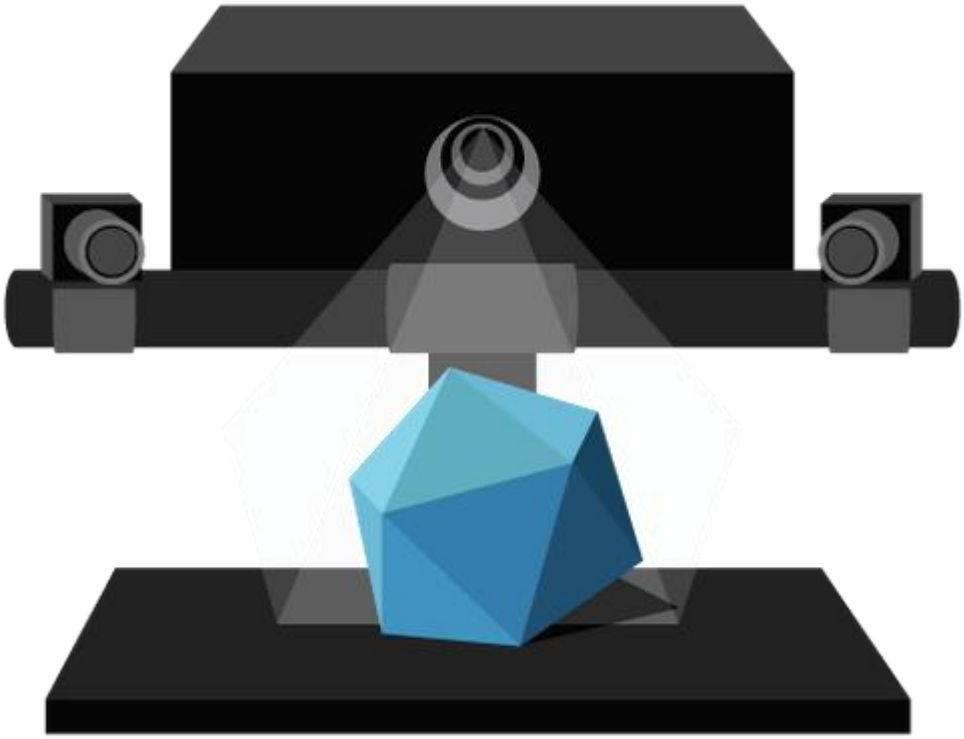
Rapport du projet
Scanner 3D



Axel Jacquot

M1 SER

ynov AÉRONAUTIQUE & SYSTÈMES EMBARQUÉS

ynov CAMPUS

Table des matières

Présentation du projet

Abstract

Le développement rapide de solutions hardware et software dans le domaine du traitement de l'image digital a propulsé la recherche en computer vision pour les applications industrielles. Le développement d'outils 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 d'un objet réel est la capture des informations géométriques de la surface de l'objet physique. Ces objets réels peuvent être aussi petits qu'une pièce de monnaie et aussi grands qu'un bâtiment

Introduction

Le scan 3D est une technique permettant de capturer la forme d'un 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 l'aspect du sujet numérisé, soit la texture. Compatibles avec la CAO et l'impression 3D, le scanner 3D permet d'effectuer de la rétro-ingénierie et à l'heure actuelle, la modélisation en trois dimensions s'accroît continuellement. En effet, on retrouve cet aspect dans un nombre domaines très variés :

Topographie : architecture, plans d'intérieur et de façade, relevés industriels, ouvrage d'arts, 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.

Nous discuterons des différentes techniques de numérisation d'objets trois dimensions, des potentielles avancées et des avantages et inconvénients des méthode étudiées. Six grandes catégories peuvent être étudiées, toutes reposant sur des principes différents.

Objectif

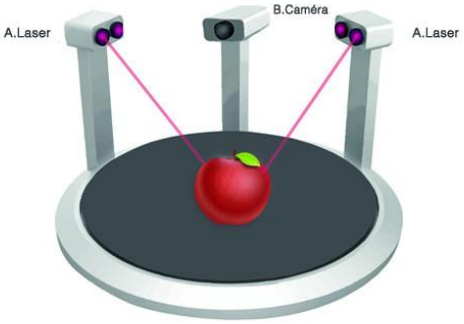
Après avoir fait un état de l'art et analysé les différentes solutions existantes, faire un choix de technologie ou de croisement de technologie à utiliser pour avoir un scan 3D automatisé d'objets d'une taille réglable par l'utilisateur avec un maximum fixé. Ce scanner devra embarquer une carte microcontrôleur pour effectuer les calculs nécessaires à l'analyse de la surface du sujet et la reconstitution du nuage de points, dans un format 3D au choix par l'utilisateur.

État de l'art

Technique de digitalisation 3D

Dans ces six catégories, on retrouve deux techniques liées à l'utilisation 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 l'aller 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 l'impulsion un grand nombre de fois, on obtient une quantité d'informations nous permettant de déterminer la surface de l'objet grâce à ses -variations de distance à la source du laser.

