Scanner 3D

Axel JACQUOT

Responsable Pédagogique : Pierre AUBRY

2019

Table des matières

Table des figures

1 Présentation du projet

1.1 Abstract

Le développement rapide de solutions hardware et software dans le domaine du traitement de limage digital a propulsé la recherche en computer vision pour les applications industrielles. Le développement doutils électroniques innovants et la tendance baissière de leur prix rend possible de nouvelles avancées encore possible seulement dans nos imaginations il y a quelques décennies. Concrètement, la première étape de la création du modèle 3D dun objet réel est la capture des informations géométriques de la surface de lobjet physique. Ces objets réels peuvent être aussi petits quune pièce de monnaie et aussi grands quun bâtiment

1.2 Introduction

Le scan 3D est une technique permettant de capturer la forme dun objet en utilisant un outil scannant sa surface. Le résultat obtenu est un fichier 3D informatique pouvant être stocké, édité et imprimé.

Plusieurs technologies permettent de scanner en 3D des objets, environnements et personnes, chacune a ses avantages, ses inconvénients. Certaines technologies sont aussi capables de capturer en plus du volume et des dimensions, les couleurs et laspect du sujet numérisé, soit la texture. Compatibles avec la CAO et limpression 3D, le scanner 3D permet deffectuer de la rétro-ingénierie et à lheure actuelle, la modélisation en trois dimensions saccroît continuellement. En effet, on retrouve cet aspect dans un nombre domaines très variés :

Topographie : architecture, plans dintérieur et de façade, relevés industriels, ouvrage darts, conservation du patrimoine.

Conception : maquette de projets grande échelle, aéronautique, automobile, joaillerie.

Jeu vidéo : amélioration du graphisme vers un réalisme grandissant.

La technologie est aussi utilisée dans le médical, avec des applications dans le soin dentaire, par exemple.

Ce projet fait suite à un projet antérieur de scanner 3D.

1.3 Objectif

Pour cette version du projet nous travaillerons sur divers aspect, qui n'ont pas été abordé dans la première version du projet. Dans cette version, nous travaillerons sur différents point dans le but d'améliorer le projet :

- Amélioration de l'interface graphique.
- Importation des différents calculs sur carte graphique.
- Rajout d'une possibilité de scanner des pièces sans plateforme.
- Faire un scanner portatif.
- Scanner une cible peut importe sa taille

2 Etat de l'art

2.1 Technique de digitalisation 3D

Il existe six catégories pour le scan 3D.

2.1.1 Triangulation Laser

Dans ces six catégories, on retrouve deux techniques liées à lutilisation de la technologie laser : **la lasergrammétrie et la triangulation laser**. La première est connue puisque le capteur LIDAR
utilise ce principe : elle est basée sur le calcul de la durée mise par un laser entre laller et le retour à la surface étudiée. On utilise ici la vitesse de la lumière et un calculateur précis mesurant la durée du trajet. En répétant limpulsion un grand nombre de fois, on obtient une quantité dinformations nous permettant de déterminer la surface de lobjet grâce à ses variations de distance à la source du laser.

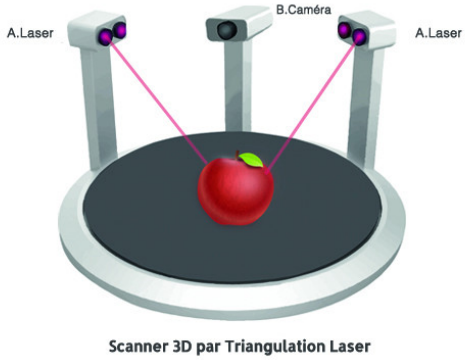


FIGURE 2.1 – Triangulation laser

Dans le cas de la triangulation laser, le processus repose sur trois éléments principaux : un laser, une caméra et lobjet à numériser.

Le laser vient se déformer au contact de la surface scannée, puis la caméra permet danalyser la déformation du signal laser sur les reliefs de lobjet et détermine grâce à des calculs trigonométriques sa position dans lespace. On obtient à la fin du scan un nuage de point permettant de reconstituer un objet 3D.
Cette méthode a un prix réduit et de nombreux projets DIY ont été basé sur ce choix. Elle permet dobtenir rapidement un résultat avec une précision de lordre de 0.01 mm.
Valable aussi pour le LIDAR, un majeur inconvénient est celui des surfaces transparentes ou réfléchissantes, le processus peut savérer fastidieux, mais contournable grâce à une poudre blanche, par exemple.

Sa portée limitée est aussi un défaut majeur puisque réduit la gamme dapplications possibles.

2.1.2 Lumière structuré

Le scan 3D par **lumière structurée** est une autre catégorie, utilisant des motifs de lumière projetée et une caméra. Il pointe sur le sujet une série de motifs géométriques (cercle, carré...) avec le projecteur de lumière et la déformation en fonction de la distance de la projection est capturée par la caméra décalée dun angle théta. Après un nombre déchantillons capturé suffisant, on utilise un calculateur déterminant la forme 3D de lobjet en fonction de lensemble des projections déformées analysées.

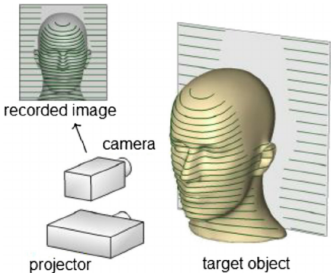


FIGURE 2.2 – Lumière Structuré

Bien plus rapide que le scan laser, la caméra na besoin de capturer que 12 images pour compléter le processus. Elle est

plus coûteuse que la méthode laser et dotée dune portée encore inférieure à celle-ci. Aussi, à faible distance et pour des objets de petite taille, la mise au point nest souvent pas possible pour un appareil abordable, il est donc préférable dutiliser un laser. Daprès des études comparatives faites sur un grand nombre dexpériences, les précisions des Scanner Laser et des Scanner par Lumière Structurée sont identiques en utilisant la même caméra.

2.1.3 Photogramétrie

Avec la **photogrammétrie**, les modèles 3D sont obtenus en utilisant des photos du sujet. Le principe est simple : on prend une série de clichés du sujet, et le logiciel trouvera ensuite les points de concordance. Grâce au décalage créé par les différents points de vue, il est alors possible de déduire la position des caméras et de créer un nuage de points de concordance en 3D.

FIGURE 2.3 – Photogrametrie

Malheureusement, dautres facteurs entrent en compte, compliquant le processus de calcul : la profondeur de champ gêne la détection des points, le bruit numérique est une entrave qui ralentit le calcul par le logiciel et lobjectif de lappareil photo applique des déformations quil faut analyser et compenser. Les logiciels ne sont vraiment apparus de façon pertinente que récemment pour ces raisons.

2.2 Projet existant

Logiciel de Microsoft.

Pour la Kinectv2, Microsoft a développé un logiciel pour les ordinateurs sous Windows 10, permettant de faire de la numérisation 3D. La configuration minimale pour lutilisation dun scan 3D via le logiciel est la suivante : un processeur deux curs ainsi quun processeur graphique et 4Go ram.

