

4. Nous devons créer un lecteur qui va récupérer les images provenant de la source de profondeur. Pour cela, il faut utiliser un gestionnaire d'évènement qui contient le code à exécuter lorsque l'évènement souhaité se produit. Les images capturées sont contenues à l'intérieur dans un objet *VideoMediaFrame* qui fournit un flux vidéo.

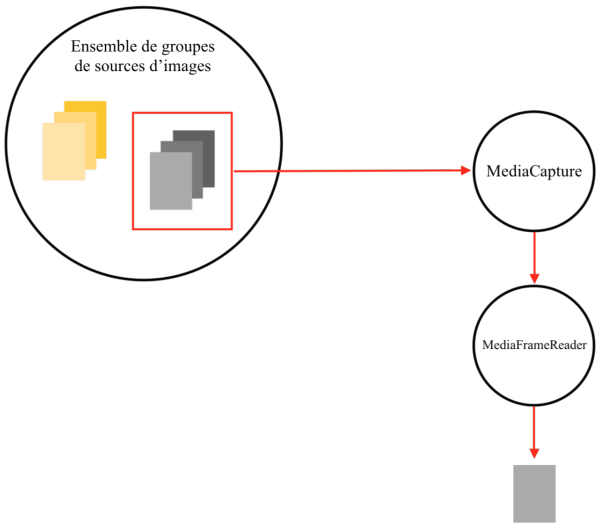


Figure 7 – *Processus simplifié de récupération des images du flux vidéo*

3.3.2 Présentation à l'utilisateur

Comme dit précédemment, les images sont récupérées sous la forme d'un objet *SoftwareBitmap*, récupérables par l'objet *VideoMediaFrame*. Il est alors possible de les afficher dans une interface XAML. Il s'agit d'un langage basé sur le XML qui s'appuie sur l'utilisation de balises pour permettre la création d'éléments graphiques (comme des images, des contrôles ou encore du texte).

A la place de créer un fichier .cs, il faut créer un fichier .xaml. Un fichier .cs lui est associé automatiquement. Pour afficher notre image, il faut créer un composant *Image* dans le fichier .xaml grâce à la balise *Image* :

```
<Grid x:Name="DepthPreviewBlock" BorderThickness="1" Grid.Column="0">
    <StackPanel>
        <Image Name="depthImage" />
    </StackPanel>
</Grid>
```

Figure 8 – *Balise XAML*

Dans le fichier .cs, une méthode *InitializeComponent()* permet de charger l'interface XAML. Il faut rajouter les lignes suivantes :

```
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.Windows.UI.Xaml.Build.Tasks", " 10.0.17.0")]
private global::Windows.UI.Xaml.Controls.Image depthImage;
```

Figure 9 – *Image de l'interface XAML accessible*

pour nous permettre d'utiliser le composant et lui assigner un objet *SoftwareBitmap*. Cette procédure se déroule dans la classe *ProcessFrame*.

L'utilisateur visualise enfin la profondeur détectée.

3.4 Deuxième version : récupération des données de profondeur

Il est nécessaire de sauvegarder les données contenues dans les images Bitmap. Pour chaque pixel, il faut récupérer les coordonnées x, y et la valeur de la profondeur, correspondant à l'axe z de la scène.

Pour montrer que la profondeur est bien récupérée par les lunettes HoloLens, j'ai attribué une couleur spécifique à chaque pixel des images Bitmap obtenues en fonction de la valeur. Ainsi, plus la profondeur calculée est importante, plus la couleur des pixels tend vers le bleu. A l'inverse, elle tend vers le rouge quand la cible est proche de l'utilisateur.

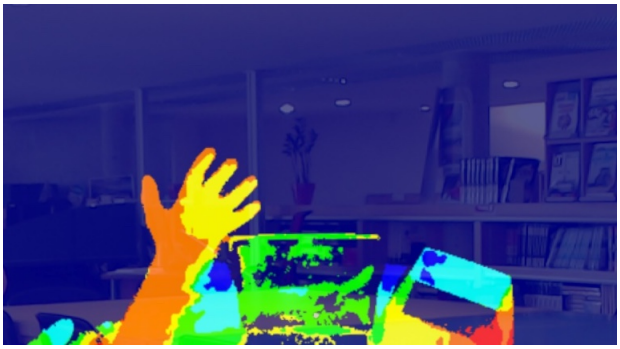


Figure 10 – *Visualisation de la profondeur*

Cette coloration se faisait dans la classe *ColorUtil*, qui n'existe plus dans la version actuelle. Cette classe possède en attribut une palette de couleurs (bleu, vert, jaune, rouge, ...) au format *rgb* mais également sous forme d'entiers puisque l'image que nous manipulons est composée d'octets (si nous souhaitons modifier en la couleur, il faut en modifier la valeur).

Pour chaque pixel, j'utilise ensuite une formule appliquée à la valeur de l'octet pour obtenir la valeur de la profondeur ainsi qu'une simple échelle pour attribuer la couleur.

3.5 Version finale

3.5.1 Changement du référentiel

Dans cette version, je ne crée plus d'interface XAML et j'utilise des fonctionnalités de Unity, ce qui ajoute quelques contraintes supplémentaires. En effet, les méthodes de Unity ne peuvent être utilisées que dans le thread principal, ce qui fait que j'ai dû m'astreindre de certaines classes.

Nous conservons la formule utilisée dans la classe *ColorUtil*, mais nous supprimons l'affichage en couleur des images. Cette classe porte désormais le nom de *DepthValueUtil*. Nous ne modifions également plus la valeur de l'octet, mais retournons la valeur de la profondeur.

Une fois ce premier traitement terminé, il faut convertir tous les points (x, y) de l'image en point (x, y, z) de la scène. Cette partie a été la plus compliqué à mettre en oeuvre. Unity possède une méthode *ScreenToWorldPoint* qui permet de convertir un point d'un espace 2D à un espace 3D. Pourtant, pour des raisons inconnues, la conversion n'est pas réalisée. Plusieurs points ont été soulevés :

— Un problème d'unités. Comme Unity possède à la fois un référentiel écran et un référentiel local, il faut faire attention aux unités,

— Les échelles entre les axes ne sont pas proportionnelles, ce qui pourrait cacher la profondeur,

— Les axes x, y et z de Unity ne sont pas les mêmes que ceux du logiciel de visualisation du nuage de points.

Comme dit dans la partie Principe algorithmique, deux possibilités de traitement étaient possibles. J'ai opté pour le choix de réaliser un changement de référentiel à chaque image capturée.

Un inconvénient majeur est que cette méthode ne peut pas être appelée hors du thread principal. Cela a pour conséquence qu'il n'est pas possible de l'utiliser dans une autre classe.

3.5.2 Ecriture d'un fichier .obj

L'écriture du fichier se fait dans la classe *FileUtil*. Le fichier .obj est sauvegardé sur les lunettes HoloLens. Pour pouvoir réaliser cela, il faut utiliser la méthode *Application.persistentDataPath* de Unity. Cependant, comme la méthode *ScreenToWorldPoint*, elle ne peut pas être appelée en dehors du thread principal.

Pour visualiser le contenu du fichier, j'ai utilisé le logiciel Blender qui permet de faire de la modélisation 3D et qui supporte le format .obj.

5 Bilan et Conclusion

1 Bilan du PRD1

Mon projet est un sujet innovant qui se base sur la réalité augmentée que je n'avais jamais pu utiliser. Les possibilités offertes sont grandes et il était très intéressant de se renseigner sur le fonctionnement des lunettes qui embarque la réalité augmentée. Il s'agit cependant d'une technologie très récente qui dispose, pour le moment, de peu de documentation et d'une communauté réduite. Cela rend donc les informations disponibles sur les lunettes HoloLens peu nombreuses en comparaison à d'autres technologies.