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

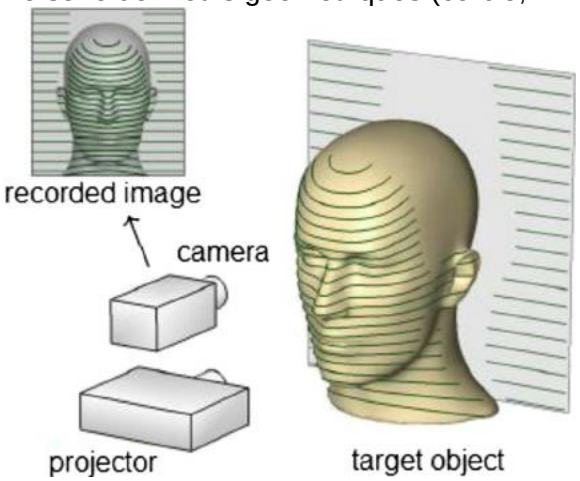


Scanner 3D par Triangulation Laser

Le laser vient se déformer au contact de la surface scannée, puis la caméra permet d'analyser la déformation du signal laser sur les reliefs de l'objet et détermine grâce à des calculs trigonométriques sa position dans l'espace. 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 d'obtenir rapidement un résultat avec une précision de l'ordre de 0.01 mm.
Valable aussi pour le LIDAR, un majeur inconvénient est celui des surfaces transparentes ou réfléchissantes, le processus peut s'avérer fastidieux, mais contournable grâce à une poudre blanche, par exemple.

Sa portée limitée est aussi un défaut majeur puisque réduit la gamme d'applications possibles.

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 d'un angle thêta. Après un nombre d'échantillons capturé suffisant, on utilise un calculateur déterminant la forme 3D de l'objet en fonction de l'ensemble des projections déformées analysées.



Bien plus rapide que le scan laser, la caméra n'a besoin de capturer que 12 images pour compléter le processus. Elle est plus coûteuse que la méthode laser et dotée d'une portée encore inférieure à celle-ci. Aussi, à faible distance et pour des objets de petite taille, la mise au point n'est souvent pas possible pour un appareil abordable, il est donc préférable d'utiliser un laser. D'après des études comparatives faites sur un grand nombre d'expériences, les précisions des Scanner Laser et des Scanner par Lumière Structurée sont identiques en utilisant la même caméra.

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.

Malheureusement, d'autres 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 l'objectif de l'appareil photo applique des déformations qu'il faut analyser et compenser. Les logiciels ne sont vraiment apparus de façon pertinente que récemment pour ces raisons.

La méthode est très peu coûteuse et donne d'excellents résultats pour un débutant. Pourtant, elle n'est pas la méthode "go to", il est par exemple impossible de scanner avec précision un objet en mouvement avec une seule caméra, le coût peut donc être vite multiplié. Un mauvais éclairage, qu'il soit trop faible ou trop intense empêchera l'acquisition d'un modèle 3D précis et cohérent avec le sujet initial.

Le **scan 3D par contact** réalise plusieurs mesures d'une surface en la touchant. On en déduit les informations avec les variations de relief et un fichier 3D est créé. Très peu utilisé pour les projets à faible coût car très cher à automatiser, il est doté d'une très grande précision si les moyens sont mis en place.

La **radargrammétrie** constitue une technique permettant de prendre des mesures de l'aller retour d'une onde radio grâce à sa vitesse et la distance au sujet. Similaire au LIDAR, la différence réside dans la longueur d'onde utilisée. (126 nm - 10,6 µm pour le LIDAR ; 2 cm pour la radargrammétrie.

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 l'utilisation d'un scan 3D via le logiciel est la suivante : un processeur deux cœurs ainsi qu'un 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 qu'afficher 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 d'un laser 5mW, d'une PiCam et d'une Raspberry Pi. La Raspberry contrôle le système et renvoie les données sur un ordinateur qui traite la modélisation 3D.

Fabscan Pi Project

Ciclop.

Il s'agit d'un projet utilisant des lasers pour faire le scan 3D d'un objet posé sur une plateforme. Les lasers permettent de récupérer des données composées d'un 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 d'un test du ciclop.

3D Sense.

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

Prix : 469€ sur Amazon

Lien d'un test du 3D Sense.

NextEngine Ultra HD.

Scanner pouvant être portable ou sur plateforme. Il n'a pas de limite de taille pour l'objet à scanner. Fonctionne à l'aide de triangulation laser. Grâce à sa définition il permet de prendre plus de détail sur la cible.

Prix : 2560€ sur Aniwaa

Lien

Cahier des charges

Le besoin

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

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 d'une cible.
- La cible doit permettre au scanner de prendre une image nette.
- Le scanner doit retourner l'image capturée selon le format souhaité par l'utilisateur.
- L'utilisateur peut envoyer des ordres au scanner et lire son état.
- Peut scanner une cible de la taille d'une figurine

S'adapter à des cibles de différentes tailles. (Secondaire)

- Le scanner peut scanner des objets de différentes catégories de taille.
- L'utilisateur 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 l'image.
- Utiliser des technologies fiables industrialisables.

Bonus

- Envoyer le fichier à une imprimante 3D.