Lien

Projet sous la Raspberry-PI.

Un projet de Scanner 3D avec une Raspberry-PI et la Kinect a bien été tenté mais le problème de ce projet et le fait que la Raspberry ne fait quafficher le visuel de la caméra de la kinect.

Lien

Un autre projet de Scanner a cette fois été fait avec la technologie de triangulation laser : le système est composé dun laser 5mW, dune PiCam et dune Raspberry Pi. La Raspberry contrôle le système et renvoie les données sur un ordinateur qui traite la modélisation 3D.

Fabscan Pi Project

Ciclop.

Il sagit dun projet utilisant des lasers pour faire le scan 3D dun objet posé sur une plateforme. Les lasers permettent de récupérer des données composées dun nuage de point mais ces données sont stockées sur un fichier brut et ne sont pas directement traitées. Ce

qui fait que le pour le fonctionnement du scanner il faut en plus un ordinateur qui va faire le traitement des différentes informations. La précision reste assez pauvre même avec une taille maximum de 25cm.

Prix : 200 sur Amazon
Lien dun test du ciclop.

3D Sense.

Ce scanner 3d est portable mais doit être lié à un pc. Il na pas réellement de limite de taille pour la cible vu quil est portable. Ce scanner utilise comme technologie de la lumière Structurée. La précision reste un problème, tout comme le scanner précédent.

Prix : 469 surAmazon
Lien dun test du 3D Sense.

NextEngine Ultra HD.

Scanner pouvant être portable ou sur plateforme. Il na pas de limite de taille pour lobjet à scanner. Fonctionne à laide de triangulation laser. Grâce à sa définition il permet de prendre plus de détail sur la cible.

Prix : 2560 sur Aniwaa
Lien

3 Cahier des charges

3.1 Le besoin

Le projet doit aboutir à la conception dun système capable de scanner du gabarit dun être humain et den renvoyer une image 3D utilisable par un logiciel et une imprimante 3D. Tout le système doit être embarqué.

3.2 Fonctionnalités

Le projet devra répondre à plusieurs problématiques et répondre aux fonctionnalités suivantes :

Scanner une cible en 3D. (Principale)

— Le scanner doit capturer une image en 3 dimensions dune cible.
— La cible doit permettre au scanner de prendre une image nette.
— Le scanner doit retourner limage capturée selon le format souhaité par lutilisateur.
— Lutilisateur peut envoyer des ordres au scanner et lire son état.
— Peut scanner une cible de la taille dune figurine

Sadapter à des cibles de différentes tailles. (Secondaire)

— Le scanner peut scanner des objets de différentes catégories de taille.
— Lutilisateur peut changer la position du scanner par rapport à la cible.

Être autonome.

— Gestion de la taille de la cible partiellement ou totalement autonome.

Fiabilité et industrialisation.

— Coupler plusieurs capteurs pour optimiser la précision de limage.
— Utiliser des technologies fiables industrialisables.

Bonus

— Envoyer le fichier à une imprimante 3D.

3.3 Solutions technologiques

Pour répondre au mieux aux fonctionnalités listées plus haut, nous avons pensé à certaines solutions.

Scanner une cible en 3d (Principale)

— Le scanner doit capturer une image en 3 dimensions dune cible.
— Système permettant la capture de la cible en 3D.
— La cible doit permettre au scanner de prendre une image nette.
— La cible doit tourner sur la plateforme à une vitesse permettant une prise nette.
— Le scanner doit retourner limage capturée selon un format lisible.
— Le système est en capacité de stocker le fichier 3D.
— Lutilisateur peut envoyer des ordres au scanner et lire son état.
— Logiciel accessible pour nimporte quel utilisateur.

Sadapter à des cibles de différentes tailles (Secondaire)

— Le scanner peut scanner des objets de différentes dimensions.
— Lutilisateur doit pouvoir rentrer la taille de la cible et le logiciel est capable de déterminer la position que doit avoir le scanner

Fiabilité et industrialisation

— Coupler plusieurs capteurs pour optimiser la précision de limage.

— Optimisation de la prise 3D à laide de laser de ligne.
— Le scanner doit utiliser des technologies industrialisables.

Bonus

— Lié à une imprimante 3D directement.

3.4 Technologies possibles

Caméra de profondeur :

Caméra capable de détecter la profondeur de champ. Type Kinect ou Intel RealSense.

— Avantage(s) : Peut être facilement testé sur un PC.
— Inconvénient(s) : Résolution (plus la cible est grande, moins lobjet 3D sera réussi)

Impulsion Laser :

Mesure du temps de propagation dun laser entre son émission et sa réflexion pour la réception.

— Avantage(s) : Peut scanner de grandes choses.
— Inconvénient(s) : Lent.

Lumière Structurée :

Projection dune lumière connue sous forme de grille sur la cible à scanner. Consiste à mesurer la déformation de la lumière et grâce à cette déformation de recréer lobjet en 3D.

— Avantage(s) : Résolution, vitesse de scan, capable de scanner des personnes en détails.
— Inconvénient(s) : Sensible à la lumière. (peut fausser la mesure)

Triangulation Laser :

Projection dun laser et étude de sa trajectoire pour en déduire la forme de la cible.

— Avantage(s) : Résolution, Précision.
— Inconvénient(s) : Sensible à la surface (Brillant ou Transparent)

Pour plus dexplications : Lien

Nous avons fait le choix de prendre une caméra de profondeur pour effectuer la prise 3D car notre application se trouvera sur pc et donc nous auront des composants en moins pour effectuer la prise 3D.

3.5 Travail à effectuer

Dans cette nouvelle version du projet nous travaillerons sur plusieurs points, dans le but d'améliorer le projet actuel et de lui rajouter différentes fonctionnalités. Dans la première version du projet, le système marchait à l'aide d'une plateforme, les calculs 3D ne s'effectuaient pas sur une carte graphique, et il possédait aussi une interface utilisateur primaire.

Le but de cette nouvelle version est de porter dans un premier temps les différents calculs 3D sur carte graphique pour accélérer le processus de création du modèle 3D. Ensuite de rendre le système mobile, donc sans plateforme, que ce soit uniquement avec la caméra relié à un ordinateur ou via un système embarquant un écran et une carte pour que le système soit entièrement mobile. Un remaniement de l'interface graphique devra être effectuer pour permettre une meilleure expérience utilisateur. Pour finir une optimisation de la mémoire pour la conception des modèles 3D sera effectuer pour réduire la taille mémoire prise pour pouvoir l'embarquait.

4 Scanner 3D

4.1 Choix de la solution

Nous avons plusieurs technologies pour effectuer un scan 3D, que nous avons expliqué dans le rapport de groupe.

Lors du choix de la solution, nous avons eu plusieurs choix à faire pour pouvoir effectuer le travail demandé, ces choix sont le prix, la mise en uvre de la solution, le type de la technologie, la communauté travaillant sur cette solution. Différents types de technologies ont été présenté dans le rapport de groupe.