Concernant la mise en place de l'environnement de développement, j'ai pu suivre une formation faite par Jordan NICOT qui m'a expliqué le fonctionnement des logiciels utilisés et celui des lunettes HoloLens. Cela m'a permis de rester en phase avec le planning de mon projet.

Cette première partie de projet m'aura également permis de mettre en application une part importante de gestion de projet. J'ai été légèrement perdu au début pour savoir comment m'orienter sur le long terme. Je pense au final avoir relativement bien suivi le planning établi. Celui-ci aura cependant été modifié deux fois au cours du semestre, mais sans conséquence importante.

Enfin, le projet appartenant plus au domaine de la recherche, les résultats sont encore incertains et plusieurs expérimentations seront sans aucun doute nécessaires pendant la deuxième phase du projet.

2 Bilan du PRD2

La deuxième partie du projet s'est révélée plus compliquée que prévu. En effet, il m'a fallu un certain temps de compréhension du processus de récupération du flux vidéo. Même si Microsoft met à disposition des codes d'exemples, j'ai trouvé qu'ils n'étaient pas suffisamment documentés. J'ai dû faire plusieurs recherches pour comprendre le rôle de certains objets et de certaines fonctionnalités du langage C#.

Certaines contraintes techniques du langage auxquelles je n'avais pas pensé sont apparues au cours du développement. La technologie étant nouvelle, peu de solutions sont proposées sur les forums et j'ai pris beaucoup de retard dans l'avancée de l'application. Cela m'a apporté une meilleure approche concernant la gestion d'un projet : j'ai réalisé qu'il était important de

privilégier la quantité de temps attribuée à une tâche au nombre de fonctionnalités prévues dans l'application.

Le projet m'a permis de me perfectionner dans le langage C#. En utilisant la caméra de profondeur des lunettes HoloLens, j'ai pu comprendre un peu plus comment la réalité augmentée fonctionne pour disposer des hologrammes dans une scène.

Pour terminer, même si le projet n'a pas abouti, il va être repris par un stagiaire pendant deux mois. Plusieurs réunions sont prévues afin de discuter de ce qui a été réalisé et de ce qui est attendu.

Je retiens de ce projet une expérience enrichissante qui m'a permis de découvrir une technologie en pleine croissance.

Annexes

A Spécifications fonctionnelles

Nous travaillerons avec les jeux de données suivants :

— Une souris d'ordinateur,
— Une glène d'omoplate

1 Récupérer un flux d'images Bitmap

Nous devons récupérer les images Bitmap de différents points de vue d'un utilisateur en mouvement autour d'une cible fixe. Cet objectif sera validé lorsque nous pourrons présenter ces images.

La durée de réalisation est estimée à deux semaines de projet.

Rôle : primaire
Entrée : flux vidéo
Pré-conditions : -
Sortie : ensemble d'images Bitmap de la cible
Post-conditions : -

2 Déterminer la profondeur des points de la cible

2.1 Version 1

Cette fonctionnalité permet de déterminer la profondeur de chaque pixel des images Bitmap. Nous récupérons pour cela les données de profondeur (distance de fiabilité minimale et maximale).

Dans cette première version, on associera aux pixels une couleur en fonction du niveau de profondeur déterminé. Cet objectif sera validé lorsque nous pourrons présenter des images Bitmap colorées.

La durée de réalisation est estimée à deux semaines de projet.

Rôle : primaire
Entrée : flux vidéo + ensemble d'images Bitmap
Pré-conditions : -
Sortie : ensemble d'images Bitmap
Post-conditions : -

2.2 Version 2

Nous souhaitons cette fois-ci récupérer uniquement les points de la cible. Nous associerons une couleur en fonction des différents niveaux de profondeur déterminés. Cet objectif sera validé lorsque nous pourrons présenter des images Bitmap colorées uniquement au niveau de la cible.

La durée de réalisation est estimée à une semaine de projet.

Rôle : primaire
Entrée : flux vidéo + ensemble d'images Bitmap
Pré-conditions : -
Sortie : ensemble d'images Bitmap
Post-conditions : -

3 Obtention d'un fichier .obj

Cette fonctionnalité sera réalisée de deux façons différentes. Dans la première version, le fichier .obj sera complété à chaque image obtenue (le contenu du fichier sera donc écrit par bloc). Dans la deuxième version, le fichier ne sera créé qu'une fois toutes les images obtenues. Cela nous permettra ainsi de comparer les performances des deux méthodes.

Cet objectif sera validé quand nous pourrons présenter deux modèles 3D. En fonction de la qualité du nuage de points, nous serons alors capable de déterminer s'il existe une méthode meilleure que l'autre.

3.1 Changement de référentiel

La durée de réalisation est estimée à trois semaines de projet.

Rôle : primaire
Entrée : ensemble de vecteurs (x, y, z)
Pré-conditions : coordonnées dans le référentiel de l'écran
Sortie : ensemble de vecteurs (x, y, z)
Post-conditions : coordonnées dans le référentiel de la scène

3.2 Ecriture dans un fichier .obj

Nous devons générer un fichier 3D qui contiendra les données de la cible. Nous pourrons importer le fichier dans un logiciel de visualisation 3D comme Blender.

La durée de réalisation est estimée à une semaine de projet.

Rôle : primaire
Entrée : ensemble de vecteurs (x, y, z)

Pré-conditions : coordonnées dans le référentiel de la scène
Sortie : fichier .obj
Post-conditions : -
Algorithme :

Ecrire la ligne $v\ x\ y\ z$ dans le fichier

4 Suppression des doublons du fichier .obj

Lorsque l'utilisateur tourne autour de la cible, plusieurs mêmes points sont récupérés et sauvegardés dans le fichier .obj, ce que nous souhaitons éviter pour des raisons de performance (en terme d'écriture et de lecture du fichier) mais également pour réduire la taille du fichier.

La durée de réalisation est estimée à deux semaines de projet.

Rôle : primaire
Entrée : fichier .obj
Pré-conditions : format du fichier .obj valide
Sortie : fichier .obj
Post-conditions : taille du fichier réduite

5 Démarrage et fin de l'application

Cette fonctionnalité permet le lancement et l'arrêt de l'application grâce au geste "air tap".

5.1 Version 1

Cette première version permet de valider la reconnaissance gestuel. Elle sera validée lorsqu'un enregistrement vidéo pourra être présenté.

Rôle : secondaire
Entrée : -
Pré-conditions : -
Sortie : un enregistrement vidéo
Post-conditions : -

5.2 Version 2

La deuxième version permet l'exécution du programme de traitement et de récupérer le fichier 3D contenant les données de la cible. Il sera validé lorsque l'on obtiendra un fichier .obj.

Rôle : secondaire
Entrée : cible réelle de la scène
Pré-conditions : -
Sortie : fichier .obj
Post-conditions : -

B Spécifications non fonctionnelles

1 Contraintes de développement

1.1 Matériel

Nous utiliserons les lunettes de réalité augmentée, les HoloLens.

1.2 Logiciels

Comme dit précédemment dans ce rapport, nous avons besoin :

- **Unity**, et plus précisément la version 2017.4.12f1 LTS,
- **Visual Studio 2017**, avec l'utilisation du kit de développement de jeux avec Unity,
- **Windows 10**, la version 10.0.17134.0.