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 d'une 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 l'image capturée selon un format lisible.
- Le système est en capacité de stocker le fichier 3D.
- L'utilisateur peut envoyer des ordres au scanner et lire son état.
- Logiciel accessible pour n'importe quel utilisateur.

S'adapter à des cibles de différentes tailles (Secondaire)

- Le scanner peut scanner des objets de différentes dimensions.
- L'utilisateur 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 l'image.
- Optimisation de la prise 3D à l'aide de laser de ligne.
- Le scanner doit utiliser des technologies industrialisables.

Bonus

- Lié à une imprimante 3D directement.

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 l'objet 3D sera réussi)

Impulsion Laser :

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

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

Lumière Structurée :

Projection d'une 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 l'objet 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 d'un 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 d'explications : Lien // Etat de l'art

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.

Changement de cahier des charges effectué

Au début, nous étions trois personnes à devoir travailler sur ce projet. L'objectif de départ de ce projet était de scanner un être humain et de faire fonctionner tout le projet grâce à différentes cartes électronique et donc de ne pas faire le projet pour l'ordinateur. Ces différents objectifs ont été revus à la baisse du fait que je sois pour le faire.

Explication du projet

Pour ce projet, nous devons faire un scanner 3D. Il sera composé d'une caméra de profondeur, d'une plateforme rotative et d'un ordinateur qui devra contrôler le tout. Pour ce faire, nous devrons effectuer une étude sur une caméra de profondeur, sur la création d'un modèle 3D, la conception de la plateforme, un affichage 3D et tous autres éléments qui tourne autour de ce style de projet.

Scanner 3D

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 œuvre 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.

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€.

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 l'intérieur de la caméra, ce qui va nous permettre d'alléger la charge de travail au niveau de la carte microcontrôleur principal.

	D415	**D435**
FOV	63.4° * 40.4°	85.2° * 58°
Sensor Technology	Rolling Shutter	Global Shutter
Résolution de la caméra de profondeur	1280 * 720 pixels	1280 * 720 pixels
Images par secondes (fps)	90	90

Distance minimale pour la profondeur	0.16m	0.11m
Distance maximale pour la profondeur	10m	10m
Résolution de la caméra couleur	1920 * 1080 pixels	1920 * 1080 pixels
Prix	149$ soit 130.032€	179$ soit 156.213€

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

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

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 nettes lors 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 angle de vision plus grand ce qui permet de réduire la longueur du bras et la partie mécanique.

Installation de la caméra

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 de la realsense :

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

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 l'installer. 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/aptrepo xenial main' | sudo tee /etc/apt/sources.list.d/realsensepublic.list
```

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 6F3EFCDE
```

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

```
sudo apt-get update
```

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

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

Nous téléchargeons le firmware de la caméra, pour ce faire nous allons sur le site suivant :

```
https://downloadcenter.intel.com/download/28573/Latest-Firmware-for-Intel-RealSense-D400-Product-Family?v=t
```

Pour installer le firmware sur la caméra, nous devons récupérer le bus et le « device » de celui-ci. La caméra est représentée sous le nom « Intel Corp. ». 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'
```

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 l'utilisation 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 s'offraient 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 l'ordinateur se fera avec le langage Python.

Présentation des librairies utilisées

Une librairie en Python est disponible pour la communication avec la caméra.

```
pip install pyrealsense2
```

Fonctionnement de la caméra

L'Intel RealSense D435 possède une caméra de type RGB, un diffuseur d'infrarouge, 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 l'aide des deux images que créer les deux récepteurs. Ces calculs peuvent se faire grâce à la réflexion de l'infrarouge.

Utilisation de la caméra

Configuration de la résolution :

La caméra propose plusieurs résolutions et taux d'images, un programme en C++ présent sur le dépôt GitHub de la caméra propose de tous les afficher selon le modèle qui est branché sur l'ordinateur lors de l'exécution de celui-ci.

Dans cette configuration, nous utilisons la caméra de profondeur ainsi que la caméra couleur. Les deux caméras sont réglées à une résolution 720p avec un taux d'images de 30fps.

```
config = rs.config()
# Configure Stream Depth
config.enable_stream(rs.stream.depth, 1280, 720, rs.format.z16, 30)
#other_stream, other_format = rs.stream.infrared, rs.format.y8
other_stream, other_format = rs.stream.color, rs.format.rgb8
# Configure Stream Color
config.enable_stream(other_stream, 1280, 720, other_format, 30)
```

Lancement du streaming :

Ces deux lignes permettent de lancer le streaming de la caméra dans la configuration dans laquelle celle-ci a été régler.

```
# Start streaming
pipeline = rs.pipeline()
pipeline.start(config)
```

Récupération des images :

Pour la récupération des images provenant de la caméra, nous devons d'abords attendre la réception des images et ensuite nous récupérons deux images, une pour la profondeur et une autre pour la couleur.