Nous avons sélectionné de prendre une caméra de profondeur comme technologies. Dans cette technologie nous avons deux produits qui ressortent qui sont la Kinect et les caméra RealSense de Intel.

4.1.1 Kinect v2

La Kinect est une caméra de profondeur développé par Microsoft. Microsoft a développé une Application pour ordinateur, pour le traitement des données de la Kinect. Sur la Kinect nous retrouvons une résolution pour la caméra de profondeur de 512 x 424 pixels et de $1920 * 1080$ pour la caméra couleur. Elle a une distance de profondeur de 4.5m au maximum et 50cm au minimum. Elle est capable de prendre 30 images par seconde. Elle a un prix de 100 et en plus il nous faut un adaptateur pour pouvoir la brancher qui est dans les 30, ce qui nous fait un total de 130.

FIGURE 4.1 – Kinect

4.1.2 Intel RealSense

Les caméras RealSense de Intel quant à eux pour la gamme D4. Cette gamme à un avantage, cet avantage est le fait que le traitement 3D se fait directement à lintérieur de la caméra, ce qui va nous permettre dalléger la charge de travail au niveau de la carte microcontrôleur principal.

Caractéristique	D415	D435	D435i
FOV	63.4ř * 40.4ř	85.2ř * 58ř	85.2ř * 58ř
Technologie du capteur	Rolling Shutter	Global Shutter	Global Shutter
Résolution de la caméra de profondeur	1280 * 720	1280 * 720	1280 * 720
Résolution de la caméra de couleur	1920 * 1080	1920 * 1080	1920 * 1080
Image par secondes maximum (fps)	90	90	90
Distance minimale pour la détection de la profondeur	0.16m	0.11m	0.11m
Distance maximale pour la détection de la profondeur	10m	10m	10m
Prix chez Intel	149 $	179 $	199 $

4.1.2.1 Global Shutter(Obstructeur Globale)et Rolling Shutter(Obstructeur Déroulant)

La différence entre les deux se trouve dans la méthode de captures dune image. Pour le Global Shutter limage est prise en une fois ce qui permet que limage prise ne soit pas déformée.Pour le Rolling Shutter, limage est prise ligne de pixels par ligne de pixels, ce qui déforme limage.

4.1.3 Solution choisie

Pour la solution, nous avons fait le choix de prendre la caméra RealSense D435 de Intel pour plusieurs raisons :

- Global Shutter : ça permettra de prendre des images netteslors de la rotation de la cible sur la plateforme et ainsi lorsque le sujet bouge lors du scan.Cela va nous permettre aussi de mettre moins de temps pour faire un tour car la plateforme marchera en continue.
- FOV : Cette caméra à un anglede vision plus grand ce qui permet de réduire la longueur du bras et la partie mécanique.

4.2 Installation de la caméra

4.2.1 Installation des utilitaires pour la caméra

Ajout des clés publiques :

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-key
C8B3A55A6F3EFCDE || sudo apt-key adv --keyserver
hkp://keyserver.ubuntu.com:80 --recv-key C8B3A55A6F3EFCDE
```

Ajout du serveur à la liste des dépôts :

```
sudo add-apt-repository "deb
http://realsense-hw-public.s3.amazonaws.com/Debian/apt-repo
bionic main" -u
```

Téléchargement des différents paquets pour la caméra RealSense :

```
sudo apt-get install librealsense2-dkms librealsense2-utils
librealsense2-dev librealsense2-dbg
```

4.2.2 Mise à jour du firmware de la caméra

Leur de la réception de la caméra Intel RealSense, nous avons dû installer le firmware pour le processeur se trouvant dans celle-ci. Sachant que nous travaillons se projet sur linux pour ce faire, nous utiliserons le terminal pour pouvoir linstaller.En premier lieu nous devons ajouter le dépôt Intel de la caméra à notre liste de dépôt, pour faire ceci dans le terminal nous devons taper la ligne suivante :

```
echo 'deb http://realsense-hw-public.s3.amazonaws.com/Debian/a
xenial main' | sudo tee /etc/apt/sources.list.d/realsensepubli
```

En second temps, nous devons ajouter la clé publique pour pouvoir accéder au dépôt, pour ce faire nous devons écrire dans le terminal la ligne suivante :

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-key 6F3EFCD
```

Une fois ceci fait, nous mettons à jour la liste des dépôts, avec la commande suivante :

```
sudo apt update
```

Maintenant, nous pouvons télécharger l'installeur du firmware à l'aide de la ligne suivante :

```
sudo apt install intel-realsense-dfu*
```

Pour installer le firmwaresur la caméra, nous devons récupérer le bus et leńdeviceżde celui-ci. La caméra est représentéesous le nom ńIntel Corp.ż. Pour récupérer ceci, il faut que nous tapions la commande suivante :

```
lsusb
```

Nous pouvons enfin passer à l'installation du firmware sur la caméra, pour ce faire nous utiliserons la commande suivante :

```
Intel-realsense-dfu -b "Bus" -d "Device" -f -I "Chemin du
firmware télécharger"
```

4.3 Choix du langage de programmation

Nous avons dû effectuer un choix de langage de programmation pour faire ce projet. Au début du projet, nous nous sommes dirigés vers le langage C++ pour lutilisation de la caméra mais en raison de manque de connaissance dans ce langage, nous avons fait le choix de changer de langage de programmation. Plusieurs choix soffraient à nous que ce soit du C, du LabVIEW, du Matlab, etc. Nous nous sommes dirigés vers le Python, qui est un langage utilisable pour la caméra, qui semblait être la meilleure solution entre ces différents langages. Donc la programmation devant se trouver sur lordinateur se fera avec le langage Python.

Lors de la première version du projet, nous avons utilisés le python dans sa version 2.7. Pour cette nouvelle version , nous avons changer la version de python pour passer en python 3, et ainsi pouvoir avoir un programme qui sera maintenable et améliorable dans le temps.

4.4 Présentation des librairies utilisées

Une librairie Python existe pour le fonctionnement de la caméra, il s'agit d'une librairie développé par Intel pour celle-ci.

```
pip3 install pyrealsense2
```

4.5 Fonctionnement de la caméra

LIntel RealSense D435 possède une caméra de type RGB, un diffuseur dinfrarouge, et deux récepteurs infrarouges. La profondeur est créée grâce aux deux récepteurs infrarouges car dans la caméra il se trouve un processeur qui fait différents calculs pour récupérer la profondeur à laide des deux images que créer les deux récepteurs. Ces calculs peuvent se faire grâce à la réflexion de linfrarouge.