1.3 Langage de programmation

L'utilisation de l'environnement Unity impose, dans le cadre de ce projet, le développement de scripts en C#.

2 Contraintes de fonctionnement

2.1 Performances

Les performances des lunettes HoloLens sont bonnes. La batterie et la sécurité sont toutefois légèrement affectées lorsque les lunettes sont en mode *Research*.

2.2 Capacités

L'un des enjeux importants du projet est de démontrer sa faisabilité.

2.3 Maintenance et évolution du système

Des développeurs doivent pouvoir modifier ou ajouter des fonctionnalités simplement. Pour cela, nous devons assurer un développement de type génie logiciel.

C Dossier de tests

Test n°1

Description : récupération du flux de profondeur.

Attendu : une image faite de nuances de gris en fonction de la distance entre les lunettes HoloLens et une cible.

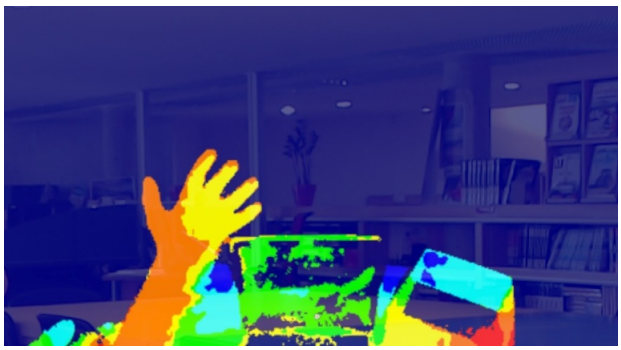
Résultat : succès.

Test n°2

Description : coloration de l'image en fonction de la valeur de la profondeur des objets. Le test s'effectue sur une main, un écran d'ordinateur et une boîte.

Attendu : une image colorée de rouge, jaune, vert et bleu.

Résultat : succès



Test n°3

Description : coloration des objets situés à une certaine distance uniquement. Le test s'effectue sur des mains, un écran d'ordinateur et une boîte. On souhaite récupérer uniquement ces trois

objets, sans la détection du reste de la salle. La distance maximale avec la caméra est d'environ 40cm.

Attendu : une image colorée de rouge, jaune, vert et bleu au niveau des trois objets à détecter. Le reste de la pièce reste blanc.

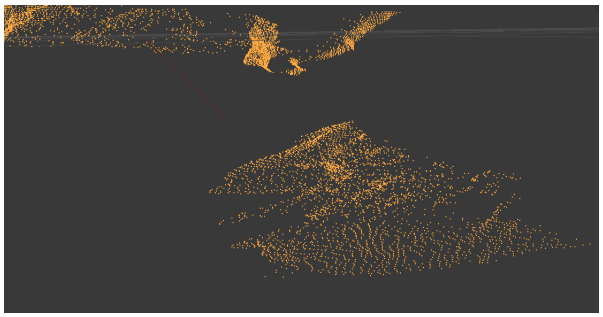
Résultat : succès

Test n°4

Description : récupération d'un nuage de points d'une souris d'ordinateur.

Attendu : un nuage de points représentant la souris d'ordinateur en 3D.

Résultat : échec



D Gestion de projet

1 Méthodologie

Dans un premier temps, je comptais suivre une méthode de développement de type *cycle en V*. Je trouvais cette méthode de travail très bien adaptée au projet. En effet, les phase d'analyse des besoins, des spécifications et de la conception correspondent à la première partie du PRD, tandis que la phase de développement est réservée au second semestre.

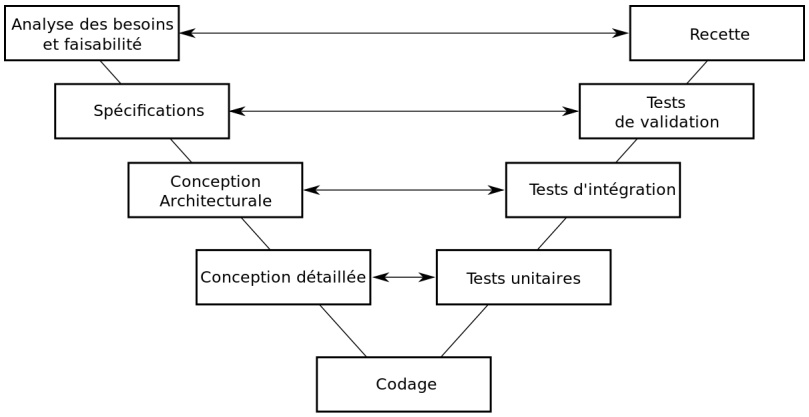


Figure 1 – *Le cycle en V*

Ce choix était également justifié par le fait que l'objectif était fixé dès le départ.

Cependant, pendant la phase de développement, je me suis aperçu que ma méthode de travail s'approchait d'une méthode agile. Au final, j'ai réalisé trois livrables que j'améliorai pour me rapprocher de l'objectif du projet.

2 Plannings

Le planning de la première partie du projet a été correctement respecté. Suite à une réunion avec mes tuteurs, nous avons pu clairement définir les différentes étapes et c'est ce qui a permis d'éviter tout retard. J'ai donc apporté une modification à mon planning initial qui prévoyait

d'étudier le matching entre l'objet réel et virtuel. J'ai supprimé cet objectif pour me concentrer sur la récupération d'un nuage de points. Plus tard, quelques modifications ont été effectuées dans la prévision de l'étude du mode *Research* qui est passée d'une à trois semaines de recherches, mais cela n'a pas eu de conséquences importantes sur l'avancée du projet. Au contraire, cela m'aura permis d'acquérir de meilleures connaissances sur le fonctionnement du mode.

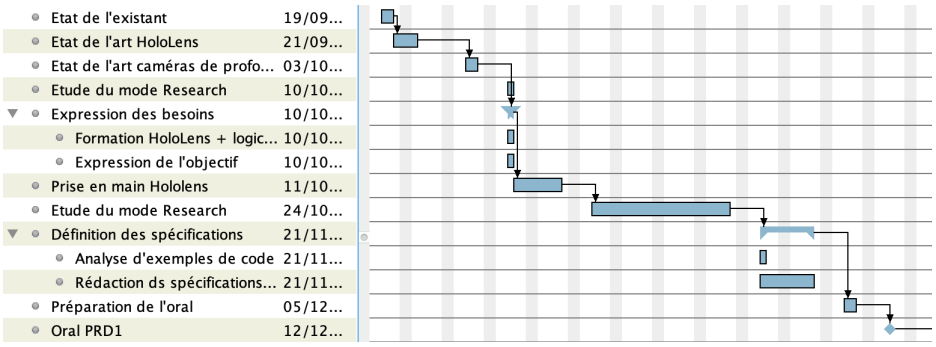


Figure 2 – *Diagramme de Gantt - PRD1*

Pour la deuxième partie du projet, il s'agit de visualiser le futur développement de l'application. Nous pouvons élaborer le diagramme prévisionnel suivant :

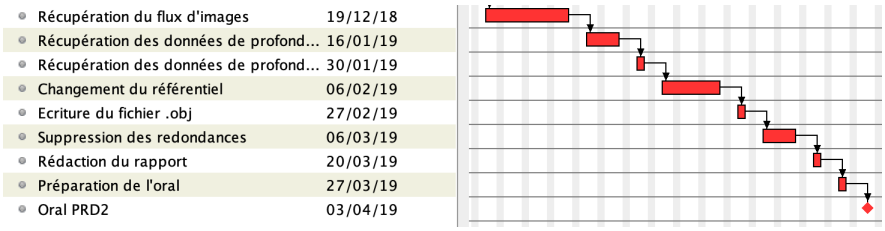


Figure 3 – *Diagramme de Gantt prévisionnel - PRD2*

Suite à des difficultés de développement, les tâches du diagramme précédent ont dû être évaluées de nouveau :