```
#attnte des frames
success, frames = pipeline.try_wait_for_frames(timeout_ms=0)
#récupération de la frame profondeur
depth_frame = frames.get_depth_frame()
#récupération de la frame couleur
color = frames.get_color_frame()
```

Post-Processing :

La librairie de la caméra fournit une fonction permettant de faire du post-processing. Cette fonction applique différents filtres aux images en sorties de la caméra pour la lisser et améliorer la qualité de celle-ci.

Test de la caméra

Pour appréhender la programmation de la caméra, je me suis aidé des différents programmes que Intel fournit sur le GitHub de la caméra. Il fournisse différents codes pour

pouvoir tester différents aspects de l'utilisation de la caméra, ces programmes permettent aux développeurs de comprendre le fonctionnement de la caméra car il n'y a pas de documentation pour la caméra.

Après avoir appréhender les différents aspects, j'ai adapté les différents programmes pour pouvoir tester différentes configurations, comme la modification de la résolution, du nombre d'images par seconde ainsi que d'autres différentes fonctions que nous pouvons faire avec la caméra.

Capture du modèle 3D

Avec cette caméra, nous pouvons effectuer la capture d'un modèle 3D. Pour ce faire, nous utiliserons les images que prend la caméra. Dans la librairie de celle-ci, il existe une fonction qui permet de calculer la position des points 3D grâce à l'image de la caméra de profondeur.

```
points = pc.calculate(depth_frame)
```

Avec ces coordonnées nous pouvons les exporter dans un fichier PLY

```
points.export_to_ply(ply, color)
```

La capture de ce que visualise la caméra vient d'être effectué.

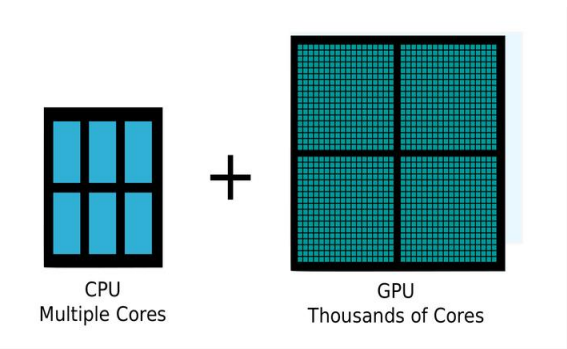
Algorigramme :

Application 3D

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

Processeur Graphique ou GPU

Un processeur graphique permet d'assurer les calculs d'affichage et 3D. Il s'agit d'une structure parallèle formé de plusieurs cœurs permettant d'effectuer plusieurs tâches en même temps. Ces cœurs ne sont pas aussi performants que les CPUs mais plus nombreux donc ils sont utilisés pour effectuer des calculs 3D ou d'affichages car les calculs sont moins lourds mais plus nombreux.



Choix de l'API graphique

OpenGL :

OpenGL est une API graphique se trouvant la plupart du temps déjà intégrée au GPU, elle permet d'utiliser le processeur graphique pour effectuer les différents calculs pour le traitement 3D et pour l'affichage.

Vulkan :

Vulkan est une API graphique, il s'agit de l'API 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

Après différents tests avec les deux API, nous avons choisi que pour l'affichage 3D, nous utiliserons OpenGL pour effectuer cette tâche.

Librairies utilisées

Pyglet :

La librairie Python pyglet, est la librairie OpenGL mais dédié au Python. Donc elle est utilisée pour faire des affichages 3D. Elle s'installe par le biais de la commande suivantes :