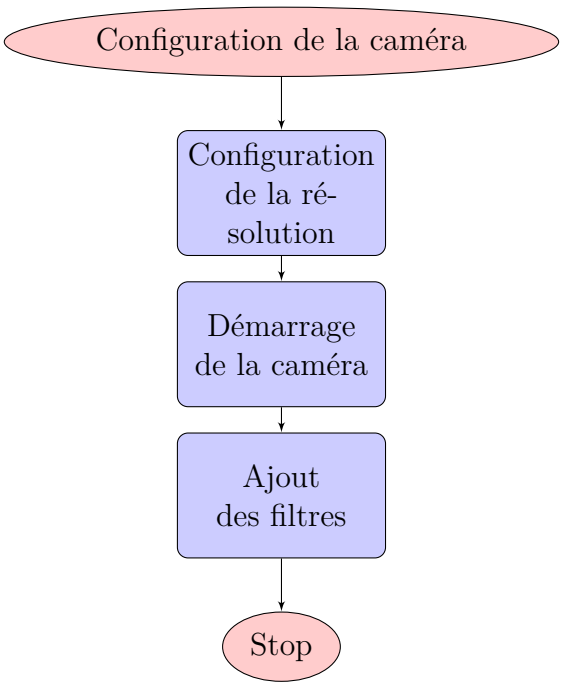
4.6 Utilisation de la caméra

Pour cette nouvelle version du scanner 3D, nous avons changé différente choses autour de la caméra. En premier temps nous avons créer une classe propre à la caméra, ce qui permet de mieux comprendre le fonctionnement de celle-ci. Nous avons grâce à la classe pu disperser chaque partit du fonctionnement de la caméra. Maintenant nous avons différentes parties

- Initialisation et Configuration de la caméra
- Récupération des données de la caméra
- Construction du modèle

4.6.0.1 Initialisation et configuration de la caméra

Dans cette nouvelle version, l'utilisateur peut changer la résolution de la caméra même quand celle-ci a déjà été utilisé, lors de test, ceci permet à l'utilisateur de réduire le nombre de données si il décerne un manque de puissance sur ordinateur ou avec sa carte embarqué.



4.6.0.2 Récupération des données de la caméra

Sur l'ancienne version du logiciel du scanner 3D, nous récupérions les données 3D de la caméra dans des fichiers de format "PLY". Ce choix, que nous avons effectué, rendait le programme lourd car nous travaillions avec des fichiers externe et non directement sur la mémoire vive de la cible. Donc nous avons du remanié notre code pour les données soit stockés directement dans la Ram. Pour faire ceci, nous avons utilisé une librairie Python

qui est numpy, elle nous a permit de créer une matrice pour stockées les donnée de sortie de la caméra pour ce faire nous sommes passés par cette ligne de code.

```
matrice = np.array(points.get_vertices(2))
# restructuration de la donnée dans une matrice 3*1
```

5 Création du Modèle 3D

5.1 Présentation des librairies utilisées

Pour effectuer les différentes opérations liées aux modèles 3D, nous avons avons besoin de différentes librairies python.

5.1.1 Numpy

Numpy est une librairie utilisé pour les calculs matriciels. Nous pouvons retrouver plus d'information sur cette librairie directement sur le site qui lui est dédié : https ://www.numpy.org/

5.1.1.1 Installation de Numpy

```
pip3 install numpy
```

5.1.2 Math

Math est une librairie de base de Python, elle permet d'effectuer différents calculs de mathématique (conversion des degrés vers les radians, les calculs trigonométriques, ...)

5.2 Amélioration

Dans la première version du projet, nous exportions les données 3D de la caméra dans un fichier 3D de type PLY que nous devions convertir dans un fichier 3D de type PCD, pour enfin le traiter. Le problème de la première solution était le fait de travailler avec différents fichiers, ce qui ralentissez le traitement des données. Ces fichiers étaient stockées directement sur la mémoire Flash. Le second point est que tout les calculs s'effectuer directement sur le processeur.

Dans cette nouvelle version les données seront stockées directement dans des matrices, ce qui permettra d'améliorer le temps de traitement des données 3D, pour améliorer encore le temps de traitement, nous déporterons les différents calculs 3D sur carte graphique

Les formats 3D que nous utilisons sont les mêmes que dans la 1ère version. Et la manière de les récupérer en sortie avec la conversion reste inchangé.

5.3 Rotation, Translation et Délimitation

Cette partie n'a pas été modifié depuis la 1ere version du projet. Pour avoir ces informations, veuillez-vous référer au rapport de la 1ere version.

5.4 Calculs sur carte graphique

Pour améliorer la partie traitement, nous devons déporter les calculs de rotations, de translations, et de limitations de la zone de traitement. Pour faire ceci, il existe plusieurs solutions avec différents langages de programmation pour cartes graphiques aussi appelées GPGPU .

5.4.1 Choix de la solution

5.4.1.1 GPGPU

GPGPU soit en manière non abrégé "General-Purpose computing on Graphics Processing Units" soit en français "Informatique polyvalente sur les unités de traitement graphique".

5.4.1.2 CUDA

Cuda est le langage de programmation pour carte graphique NVIDIA, ce langage de programmation à un problème c'est qu'il spécifique aux cartes graphiques NVIDIA, ce qui pose un problème. Mais il optimisé justement pour les architectures de NVIDIA, ce qui fait qu'il a des meilleures performances.

Le problème de CUDA, est le fait qu'il soit utilisable seulement avec les cartes NVIDIA, ce qui fera que les systèmes utilisant des cartes graphiques d'autres fabricants. Ce problème est le même pour tous les langages de programmation propriétaire à un fabricant. Donc par soucis de portabilité de la solution, ne nous orienterons pas vers cette solution.

5.4.1.3 OpenCL

OpenCL ou Open Computing Language est une API avec un langage de programmation dérivé du langage C. OpenCL est une API Open Source. Cette API est conçu pour la programmation de système parallèle, impliquant les CPUs et les GPUs, grâce à cette API nous pouvons exécuter des calculs intensifs.

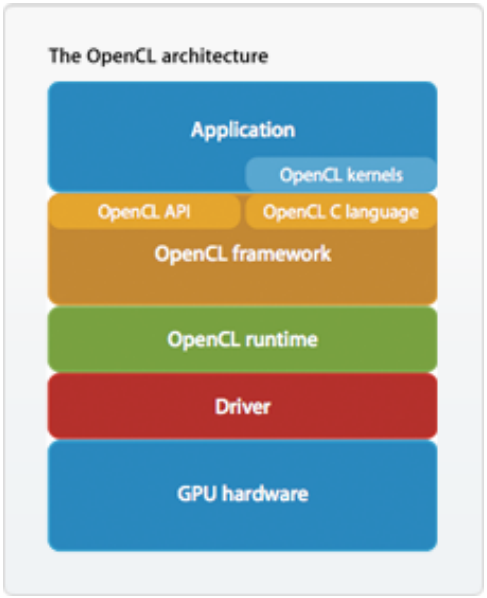


FIGURE 5.1 – Architecture OpenCL

Le développement OpenCL a plusieurs couches :

La première couche d'OpenCL est le code source du programme exécuté par le CPU

La seconde est le framework d'OpenCL, contenant l'API qui permet la communication avec le code source à l'aide de différente fonction, et nous retrouvons aussi le langage d'OpenCL permettant l'exécution des différents fonctions sur la carte graphique. Le framework permet de lié les deux parties pour l'exécution.

La troisième est la partie de l'exécution du code sur la carte graphique.

La quatrième et la cinquième sont propres au GPU, le driver permet la communication entre OpenCL et la carte graphique.