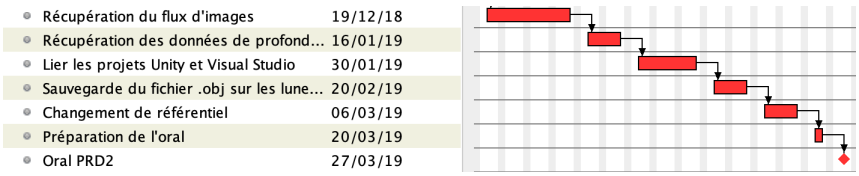


Figure 4 – *Diagramme de Gantt réellement effectué - PRD2*

E Cahier du développeur

1 Mise en place de l'environnement

1.1 Installation des logiciels requis

Les logiciels nécessaires sont disponibles sur le site de Microsoft : Install the tools. Voici les versions installées pour ce projet :

1. Unity 2017.4.12f1 LTS,

2. Visual Studio 2017 Community. Il ne faut pas oublier de télécharger la liste des composants suivants :

   — Développement .NET Desktop,
   — Développement Desktop en C++,
   — Développement pour la plateforme Windows universelle,
   — Développement de jeux avec Unity.

   Si tous les composants ont bien été sélectionnés, on obtient cette liste récapitulative :

Figure 1 – *Détails de l'installation*

   Dans *Composants individuels*, il faut vérifier que le SDK Windows 10 (10.0.17134.0) est bien sélectionné. Il s'agit de la version installée sur les lunettes HoloLens.

1.2 Exportation d'un projet sous Unity

Le plus simple est de suivre le tutoriel disponible sur le site de Microsoft : MR Basics 101.

De manière générale, voici les étapes nécessaires à l'exportation d'un projet :

1. Sélectionner **File > Build Settings**,
2. Dans **Scenes In Build**, sélectionner la scène requise. Sélectionner **Add Open Scenes** pour en ajouter une si le champ est vide,
3. Dans **Platform**, sélectionner *Universal Windows Platform*,
4. Configurer le reste du projet comme ci-dessous :

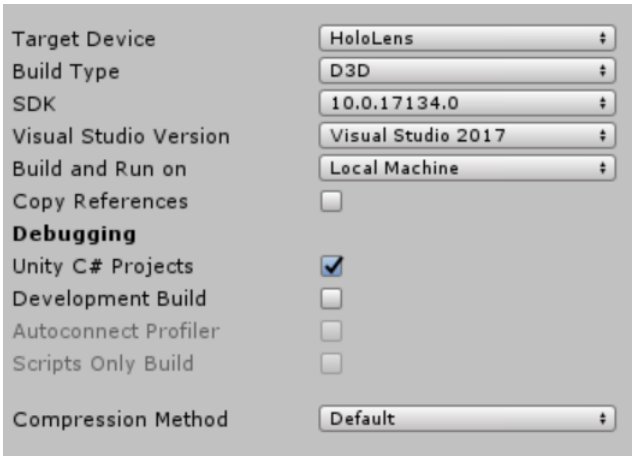


Figure 2 – *Configuration*

5. Sélectionner **Player Settings**. Dans le panneau **Inspector** de droite, choisir les paramètres *Universal Windows Platform* :

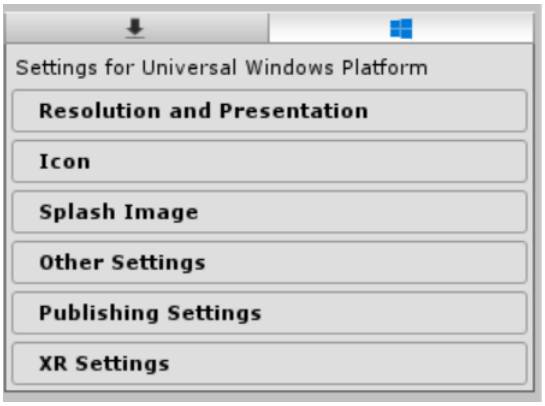


Figure 3 – *Inpector*

Vérifier ensuite la configuration de la partie **Other Settings** et celle de la partie **XR Settings**.

Figure 4 – *Configuration des paramètres*

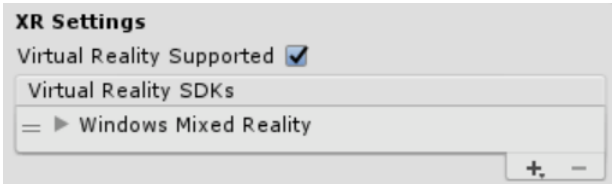


Figure 5 – *Configuration des paramètres*

6. Sélectionner **Build** pour déployer le projet. Il faut choisir un dossier dans lequel sera placé la solution .sln. Pour cela, le plus simple est de créer un dossier UWP.

1.3 Compilation d'un projet sous Visual Studio

1. Avant d'importer l'application sur les lunettes, il faut d'abord vérifier qu'elles sont reconnues. Pour cela, le plus simple est de connecter les lunettes par câble USB,

2. Dans le dossier UWP du projet Unity précédemment généré, sélectionner le fichier *.sln*,

3. Configurer Visual Studio comme ci-dessous :

Figure 6 – *Configuration de Visual Studio*

1.4 Autorisation de permissions

Pour que l'application soit fonctionnelle, il est nécessaire d'autoriser certains accès aux lunettes HoloLens.

```
xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10"
xmlns:mp="http://schemas.microsoft.com/appx/2014/phone/manifest"
xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10"
xmlns:uap2="http://schemas.microsoft.com/appx/manifest/uap2/windows10"
xmlns:rescap="http://schemas.microsoft.com/appx/manifest/foundation/windows10/restrictedcapabilities"
IgnorableNamespaces="uap uap2 mp rescap">
```

Figure 7 – *Les packages à importer*

```
<Capabilities>
  <Capability Name="internetClient" />
  <Capability Name="internetClientServer" />
  <Capability Name="privateNetworkClientServer" />
  <uap2:Capability Name="spatialPerception" />
  <rescap:Capability Name="perceptionSensorsExperimental" />
  <DeviceCapability Name="microphone" />
  <DeviceCapability Name="webcam" />
  <DeviceCapability Name="bluetooth" />
</Capabilities>
```

Figure 8 – *Les permissions à accorder*

2 Créer un projet XAML

1. Sélectionner **Fichier** > **Nouveau Projet**.

2. Dans la catégorie **Visual C#** > **Windows universel**, choisir **Application vide**. Saisir le nom de l'application.

3. Dans **Version cible** (la version cible correspond à la version de Windows installée sur les lunettes HoloLens), choisir *Build 17134*.

4. Il est désormais possible d'ajouter des composants, en passant par le mode *Design* ou par le mode *XAML* en utilisant des balises.

3 Génération de la documentation C#

Sous Visual Studio, la documentation ne se crée que sous un format XML. Pour obtenir une version HTML, il faut utiliser un autre outil, par exemple Doxygen.

Voici les étapes à suivre pour générer la documentation :

1. Sélectionner le dossier source du projet.

2. Dans **Wizard** > **Project** :

   — Saisir le nom du projet,
   — Spécifier le dossier dans lequel se situe les fichiers du projet,
   — Cocher **Scan recursively**,
   — Choisir le dossier de destination du fichier HTML.