```
pip install pyglet
```

Open3D :

Open3D est une librairie Open Source pour le Data Processing 3D. Elle est utilisable en C++ et en Python. Elle permet de travailler sur différents aspects de la 3D comme la visualisation, la reconstruction … Nous pouvons l'installer via la ligne de commande suivante :

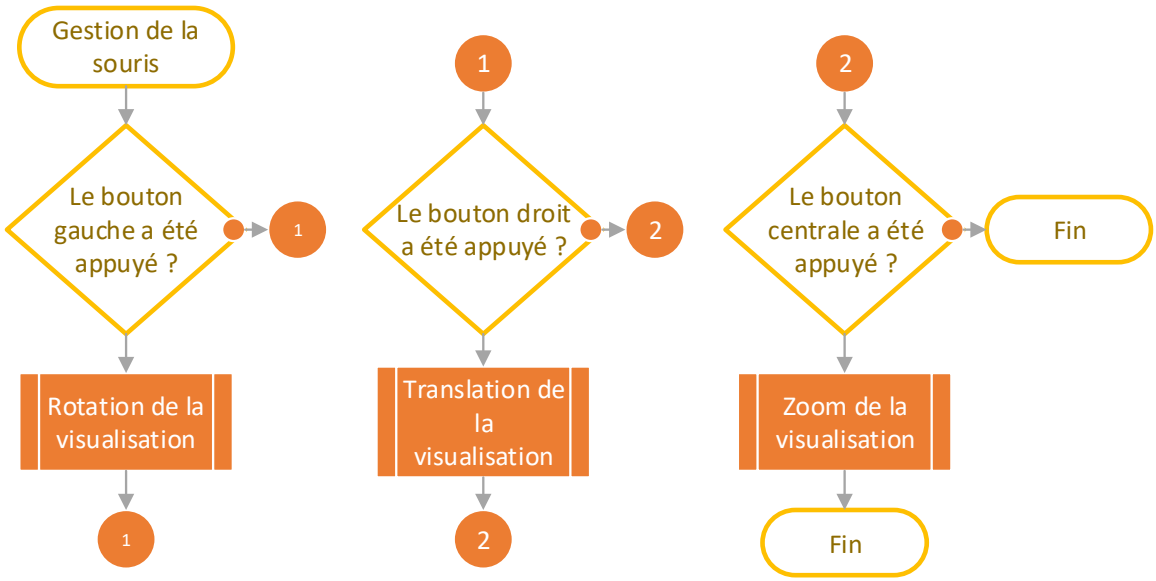
```
pip install open3d-python
```

Affichage 3D du point de vue de la caméra

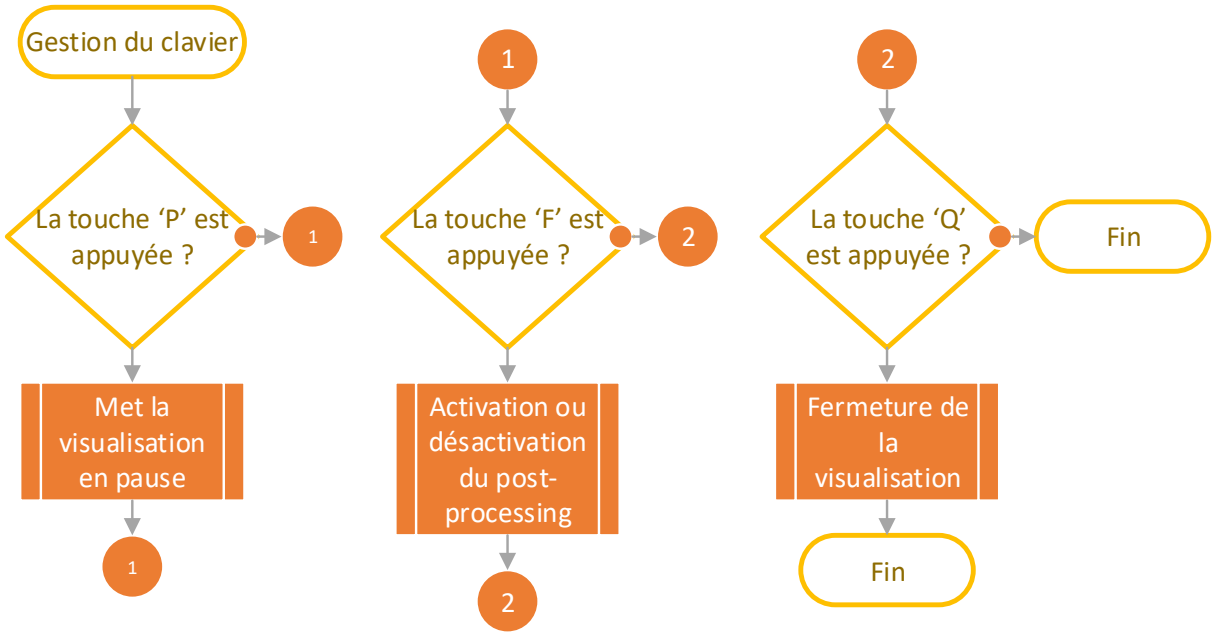
Nous avons modifié une application, faite pour la caméra, pour qu'elle convienne à mes besoins, qui pour le moment permet de voir en 3D, le point de vue de la caméra. Sur cette application nous pouvons nous déplacer à l'aide de la souris (rotation, translation et zoom). Ainsi que gérer différents paramètres de traitement de la caméra ou du logiciel à l'aide du clavier. Cette affichage marche grâce à pyglet (OpenGL).

Algorigramme :

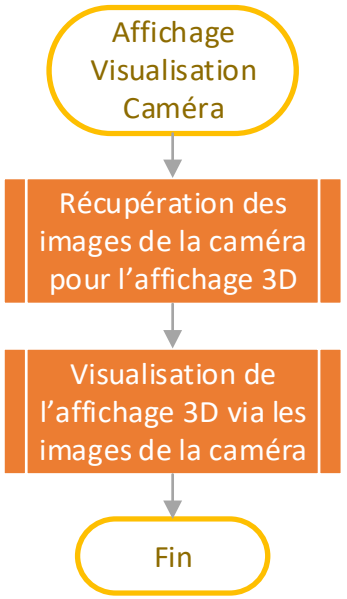
Gestion de la souris :