Comme dit précédemment, le langage de programmation d'OpenCL est un dérivé du langage C, sauf que celui-ci doit respecter différentes contraintes supplémentaire. Une fonction avec le langage d'OpenCL commence par kernel.

5.4.2 Application de la solution

Pour effectuer les calculs sur la carte graphique, nous avons fait le choix d'OpenCl car il est disponible sur plus de plateforme que le Cuda, qui est propre seulement au carte de Nvidia, alors qu'OpenCL est disponible peut importe le fabriquant de la carte graphique. Pour qu'OpenCL fonctionne correctement nous avons différente contrainte à suivre.

5.4.2.1 Création du contexte

Nous devons d'abord attribuer un processeur graphique à OpenCL, pour que celui-ci puisse fonctionner. Pour ce faire il faut nous créions un contexte qui servira lors de l'exécution des différentes fonctions à executer. Dans ce contexte nous stockerons les données de la carte graphique choisit.

5.4.2.2 Fonction OpenCL

Pour les fonction OpenCL, il faut que toute les données que nous lui envoyons, pour le langage Python, soit déclarer en pointeur dans les paramètres de la fonction. Nous devons les déclarer en pointeurs car chaque donnée en python est stockés dans une donnée pointé.

Dans une fonction OpenCL, que nous voulons éxecuter sur un processeur graphique, nous devons aussi récupérer l'indice de la donnée pour pouvoir la traité. La manière de récupérer cette indice va différer selon le nombre de donnée a traités. Pour trouver cette index, il faut que nous connaissions le nombre de données à traiter issue d'un seul pointeur.

Dans notre cas pour l'index que nous devons fournir à la fonction, ce sera pour l'utilisation de 3 données sur le pointeur. Ces données correspondront aux cordonnées de x, y et z.

Pour chaque variable qui va être utiliser, nous devons les stocker dans un buffer, fait par une fonction OpenCL et ces données seront stockées sur la mémoire de la carte graphique, où nous devrons signaler si la variable est en écriture ou en lecture. Pour s'assurer d'un bon fonctionnement, nous devons changer le type des différentes variables à l'aide de la librairie numpy, qui a des types de données se rapprochant du langage C. Nous devrons aussi rajouter le contexte d'OpenCL, la carte graphique sur laquelle le travail va s'effectuer. Ce buffer va permettre de pointer sur les données à stocker sur la carte graphique.

Pour la récupération des différentes données, nous passerons par une fonction OpenCL qui permet de faire un pont entre les données stockées sur la carte graphique et les données utilisées sur le programme Python.

Pour l'utilisation des fonctions programmer en OpenCL, nous devrons utiliser le pont permettant de lié les données sur le programme Python ainsi que celle de la carte graphique et lui rajouter le nombre de données totales qui est à traiter. A la suite, nous devrons rajouter les différentes variables, utiliser par la fonction souhaité et dans le même ordre, créer par le biais de la fonction de Buffer.

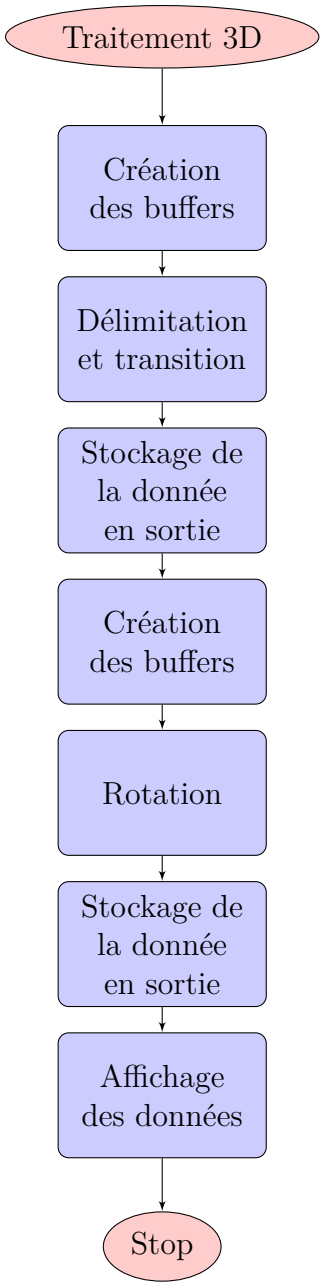
5.5 Changement entres les versions

Sur la première version, nous exécutions les calculs avec des calculs matricielles. Dans la nouvelle version, nous exécutons les calculs de façon explicite car dans OpenCL nous n'avons pas accès à des librairies comme Numpy, pour faire des calculs matricielles.

Lors de la première version du projet, les différents calculs que nous devions exécuter prenait plus de 3 minutes à s'exécuter. Avec la nouvelle version et pour le même fonctionnement, création d'un modèle via la plateforme, cela s'effectue en 14 secondes pour la partie logiciel donc s'en prendre en compte le déplacement de la plateforme.

Cette amélioration a été faite pour l'utilisation du scanner de façon mobile. Ceci à permit d'effectuer différents test pour pouvoir le faire ultérieurement.

5.6 Création du modèle



6 Affichage 3D

Pour la partie affichage 3D, nous devons choisir une méthode qui nous permettra d'afficher le modèle sous forme 3D. Pour faire ceci il existe différentes API qui existent comme Direct3D, OpenGL ou Vulkan (qui est basé sur OpenGL). Ce type dAPI, nous permettra aussi de déplacer les différents calculs graphiques sur un processeur graphique et ainsi daméliorer la durée des calculs. Les processeurs graphiques sont aussi plus adaptés à la gestion des calculs 3D.

6.1 Processeur Graphique ou GPU

Un processeur graphique permet dassurer les calculs daffichage et 3D. Il sagit dune structure parallèle formé de plusieurs curs permettant deffectuer plusieurs tâches en même temps. Ces curs ne sont pas aussi performants que les CPUs mais plus nombreux donc ils sont utilisés pour effectuer des calculs 3D ou daffichages car les calculs sont moins lourds mais plus nombreux.

6.2 Choix de l'API pour l'affichage 3D

6.2.1 Direct3D

Direct3D est un outil de rendu 3D, disponible uniquement sur Windows. Ce qui n'est pas un avantage pour notre projet sachant que le projet doit pouvoir fonctionner sur Windows, Linux.

6.2.2 OpenGL

OpenGL est un outil de rendu 3D se trouvant la plupart du temps déjà intégrée au GPU, elle permet dutiliser le processeur graphique pour effectuer les différents calculs pour laffichage.

6.2.3 Vulkan

Vulkan est une API graphique, il sagit de lAPI qui succède à OpenGL. Elle a plusieurs avantages comparer à OpenGL :

— Meilleur Gestion du multi-processing
— Meilleur Gestion de la mémoire
— Calcul non graphique pouvant être fait sur le GPU

Mais cette API est aussi plus difficile à prendre en main.

Nous avons fait le choix de partir sur l'API OpenGL pour sa facilité de prise en main, les différents tutoriels présents sur l'internet global et la communauté qui l'entoure.