3. Dans **Wizard** > **Mode** :

   — Sélectionner **All Entities**,
   — Sélectionner **Optimize for Java or C# source**.

4. Dans **Wizard** > **Output**, sélectionner le format HTML.

5. Dans **Expert** > **Build** :

   — Sélectionner **EXTRACT_PRIVATE**,
   — Sélectionner **EXTRACT_STATIC**.

— Sélectionner **EXTRACT_LOCAL_CLASSES**.

6. **Run doxygen**

4 Connexion au Windows Device Portal

Le *Windows Device Portal* est un outil qui permet de gérer les lunettes HoloLens à distance. Il y a deux façons pour y accéder :

1. Accès par un câble USB. On se connecte en *localhost* sur le port 10080 (*localhost :10080*),
2. Accès par le réseau. Dans ce cas, il faut faire attention à ce que les lunettes et l'ordinateur soient sur le même sous-réseau. On accède au portail par l'adresse IP.

5 Le mode Research

5.1 Activation/Désactivation

L'activation et la désactivation du mode *Research* se fait dans la partie *System* > *Research mode*. Pour que le choix soit effectif, il faut redémarrer les lunettes.

5.2 Exemples de codes

Le dépôt Github HoloLensForCV propose de nombreux exemples pour utiliser les possibilités offertes par le mode.

Avant de d'extraire l'archive, il faut être sûr de sélectionner **Débloquer** dans les **Propriétés**. Il faut également extraire l'intégralité de l'archive puisque l'ensemble des exemples utilisent un dossier partagé. Enfin, sous Visual Studio, la plateforme choisie par défaut est de type ARM. Il faut la remplacer par x86.

6 Description des versions du projet

Le projet a connu trois versions différentes. Chacune correspond à une fonctionnalité qui a permis d'avancer vers l'objectif final. Voici le détail de chacune de leurs classes.

6.1 ScannerGleneV0

Cette version permet de récupérer le flux de profondeur et d'afficher les différentes images extraites dans une interface XAML.

— **SensorStream** : cette classe permet de récupérer le flux de la caméra de profondeur et d'en extraire des images 2D sous le format Bitmap.

— **ProcessFrame** : cette classe permet de présenter les images capturées à l'utilisateur dans une interface XAML.

6.2 ScannerGleneColorPixels

Cette version permet de colorer les images extraites du flux de profondeur et de les présenter dans une interface XAML.

— **SensorStream** : cette classe permet de récupérer le flux de la caméra de profondeur et d'en extraire des images 2D sous le format Bitmap.

— **ProcessFrame** : cette classe permet de traiter une image Bitmap en récupérant les informations de profondeur qui y sont contenues. Chaque pixel est traité individuellement.

— **ColorUtil** : cette classe permet d'affecter une couleur à un pixel selon la valeur de la profondeur qui lui est associée. Plus le point est loin, plus la couleur tend vers le bleu. A l'inverse, plus il est proche, plus la couleur tend vers le rouge.

6.3 ScannerGleneV2

Version non fonctionnelle.

Cette version permet de capturer une cible et de la visualiser sous forme d'un nuage de points.

— **SensorStream** : cette classe permet de récupérer le flux de la caméra de profondeur et d'en extraire des images 2D sous le format Bitmap. Le changement de référentiel des coordonnées de la cible ne s'effectue qu'une fois que l'utilisateur a terminé le scan. Pour cela, les images sont sauvegardées.

— **ProcessFrame** : cette classe permet de traiter une image Bitmap en récupérant les informations de profondeur qui y sont contenues. Chaque pixel est traité individuellement.

— **DepthValueUtil** : cette classe permet de calculer la valeur de la profondeur associée à un point. Cette classe s'approche de la classe *ColorUtil* précédente sans l'étape de la coloration. Si la valeur de la profondeur est trop importante, alors elle n'est pas sauvegardée (car cela signifie que la cible est trop loin).

— **ScreenToWorld** : cette classe permet de changer le référentiel d'un point, en transformant des coordonnées 2D (correspondant à l'écran) en coordonnées 3D (correspondant à la scène).

— **FileUtil** : cette classe permet de sauvegarder dans un fichier .obj les coordonnées 3D de la cible.

7 Données d'authentification

Les login et mots de passe, nécessaires à l'utilisation des divers logiciels, ont été transmis à mes encadrants.

F Guide utilisateur

1 Lancement de l'application

Il est nécessaire que l'utilisateur connaisse et maîtrise les gestes expliqués dans la partie La gestuelle et la reconnaissance vocale.

L'application s'ouvre en cliquant sur l'icône correspondante *ScannerGleneV2*. Celle-ci peut être trouvée soit dans le menu d'accueil, soit parmi le reste des applications, accessibles en cliquant sur *All apps*.

Le logo d'Unity apparaît une fois l'application lancée. Si l'application a été de nouveau générée (ajout de lignes de code par exemple), un message demandant la permission d'accéder à la caméra des lunettes s'affiche. Sinon l'application se lance automatiquement.

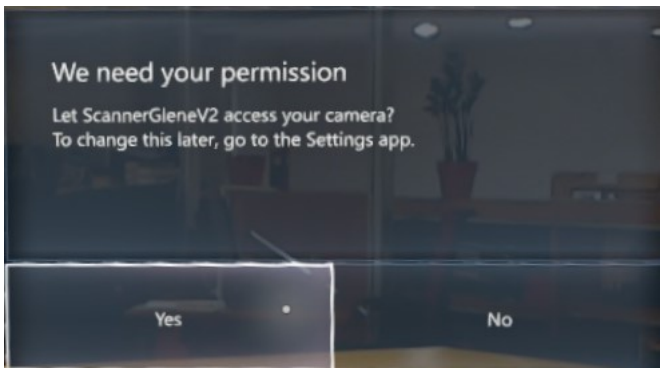


Figure 1 – *Permission d'accès à la caméra*

2 Utilisation de l'application

Pour récupérer le nuage de points d'une cible, il faut tourner autour d'elle à une distance d'environ 30cm. Une fois le temps jugé suffisant, il faut réaliser le geste de floraison pour quitter l'application.

3 Visualisation du nuage de points

Le fichier .obj créé est sauvegardé sur les lunettes. Par le *Device Portal*, il faut aller dans *System* > *File Explorer* > *AppData* > *Local* > *Packages* > *ScannerGleneV2* > *LocalState*. Il faut cliquer sur l'icône en forme de disquette pour télécharger le fichier.

Le nuage de point peut ensuite être visualisé sous n'importe quel logiciel supportant le format .obj. Sous Blender, il faut aller dans *File* > *Import* > *Wavefront (.obj)* pour importer le fichier.

Comptes rendus hebdomadaires

Compte rendu n°1 du 19/09/2018

Comme entrevu avec Mr Proust, mes recherches se sont concentrées autour de l'Hololens et des images 3D de façon générale (vous trouverez en pièces jointes mes prises de note un peu plus détaillées).

Hololens :

- Les composants matériels
- Les caractéristiques techniques (OS, connectique,...)
- Le fonctionnement général du lunettes et de ses composants

Image 3D :

- Composition d'un objet 3D
- Comment comparer deux objets 3D ?
- Triangulation pour reconstituer une surface à partir de points
- Algorithme de recherche de points caractéristiques

Compte rendu n°2 du 26/09/2018

Comme précisé dans cr0, je serai bien présent de 8h15 à 12h30 et de 14h à 18h15 à la bibliothèque. En cas d'imprévu je vous en informerai.