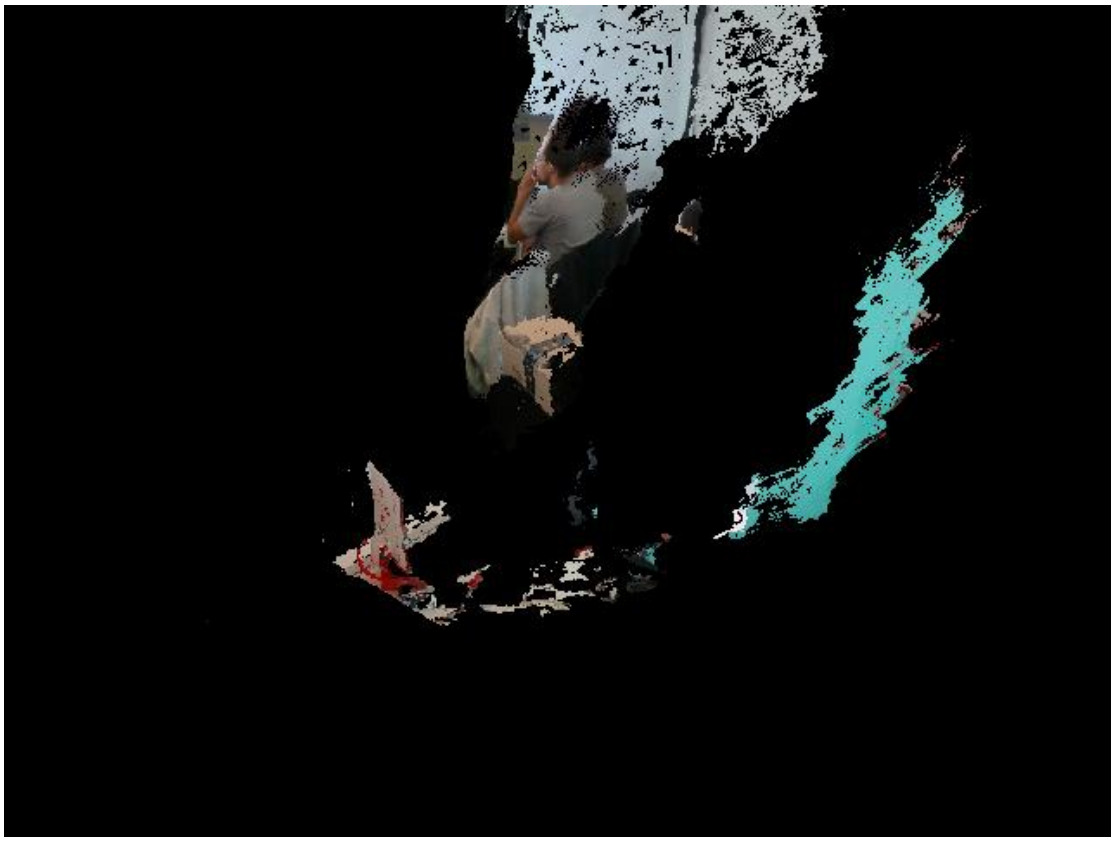
Gestion du clavier :



Affichage :



Test :



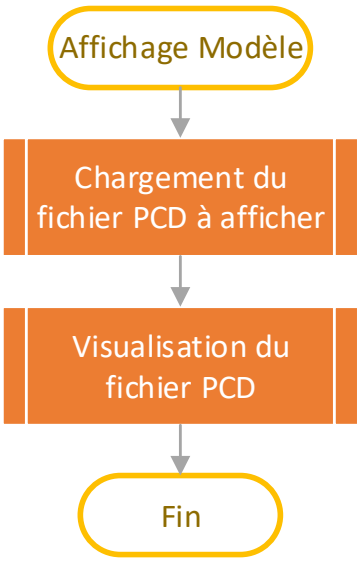
Affichage des modèles 3D en format PCD

Pour l'affichage des modèles 3D qui se trouve sous le format PCD, nous utiliserons la librairie Open3D. Cette librairie contient une fonction qui permet d'afficher un fichier PCD. Pour utiliser cette fonction nous devons une fonction pour charger le fichier que nous voulons visualiser et après lui passer en paramètres de la fonction de visualisation.

La fonction de chargement du fichier, permet de mettre le fichier que nous voulons charger sous le bon format, ainsi que de limiter le nombre de point qui seront charger lors de l'affichage et ainsi de réduire l'impact de l'affichage pour certaines configurations.

La fonction de visualisation marche avec l'aide d'OpenGL et permet uniquement d'afficher des nuages de points qui sont charger dans le bon format au préalable.

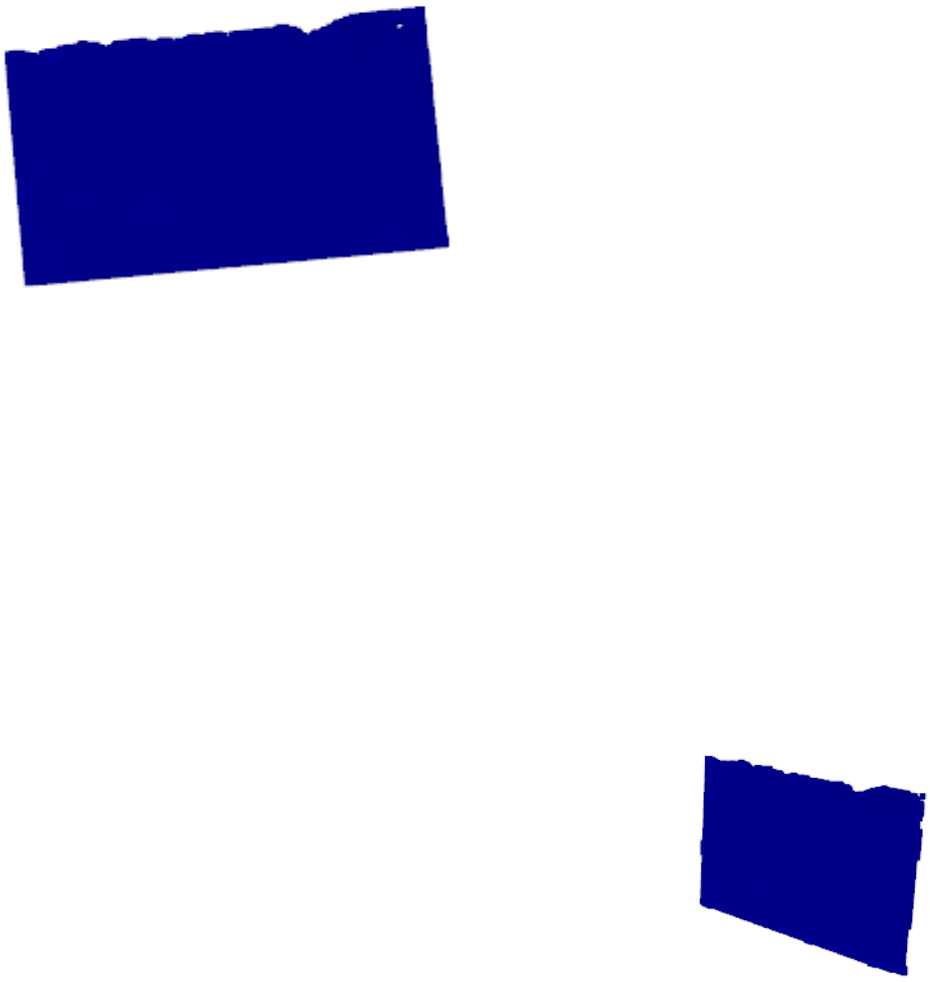
Algorigramme :