6.3 Application de la solution

Nous nous sommes donc dirigés vers OpenGL pour effectué l'affichage pour ce faire nous devions déterminer le meilleur moyen pour afficher un modèle 3D sans couleur, et de manière la plus légère possible. Pour effectuer un affichage avec OpenGL, il faut que l'affichage que nous souhaitons faire soit une classe que ce soit dans le langage Python ou C++. Dans cette classe nous auront une fonction pour l'affichage, et des fonctions pour les activités de la souris pour effectuer la rotation, pour fixer la données que nous avons récupérer avec la caméra.

6.3.1 Affichage du modèle 3D

Pour effectuer l'affichage du modèle nous avons recherchés différentes manière de faire, pour nous diriger vers celle qui nous convenait le plus. Grâce à ces différentes recherches,

nous nous sommes orientés vers une solutions utilisant une fonction d'OpenGL utilisant des pointeurs de matrice pour stocker les données, se trouvant sous forme de matrice. Ensuite une fois les données stockés nous grâce à une autre fonction d'OpenGL, nous afficherons un point aux coordonnées de chaque matrices.

Code d'exemple :

```
# display 3D model
glEnableClientState(GL_VERTEX_ARRAY)
glVertexPointer(3, GL_FLOAT, 0, self.vert)  #Récupération des données
      pour l'affichage
glDrawArrays(GL_POINTS, 0, self.vert.shape[0])  #Affichage des
      données
glDisableClientState(GL_VERTEX_ARRAY)
```

— Pour la fonction glVertexPointer, en paramètre nous devons lui mettre la taille de la matrice, le type des coordonnées, le pas entre chaque donnée, et les données à pointer.
— Pour la fonction glDrawArrays, en paramètre nous lui mettons le moyen d'afficher la données, le pas entre chaque données et la nombre totale de données à afficher.

Dans OpenGL, il existe plusieurs moyen d'afficher une donnée, que ce soit en point, en ligne, en carré ... . Dans notre cas, nous sommes dirigés vers un affichage en point car les données que nous utilisons ne sont pas obligatoirement dans le bonne ordre. Si nous avions choisis d'effectuer un affichage avec des lignes, le modèle aurait été faux car une données peut aussi bien se trouvait à 50 cm à la droite de la caméra et la suivant à 50 cm à la gauche de celle-ci. Donc l'affichage avec des points est celui qui est le plus efficace dans notre cas.

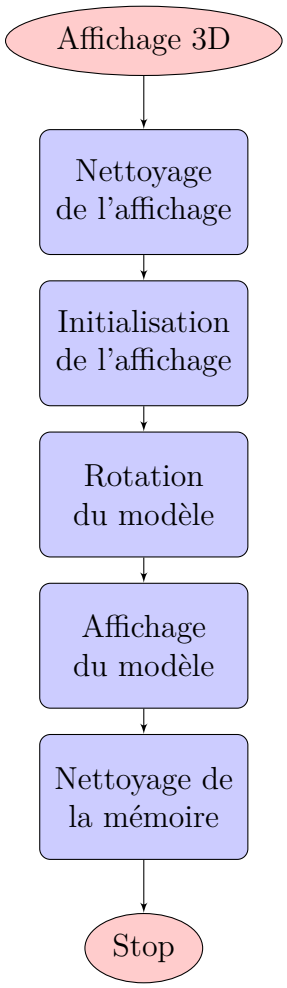
6.3.2 Rotation du modèle 3D

Pour la rotation du modèle, elle s'effectuera à l'aide de la souris. L'utilisateur lors de l'appuie du bouton gauche, pour effectuer une rotation autour du modèle 3D.

OpenGL inclut les interactions avec le clavier et la souris, il nous suffit juste de récupérer le nom que l'on doit donner à ces différentes fonctions pour avoir ces différentes interactions.

Une fois que nous avons récupérer la rotation à effectuer, lors de l'affichage du modèle 3D, nous effectuons les différentes rotations que nous avons récupérer avec l'aide d'une fonction propre à OpenGL. Cette fonction se nomme "glRotated", grâce au paramètres que nous lui mettons l'axe de rotation va changer.

6.3.3 Alogrigramme



7 Plateforme

Cette partie est celle qui a le moins bougé. Le matériel présentait dans la 1ere version reste le même, donc pour avoir ces informations vous devez vous référer au 1er rapport.

Nous avons fait une pièce pour maintenir la caméra sur la plateforme, avec l'aide de Romain Lux Quach. Si l'utilisateur souhaite utiliser une autre plateforme, ceci est possible car la partie de construction du modèle n'est pas paramétré pour cette plateforme. Il devra juste respecter le fait d'utiliser une carte STM32 pour le contrôle de sa platefrome.

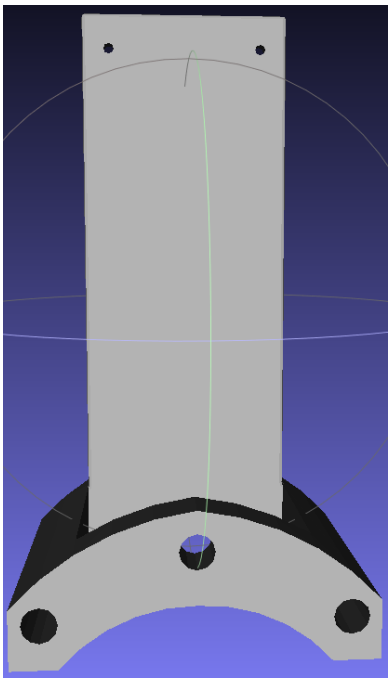


FIGURE 7.1 – Holder Caméra

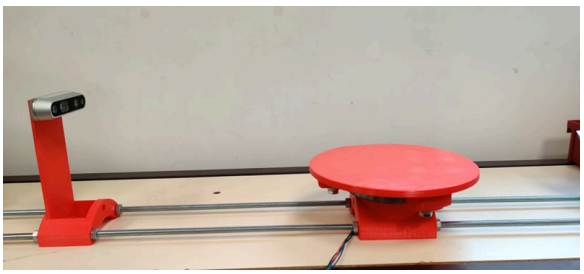


FIGURE 7.2 – Plateforme

8 Scanner mobile

Pour cette nouvelle version du projet, le fait de développer une version mobile du scanner. Cette version permettra ainsi de pouvoir scanner des cibles sans réelles limites de tailles.plus grandes. Lors de la réflexion de cette version, nous en sommes convenues qu'il devrait y avoir deux versions, la première version serait-elle connecté directement à un ordinateur et la seconde serait une version que l'on pourrait tenir comme une télécommande, avec un écran tactile, une carte électronique embarquant un processeur et un processeur graphique ainsi que des ports USB 3. Ces deux versions devront aussi embarqué un accéléromètre et un magnétomètre pour connaître les rotations et les translations effectuer par le système pour la reconstruction du modèle 3D.

8.1 Mobilité

Pour faire le scanner 3D en mobile, nous avons besoin de connaître les rotations et les translations que le système effectue. Pour ce faire nous avons besoin d'un magnétomètre, accéléromètre et d'un gyromètre.