Entrevues avec Christian Proust. Remise des rapports de PRD de Jordan NICOT, de Alexis BARON (PRD 2017-18) et d'un rapport de projet PeiP2 pour lecture. J'ai commencé la rédaction du rapport principalement sur la partie état de l'art, notamment :
a) Image 3D

- Définiton d'un objet 3D
- Définition d'une vidéo 3D
- Définiton d'un fichier OBJ
- Comment obtenir un nuage de points (un état de l'art plus précis sur les caméras de profondeur est prévu plus tard)
- Comment reconstruire l'objet à partir de ce nuage

b) L'Hololens

- Les composants matériels
- Les caractéristiques techniques

J'ai également rédigé un début de planning, je vous enverrai très prochainement un diagramme de Gantt qui sera ajouté au rapport.

- 26-27/09 = état de l'art sur hololens + image 3D
- 3-4/10 = état de l'art sur les caméras de profondeur
- 10-11/10 = état de l'art sur les librairies + SDK
- 18-?/10 = études des conditions d'initialisation + quantification du matching entre l'image virtuelle et le réel
- ?-? = rédaction du cahier des spécifications

J'ai pour le moment un peu de mal à voir sur le long terme et savoir si mes prévisions sont trop ambitieuses/pas assez. Peut-être serait-il bien d'en discuter avec vous.

Compte rendu n°3 du 03/10/2018

Etat de l'art sur les caméras de profondeur. Voici les deux points que l'on retrouve dans le rapport :

- Deux catégories de caméras,
- Les techniques les plus répandues

J'ai également mis à jour la partie "Description générale", sous-parties "Environnement" et "Utilisateur". J'ai repris les deux utilisateurs "chirurgien" et "technicien" des deux rapports précédents. Mais leur rôle a-t-il évolué par rapport à ce qui a déjà été fait ? A en discuter.

Compte rendu n°4 du 10/10/2018

* Quelques recherches sur le mode Research proposé par les lunettes.

* Réunion avec J. NICOT et C. PROUST :

- Prise en main rapide des logiciels et des lunettes HoloLens,
- Redéfinition du planning (que vous trouverez ci-dessous),
- Discussion du premier objectif du PRD : récupérer le nuage de points d'un objet. Dans un premier temps, plutôt que de travailler directement avec la forme de la glène, je travaillerai avec un objet simplifié (une souris d'ordinateur).

* Suite à cette réunion, j'ai commencé à prendre en main les lunettes grâce aux tutoriels :

- Visualisation d'hologrammes,
- Prise en main des gestes

* Mise à jour du planning :

- 26-27/09 = état de l'art sur HoloLens + image 3D
- 3-4/10 = état de l'art sur les caméras de profondeur
- 10-11/10 = début d'étude du mode Research + prise en main des lunettes grâce aux tutos
- 17-18/11 = prise en main des lunettes grâce aux tutos
- 24-25/11 = étude approfondie du mode Research

Compte rendu n°5 du 17/10/2018

J'ai continué les tutoriels disponibles sur le site de Microsoft, en particulier :

- Prise en main du curseur (qui se déplace en fonction du regard),
- Prise en main des gestes, en lien avec le curseur,
- Cartographie spatiale : reconnaissance des murs, des tables, du sol,
- Affichage d'un planétarium dans la pièce : possibilité de déplacer l'hologramme

grâce aux gestes, détection des planètes cachées par le Soleil (ajout d'une texture différente pour montrer la partie cachée).

Sur les conseils de Jordan, j'ai également modifiés de temps en temps des scripts proposés pour plusieurs raisons :

— Prendre en main l'univers Unity,
— Comprendre le code et ce qu'il produisait en affichant un message quand une action était réalisée (par exemple, le curseur qui touche un objet).

En revanche, je ne sais pas si ce que j'ai fait pendant ces deux séances doivent apparaître dans mon rapport. Pour ma part, je dirais que ce n'est pas nécessaire.

Je souhaiterais également revenir sur la technique "Temps de vol" de la caméra de profondeur des lunettes HoloLens. En effet, je n'ai pas précisé laquelle des deux techniques était utilisée. Je n'arrive en fait pas à trouver une information sûre, même après plusieurs recherches...

Compte rendu n°6 du 24/10/2018

Début de prise en main du mode Research. Suite à une mise à jour nécessaire pour utiliser ce mode, j'ai rencontré plusieurs problèmes d'Internet et d'authentification. Après une intervention de Jordan et M. Proust, les problèmes ont pu être résolus.

Je n'ai donc pas eu beaucoup de temps pour appréhender ce mode. Pour le moment, lorsque j'utilise la fonctionnalité qui permet de visualiser en direct ce qu'on voit à travers les lunettes, les hologrammes s'affichent sur un fond vert (le monde réel n'est donc plus visible) et l'affichage en est fortement affecté (en terme de performances). J'ai cependant vu sur le forum de Microsoft que je n'étais pas seul dans ce cas.

Compte rendu n°7 du 07/11/2018

Le 7/11, j'ai assisté à un colloque sur l'IA. En plus d'assister à la conférence de plusieurs intervenants, j'ai pu faire des démonstrations des lunettes et discuter des applications possibles.

Le 8/11, j'ai approfondi mes recherches sur le Research mode des lunettes. J'ai commencé à étudier en particulier deux programmes d'exemples qui pourraient servir pour le projet :

— Le premier permet d'accéder aux 4 caméras permettant la localisation dans l'espace ainsi qu'à la caméra de profondeur. Le programme présente ainsi ce que "voit" les caméras (notamment la caméra de profondeur par un jeu de couleur, que vous trouverez en pièce jointe). En analysant le code, j'ai vu que l'on pouvait récupérer un tableau contenant les différentes distances calculées et je pense qu'il s'agit de celles des différents points. En récupérant les coordonnées x et y, il serait peut-être possible de transformer un objet en nuage de points.

— Le deuxième programme permet d'accéder au flux de la caméra qui enregistre les photos et les vidéos. Il y a ensuite l'ajout d'un algorithme de Canny (avec OpenCV) effectué soit sur un ordinateur, soit sur les lunettes directement (photos en pièce jointe).

Concernant mon planning, je n'avais auparavant indiqué que les deux jours 24 et 25/10 pour l'étude du mode. Je pense rajouter au moins deux jours de plus avant de réaliser une étude plus approfondie sur la récupération du nuage de points, qui serait donc réalisée à partir du 21/11.

Compte rendu n°8 du 14/11/2018

Rendez-vous avec M. Ramel pour discuter des spécifications. Dans le rapport, j'ai mis au propre le planning sous forme de diagramme de Gantt ainsi qu'une modélisation (cas d'utilisation) de l'application.

J'ai également continué à chercher où peuvent être stockées les différentes coordonnées des points récupérées par la caméra de profondeur.

Le jeudi après-midi a eu lieu le forum des entreprises, je n'ai donc pas pu avancer autant que les séances précédentes.

Compte rendu n°9 du 21/11/2018

Cette semaine, je me suis principalement concentré sur la rédaction du rapport, la soutenance approchant.

Le 22/11, réunion avec J. NICOT et C. PROUST pour faire une mise au point du projet ainsi que pour discuter des spécifications : lecture du code source d'un exemple fourni avec le mode Research et explications à nouveau des objectifs souhaités. Nous voulons obtenir une seule image en nuage de points de la cible, ce qui signifie qu'il n'y aura qu'un seul fichier 3D à créer. J'étais parti sur une mauvaise piste en me basant "trop" sur le code d'exemple et en partant sur l'idée de créer un fichier par point de vue de l'utilisateur.