```
from open3d import *

pcd = read_point_cloud("test_0.pcd")
draw_geometries([pcd])
```

Test :

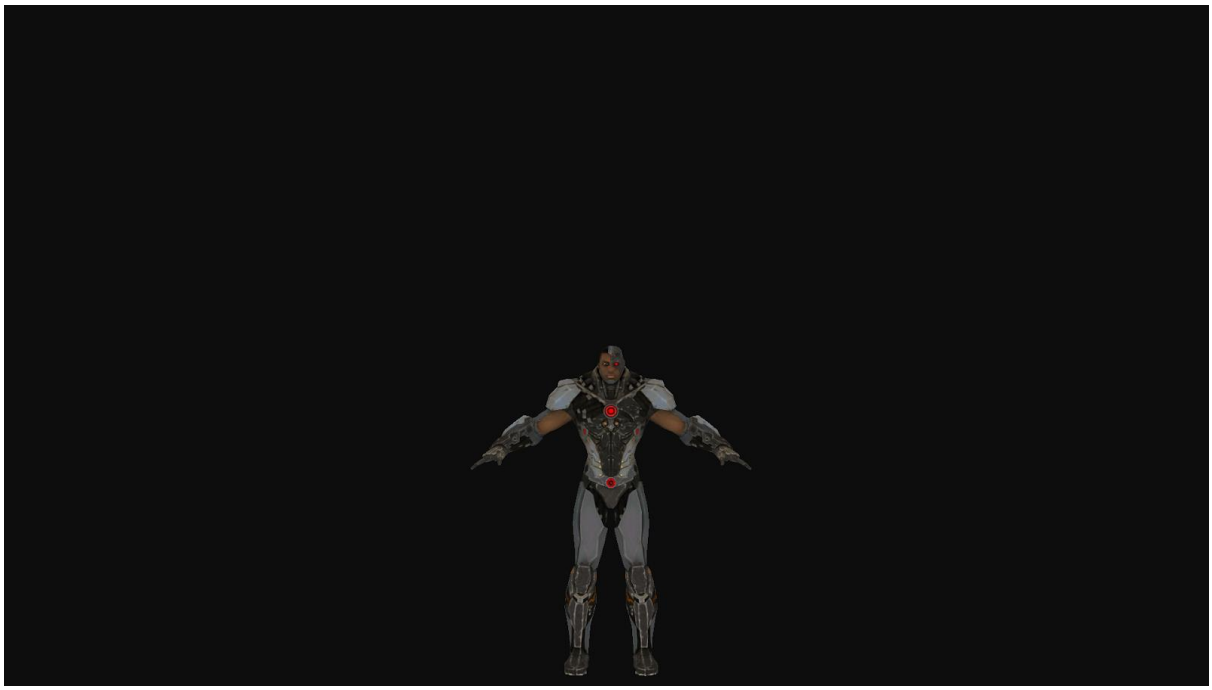


Le point représente le centre de la pièce soit la caméra dans notre cas.

Application d'affichage de modèle 3D :

Lors des débuts du projet, nous avons effectué une prise en main d'OpenGL sous le langage C++ via différent tutoriel se trouvant sur internet car nous n'avions jamais vu cela avant. De cette prise en main, en est sortie un affichage 3D de modèle se trouvant sous le format OBJ. Ce programme devait afficher les modèles 3D complets à la base mais travaillant sous format PCD mais ayant une autre solution pour afficher ce type de fichier 3D, nous n'utiliserons pas ce programme pour effectuer cette fonction. Et ce programme avait un défaut qui est le fait qu'il prenne seulement les fichiers OBJ pour effectuer l'affichage et l'utilisateur pouvant choisir différent format et pas seulement l'OBJ, nous aurions dû convertir à chaque fois le fichier dans le format que choisit l'utilisateur et en OBJ juste pour l'affichage.

Résultat :



Création du modèle 3D

Différent type de fichier 3D :

PLY :

Signifiant « Format de fichier de polygones ». Ce format fut conçu pour stocker des données 3D provenant principalement de scanner 3D. Il peut stocker la couleur en RGB et la transparence, le nombre de points, les coordonnées des point …

PCD :

Signifiant « Format de nuage de point » est un format Open-Source développé dans le cadre de la librairie PCL. Il permet de stocker les coordonnées des différents points, le nombre de point ainsi que la couleur pour chaque point.

OBJ :

Signifiant « Objet 3D ». Ce fichier est l'un des types de fichier 3D, les plus utilisés. Dans ce format les formes géométriques peuvent être stockés sous formes de polygones ou de surface lisses.

STL :

Signifiant « Stéréolithographie », qui est un procédé 3D, utilisé principalement dans la fabrication d'objet à partir d'un modèle numérique 3D. L'objet est stocké sous forme de triangle, qui sont définit par des coordonnées cartésiennes et ainsi avec ces différents triangles forme une surface.

STEP :

Signifiant « Norme d'échange de produit », ce format est propriétaire et ne peux donc pas être utilisable par une application autres que celle qui en ont les droits. Ce format est spécial car le modèle 3D est décrit de façon complète, précise et non ambiguë.

VTK :

Signifiant « Boîte à outils de visualisation », ce format est utilisé pour 'infographie 3D, le traitement de données et la visualisation. Les données 3D, sous ce format, sont stockés sous notions de maille.

Présentation des librairies utilisées

Pour effectuer les différentes opérations sur les modèles 3D, nous avons eu besoin de plusieurs librairies.

Numpy :

Il s'agit d'une librairie Python pour les calculs scientifiques comme les matrices.

Le site de la librairie se trouve ici : https://www.numpy.org/

Installation de la librairie :