Nous avons plusieurs solutions pour créer cette mobilité. La première est de prendre une carte embarquant les trois solutions et la rajouter à la caméra, sachant qu'il faut que la solution fonctionne avec l'ordinateur et avec une carte embarqué. La seconde solution est la caméra D435i embarquant le système.

8.1.1 Externe à la caméra

Pour la solution externe nous devrons rajouter un capteur pouvant nous donner la rotation, la translation. Ce qui rajoutera différent composant à rajouter à l'utilisateur pour que le système fonctionne.

8.1.2 Interne à la caméra

Pour la solution interne, nous pouvons utiliser la caméra Intel RealSense D435i, intégrant un gyroscope, magnétomètre et accéléromètre directement avec la caméra. Cette solution rendra plus simple l'utilisation du système car si l'utilisateur souhaite une solution simple, il se dirigera vers celle-ci car il n'aura besoin que d'un ordinateur et de la caméra pour la version la plus simple.

8.2 Écran tactile

Pour la version télécommande, nous aurons besoin d'un écran tactile pour que l'utilisateur puissent utiliser le logiciel dédié au scanner.

8.3 Bouton

Pour le bouton, il permettra le démarrage et l'arrêt du scanner 3D, ce sera un bouton On/Off.

8.4 Carte de développement

Pour faire un système entièrement mobile, nous avons besoin d'une carte électronique pouvant effectuer le travail qu'effectue l'ordinateur. Cette carte devra donc embarquer un port USB 3 pour la caméra, de connexions pour l'écran tactile ainsi que pour le bouton. Elle devra aussi embarqué un processeur graphique pour les différents calculs de créations de modèles qui s'effectuent sur celle-ci.

9 Logiciel

Ce projet ayant plusieurs versions, il a eu différentes interfaces.

9.1 Choix de l'interface

9.1.1 Terminal

La première interface était une interface faites sur le terminal. Cette interface a été développé pour pouvoir effectuer des tests sur le fonctionnement du système. Cette version sous terminal fonctionnait sous formes de menu dans le but de tester les différents points du projet.

9.1.2 ImGUI

La seconde interface qui a été développé est une interface sous l'API ImGUI. Au cours du développement de celle-ci, nous nous sommes aperçus que ImGUI avait quelque défauts, même ci elle nous permettait d'effectuer l'interface que nous souhaitions faire. Le défaut le plus important est le fait que des qu'un objet est déclarer (menu ou affichage 3D) une nouvelle fenetre soit créer ce qui n'est pas idéal.

9.1.3 Qt

Nous nous sommes diriger vers le GUI Qt, qui correspondait à ce que nous souhaitons faire comme types d'interfaces autant pour la tablette que pour l'ordinateur. Elle nous permet de lancer différentes connexions via les différents widget, permettant diverse actions. Elle fonctionne aussi grâce à différents actions, pouvant s'exécuter en "même temps", ce qui permet d'effectuer l'affichage 3D ainsi que l'interface, et la gestion du scanner 3D.

9.2 Interface

Lors de la conception de l'interface nous nous avons rechercher ce que l'interface logiciel devrait être capable d'effectuer. Nous en avons donc sortie une liste de différentes actions qu'elle devrait être capable de réaliser.

- Affichage 3D
- Choix de la résolution de la caméra
- Choix des différents fichiers 3D en sortie
- Sauvegarde des fichiers 3D
- Choix du mode de fonctionnement
  - Contrôle du scan
  - Paramétrage de la modélisation
  - Connexion de la caméra et de la plateforme

9.2.1 Affichage 3D

Pour l'affichage 3D, nous utiliserons l'affichage développé avec l'API OpenGL. Pour pouvoir utiliser ce type d'affichage sur Qt, nous devrons créer une classe ayant comme parent une librairie de Qt qui est QOpenGLWidget, qui permet de créer des widget pour les affichages faits avec OpenGL. Ceci permettra d'utiliser les signaux et les slots présents dans Qt pour le transfert de donnée pour l'affichage du modèle.

9.2.2 Choix de la résolution de la caméra

La caméra possédant plusieurs résolutions utilisables, nous avons fait le choix de laisser le choix à l'utilisateur de celle-ci. Ceci aura un effet sur le nombre de point 3D à traiter et ainsi sur la qualité du scan. Mais permettra une utilisation plus accessible pour les ordinateurs et cartes embarquées moins puissantes.

9.2.3 Choix des différents fichiers 3D en sortie

Le projet étant un scanner 3D, nous devions mettre en place un moyen pour que l'utilisateur puisse choisir les différents types de fichiers qu'il souhaitait en sortie pour le modèle 3D capturé. Pour ce faire, nous avons utilisé la librairie Qt, "QCheckBox", qui permet à l'utilisateur de cocher les différents fichiers qu'il souhaite et ainsi les avoir en sortie lors de la sauvegarde du modèle.

9.2.4 Sauvegarde des fichiers 3D

Nous avons dû mettre en place, un moyen de sauvegarde pour les différents fichiers 3D. Une fois que l'utilisateur a choisit les différents formats que l'utilisateur souhaite, il peut choisir un emplacement de sauvegarde spécifique afin que la récupération des fichiers se fasse le plus facilement. Pour que l'utilisateur puisse choisir l'emplacement de sauvegarde, nous utiliserons "QFileDialog" qui permet d'ouvrir une interface de gestion de fichier, que ce soit sur Windows ou sur Linux, et de sélectionner un dossier. De notre côté, nous récupérons le chemin du fichier pour effectuer les différentes sauvegardes. Pour ouvrir l'interface de gestion de fichier, l'utilisateur devra appuyé sur un bouton, correspondant à un "QPushButton" sur Qt.

Une fois que l'utilisateur a choisit, son emplacement de sauvegarde et les formats 3D qu'il souhaite. Un bouton pour la sauvegarde final du fichier se trouvera sur l'interface pour le mode Plateforme et pour le mode Mobile. Pour le fonctionnement :

- mode Plateforme : Pour que la sauvegarde s'effectue, il faut que le scan de la cible soit fini à 100%.
- mode Mobile : Pour effectuer une sauvegarde dans ce mode, il faut que l'utilisateur arrête le scan et ensuite il pourra sauvegarder.

Pour chaque condition nécessaire à la sauvegarde, si elles ne sont pas respectées un message d'erreur apparaîtra à l'écran.

9.2.5 Connexion de la caméra

Un bouton se trouve sur l'interface permettant la connexion de la RealSense. Lors de l'appuie du bouton, une fonction est déclenché qui permet de connecter la caméra. Après cela un message s'affichera à l'écran pour dire à l'utilisateur si la caméra est connecté ou non.

9.2.6 Choix du mode de fonctionnement

L'utilisateur lors de l'utilisation du logiciel à le choix entre différents modes. Ces modes sont le mode plateforme, le mode mobile, et un mode test pour la plateforme. Le choix du mode va permettre de modifier une partie de l'interface, cette partie de l'interface contient un champs pour la configuration de la limitation du champs de vision de la caméra, et un autre champs pour la gestion propre à cette partie. Le choix du mode se fera à l'aide d'une liste correspond à "QComboBox" sur Qt. Pour le changement de l'interface nous le ferons à l'aide de "QStackedWidget".