Les étapes à mettre en place, dans les grandes lignes, sont les suivantes :

1. Lancement de l'application avec un "air tap", par exemple.
2. Les coordonnées des points récupérés doivent correspondre aux coordonnées dans la scène. Unity possède des méthodes permettant de convertir des coordonnées d'un point de l'écran en coordonnées de la scène.
3. Ecriture des coordonnées dans un fichier .obj.
4. En procédant comme cela, il y aura sûrement des redondances de points. Nous envisagerons un algorithme pour résoudre ce problème plus tard.
5. Fin de l'application avec un "air tap".

J'ai mis à jour les spécifications en conséquence.

Compte rendu n°10 du 28/11/2018

L'échéance de la présentation orale arrivant prochainement (le 13/12 de 12h à 12h40), je me suis concentré sur la rédaction du rapport et des spécifications. Il manque encore quelques informations que je souhaiterais ajouter dans le rapport, mais il s'agit quasiment de la version finale.

Compte rendu n°11 du 05/12/2018

J'ai terminé la rédaction du rapport du PRD1 et j'ai terminé le diaporama de présentation. Je compte me préparer jusqu'à jeudi et je pense que ma séance de mercredi y sera consacrée.

Compte rendu n°12 du 19/12/2018

La deuxième partie du PRD a débuté et j'ai donc commencé le développement. Pour le moment, je suis en train de coder la partie permettant de récupérer le flux vidéo des lunettes HoloLens.

Compte rendu n°13 du 09/01/2019

J'ai réussi à accéder et à obtenir le flux vidéo de la caméra des HoloLens. Le code à mettre en place était assez technique, mais j 'ai pu m'appuyer sur un exemple de code qui m'a permis de bien comprendre le principe.

Compte rendu n°14 du 16/01/2019

Ces séances avaient pour objectif de colorer les pixels des images Bitmap en fonction des données de profondeur. Cela démontrera que les données de profondeur sont bien récupérées.

Le but est de colorer l'image en fonction d'une échelle (plus la profondeur est importante, plus les pixels seront d'une certaine couleur, et inversement). Je n'ai pas encore réussi. En revanche, je peux affirmer que cela sera possible puisque si je décide d'attribuer moi-même une couleur, cela fonctionne.

Compte rendu n°15 du 23/01/2019

Je peux récupérer les coordonnées du nuage de points d'une cible, image par image. Pour pouvoir visualiser cette récupération et montrer qu'elle fonctionne, chaque pixel est coloré en fonction de la valeur de la profondeur (plus un point est loin, plus il tend vers le bleu. A l'inverse, plus il est près, plus il s'approche du rouge).

Vous trouverez les images en pièce-jointe. Sur deux images, tous les pixels sont colorés. Sur les deux autres, seule une partie de l'image est colorée, afin de montrer que je peux récupérer juste la cible qui nous intéresse.

Les tests que j'ai réalisé n'ont pas de réelle valeur, je les ai fait seulement pour ces captures d'écran, mais je compte réaliser une série de tests beaucoup plus sérieuse.

Compte rendu n°16 du 30/01/2019

Je suis désormais arrivé à l'étape du changement de référentiel. Jusqu'à maintenant, j'ai réalisé mon application sans l'aide de Unity. Il faut désormais que je l'utilise afin de pouvoir récupérer les coordonnées des lunettes HoloLens dans la scène, et donc pouvoir convertir les coordonnées du référentiel lié à l'écran en celles du référentiel de la scène.

Pour le moment, je rencontre quelques difficultés dans l'accès à la caméra (la Main Camera de Unity) qui me permettrait de récupérer les coordonnées des lunettes.

Compte rendu n°17 du 06/02/2019

Mise au point avec Jordan sur l'avancée du projet : explications d'où j'en suis actuellement, les problèmes rencontrés, les objectifs futurs. Discussion rapide autour de l'algorithme de changement de référentiel.

Je me retrouve bloqué concernant le lien du projet (sous Visual Studio) avec Unity : j'obtiens des erreurs d'assembly "Windows.Foundation.ApiContract". Je me demande si cela ne vient pas du fait que j'utilise des objets UWP non supportés par Unity. Je n'ai toujours pas réglé ce problème. Les solutions proposées sur les forums ne règlent rien. De ce fait, je n'arrive pas à avancer.

Compte rendu n°18 du 13/02/2019

J'ai pu débloquer mon problème et lier mon projet Visual Studio avec Unity. Il me reste encore à régler un problème mineur d'écriture dans un fichier (qui se fait dans une méthode qui ne semble pas être appelée, alors que cela devrait être le cas).

Compte rendu n°19 du 27/02/2019

Comme discuté avec Mr Proust et Jordan samedi aux JPO, j'arrive à créer un fichier .obj et à l'importer sous Blender pour visualiser le nuage de points. Le processus qui consiste à scanner la cible jusqu'à la visualisation du résultat fonctionne donc.

En revanche, pour le moment, le nuage de points obtenu ne correspond pas à la cible scannée. Je ne sais pas encore si c'est parce qu'il n'y a pas assez de points récupérés ou si le changement de référentiel ne fonctionne pas comme il faut.

Compte rendu n°20 du 06/03/2019

Je n'ai pas encore réussi à obtenir le nuage de points correspondant à une cible. Mr Slimane m'a conseillé d'inverser les valeurs des coordonnées (y et z, j'ai également essayé avec x), sans succès.

J'ai donc réalisé la documentation du projet et avancé mon rapport que vous pourrez trouver en pièce jointe (ce n'est pas la version finale).

Compte rendu n°21 du 13/03/2019

J'ai continué la rédaction du rapport (vous le trouverez en pièce jointe) et la documentation du projet. J'ai également essayé à nouveau de régler le problème du nuage de points. Cette fois-ci, j'ai voulu visualiser à quoi ressemblait le nuage de points 2D (en essayant de rester immobile, comme pour une image fixe).

Sur la première image, on peut voir un carré central et une forme plus ovale. Cette image découle du scan de mon ordinateur et de la boîte ovale des lunettes. Sur la deuxième image, on peut voir un carré central et une forme plus rectangulaire. Cette image découle du scan de mon ordinateur et de la boîte (vue de côté) des lunettes. Je ne sais pas s'il s'agit d'une coïncidence mais cela conforte l'idée que la scène est bien récupérée et que ce n'est que la conversion 2D -> 3D qui pose problème. Dans tous les cas, je trouvais cela intéressant de vous le montrer.

Webographie

[WWW1] Jean-Daniel Boissonnat. *Reconstruire des surfaces pour l'imagerie*. 13 septembre 2005. URL : https://interstices.info/jcms/c_12845/reconstruire-des-surfaces-pour-limagerie.

[WWW2] Dell. *Réalité augmentée vs réalité virtuelle : le match*. 20 septembre 2016. URL : http://www.zdnet.fr/actualites/realite-augmentee-vs-realite-virtuelle-le-match-39841616.htm.

[WWW3] Hans.P.G. *How does Script t15 Skin Point Cloud ?* 14 avril 2009. URL : http://hanspg.web.fc2.com/Pages/csv_scripts/algo_t15.html.

[WWW4] Hans.P.G. *A Script to Skin a Point Cloud (for Blender 2.6x or Later)*. janvier 2012. URL : http://blenderartists.org/forum/showthread.php?241950-A-Script-to-Skin-a-Point-Cloud-(for-Blender-2-6x-or-Later).

[WWW5] Microsoft. *Tutorials and sample apps*. URL : https://docs.microsoft.com/en-us/windows/mixed-reality/academy.