9.2.6.1 QStackedWidget

"QStackedWidget" permet un changement de widget. Dans notre cas pour les paramètres du mode. Pour ce faire nous devons créer différents widgets auquel nous attribuerons chaque composants nécessaires. Ensuite nous devrons rajouter les widgets dans un ordre précis correspondant à la disposition présente dans la liste. Car le changement de l'interface se fera grâce à l'index de la liste.

Exemple :

```
self.stacked_gl = QStackedWidget(self)
self.stacked_gl.addWidget(self.Logo_Qt_widget)
```

```
self.stacked_gl.addWidget(self.model_view_widget)
self.stacked_gl.addWidget(self.camera_view_widget)
```

9.2.7 Mode Plateforme

9.2.7.1 Limitation

Pour la plateforme nous avons plusieurs limitation à prendre en compte. Pour chaque limitation nous utiliserons le widget "QDoubleSpinBox". Les limitations que nous aurons concernerons :

— l'angle de rotation de la plateforme
— l'axe Y dirigé vers le haut : pour pouvoir limiter selon la taille de la cible
— l'axe Y en dirigé vers le bas : pour pouvoir limiter selon la hauteur du plateau
— l'axe Z : Pour limiter la profondeur de champs
— la distance jusqu'au centre du plateau

Ces différentes limitations permettent aussi de limiter le nombre de données à traiter car elles vont permettre de réduire la taille de l'image 3D.

9.2.7.2 Gestion Scan

Sur cette partie, nous aurons accès à différents boutons ayant leurs actions propres.

— Connexion de la plateforme : Exécute une fonction permettant la connexion de la plateforme. Cette fonction doit être changer selon la plateforme utilisée.
— Lancement du scan
— Arrêt du scan
— Sauvegarde du modèle

9.2.8 Mode Mobile

9.2.8.1 Limitation

Pour la plateforme nous avons plusieurs limitation à prendre en compte. Pour chaque limitation nous utiliserons le widget "QDoubleSpinBox". Les limitations que nous aurons concernerons :

— l'axe X
— l'axe Y
— l'axe Z

Ces différentes limitations permettent aussi de limiter le nombre de données à traiter car elles vont permettre de réduire la taille de l'image 3D.

9.2.8.2 Gestion Scan

Sur cette partie, nous aurons accès à différents boutons ayant leurs actions propres.

— Lancement du scan
— Arrêt du scan
— Sauvegarde du modèle

9.2.9 Mode Test

9.2.9.1 Limitation

Pour la plateforme nous avons plusieurs limitation à prendre en compte. Pour chaque limitation nous utiliserons le widget "QDoubleSpinBox". Les limitations que nous aurons concernerons :

— l'angle de rotation de la plateforme
— l'axe Y dirigé vers le haut : pour pouvoir limiter selon la taille de la cible
— l'axe Y en dirigé vers le bas : pour pouvoir limiter selon la hauteur du plateau
— l'axe Z : Pour limiter la profondeur de champs
— la distance jusqu'au centre du plateau

Ces différentes limitations permettent aussi de limiter le nombre de données à traiter car elles vont permettre de réduire la taille de l'image 3D.

9.2.9.2 Gestion Scan

Sur cette partie, nous aurons accès à différents boutons ayant leurs actions propres.

— Lancement du test

— Arrêt du test

9.2.10 Affichage

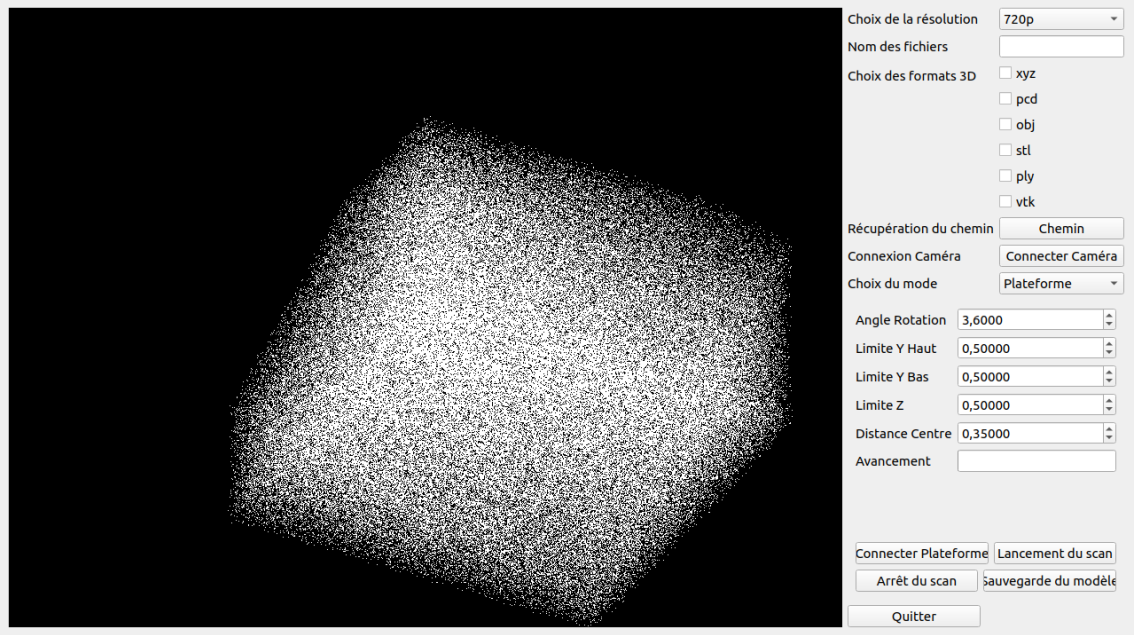


FIGURE 9.1 – Affichage du logiciel

10 Conclusion

Lors de ce projet, j'ai pu appréhender différentes API comme OpenGL et OpenCL et donc l'utilisation de la carte graphique. Le projet n'est pas encore entièrement abouti pour le programme python, il nous reste encore quelque correctif à mettre en place pour avoir une version plus complète.

Nous avons commencé à travailler sur la partie mobile du scanner 3D, donc sur carte. Pour cette nouvelle version, nous devons traduire le code python que nous avons fait en code C++, ceci permettra de réduire l'empreinte mémoire du projet et la consommation de celle-ci.

Avec les améliorations que nous avons effectué que soit pour l'interface, les calculs sur carte graphique et l'affichage 3D. Ceci a rendu dans un premier temps, le projet plus utilisable par un utilisateur lambda, et en le passant en C++, ça le rendra encore plus accessible car au final nous aurons un exécutable directement utilisable englobant tout ce que nous utilisons pour l'utilisateur.

11 Bibliographie

Lien vers le programme du scanner en Python : Lien

Lien vers le 1er rapport du projet : Lien

Lien vers le projet en C++ : Lien

Lien vers le dépôt GitHub : Lien