Bibliographie

[1] A. Baron, J. Nicot, M. Slimane et al. « Hololens, Réalité Augmentée et Épaule ». Tours : Polytech Tours, mars 2018.

[2] J. Berhouet. « Optimisation de l'implantation glénoïdienne d'une prothèse d'épaule : de la reconstruction 3D à la réalité augmentée ». Tours : Université de Tours, octobre 2016.

[3] J. Berhouet, M. Facomprez, D. Boas, C. Proust, M. Slimane et L. Favard. « Master Course in Shoulder and Elbow Surgeries, Decision making in difficult situations ». In : (10 et 11 avril 2015).

[4] Maxime Facomprez, Julien Berhouet et al. « Réalisation d'une application sur lunettes à réalité augmentée visant à assister le chirurgien au cours d'une opération d'omoplate ». Tours : Polytech Tours, mai 2015.

[5] J. Nicot, J. Berhouet, M. Slimane et al. « Prothèse d'épaule et lunettes connectées ». Tours : Polytech Tours, avril 2017.

[6] « Plus belle la réalité ». In : *l'OBS*, *pp. 114-115* (18 mai 2017).

Projet Recherche & Développement

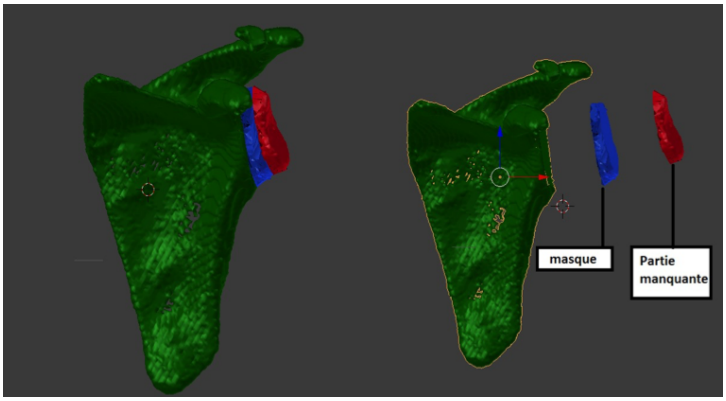
HoloLens, Réalité Augmentée et Epaule

Grégoire MARCO

Encadrement : Jordan NICOT et Mohand SLIMANE

Objectifs

— Etude de la réalité augmentée
— Assister les chirurgiens pendant une opération à l'aide d'une application



Omoplate et sa glène reconstituée virtuellement

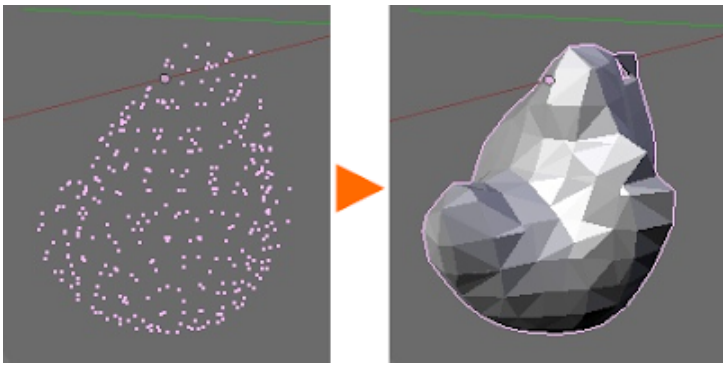
Mise en oeuvre

— Etude des lunettes HoloLens
— Etude des caméras de profondeur
— Récupération d'un nuage de points

Les lunettes HoloLens

Présentation de la démarche

— Récupération des données de profondeur d'un objet
— Construction d'une image 3D



Reconstruction d'un objet à partir d'un nuage de points

Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Projet Recherche & Développement

HoloLens, Réalité Augmentée et Epaule

Grégoire MARCO

Encadrement : Jordan NICOT et Mohand SLIMANE

Objectifs

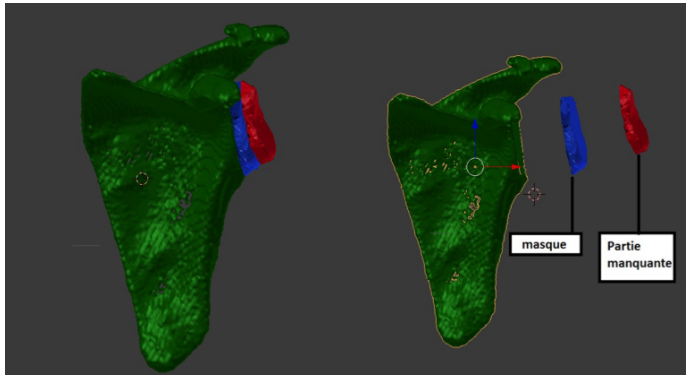
— Etude de la réalité augmentée
— Assister les chirurgiens pendant une opération à l'aide d'une application

Mise en oeuvre

— Etude des lunettes HoloLens
— Etude des caméras de profondeur
— Récupération d'un nuage de points

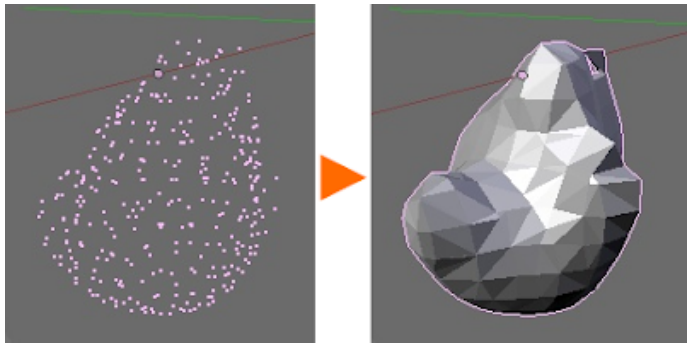
Présentation de la démarche

— Récupération des données de profondeur d'un objet
— Construction d'une image 3D



Omoplate et sa glène reconstituée virtuellement

Les lunettes HoloLens



Reconstruction d'un objet à partir d'un nuage de points

Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

HoloLens, Réalité Augmentée et Epaule

Résumé

Le projet "Hololens, Réalité Augmentée et Epaule" est la suite de plusieurs projets ayant pour finalité d'assister un chirurgien dans la pose d'une prothèse d'épaule pour remplacer la partie abîmée (la glène) par la maladie. Pour cela, nous utilisons les lunettes de réalité augmentée HoloLens. L'objectif est d'afficher en superposition à l'omoplate malade les parties manquantes reconstituées. Ce rapport présente la récupération d'un nuage de points d'une cible grâce aux lunettes Hololens. A la manière d'un scanner, le chirurgien pourra tourner autour de la glène pour en récupérer un modèle 3D.

Mots-clés

Réalité augmentée, HoloLens, omoplate, glène, Research mode, caméra de profondeur, nuage de points

Abstract

The project "Hololens, Réalité Augmentée et Epaule" is the continuation of many other projects whose the purpose is to assist a surgeon in the application of a prothesis of a shoulder to replace a diseased part (the glenoid). We use the augmented reality glasses HoloLens. The goal is to display missing parts over the disease scapula. This report presents how we get a points cloud from a target with HoloLens. Like a scanner, the surgeon will be able to move around the glenoid to get a 3D model.

Keywords

Augmented reality, HoloLens, scapula, glenoid, Research mode, depth camera, points cloud

Tuteurs académiques
Jordan NICOT
Mohand SLIMANE

Étudiant
Grégoire MARCO (DI5)