```
pip install numpy
```

Pcl :

La librairie PCL (Point Cloud Librairy soit Librairie de Nuage de Point) est une librairie Open Source, permettant le travail sur des structures 3D. La documentation pour cette librairie se trouve ici : http://www.pointclouds.org/

Le GitHub de la librairie principal (C++) se situe à cette adresse : https://github.com/PointCloudLibrary/pcl

Le GitHub pour la librairie pour le Python : https://github.com/strawlab/python-pcl

Installation de la librairie Python :

Sous linux :

```
sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl -y
sudo apt-get update -y
sudo apt-get install libpcl-all -y
pip install --upgrade pip
pip install cython==0.25.2
pip install numpy
git clone https://github.com/strawlab/python-pcl.git
cd python-pcl
python setup.py build_ext -i
python setup.py install
```

Sous Windows:

```
pip install python-pcl
```

OS :

La librairie OS est une des librairies de base du langage Python, elle s'installe en même temps que Python. Cette librairie permet de faire la connexion entre l'OS et Python (Lecture de fichier, exécution de commande …).

Math :

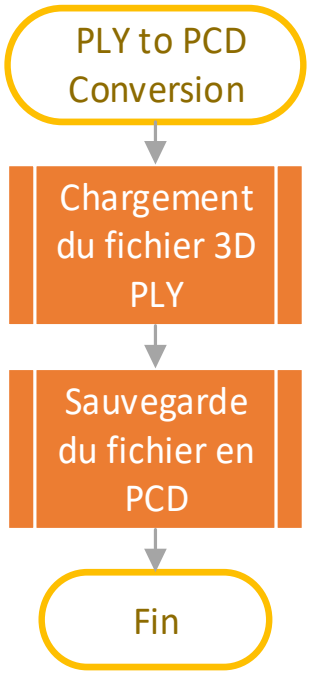
Math est aussi une librairie de base de Python, elle permet d'effectuer différents calculs de mathématique (conversion degré vers radian, calcul trigonométrique …).

Conversion du type de modèle 3D

PLY vers PCD :

Pour la conversation du format PLY vers le format PCD, nous le ferons à l'aide la librairie PCL sous Python. Dans cette librairie, il y a une fonction qui permet de la conversion d'un fichier PLY vers PCD, le défaut de cette fonction est le fait qu'elle ne prend pas en compte la couleur lors de la conversion. Nous faisons cette conversion pour pouvoir travailler avec les coordonnées des points, car dans le fichier PLY, ces coordonnées sont encodées en binaires.

Algorigramme :



Cet Algorigramme correspond à ce code :

```
import pcl    #Importation de la librairie PCL

#Chargement du fichier se trouvant dans le format PLY
clouding = pcl.load("fichier à charger en ply")
#Sauvegarde du fichier sous le format PCD
pcl.save(clouding, "fichier de sortie en format pcd")
```

PCD ou PLY vers OBJ, STL … :

Pour la conversion du modèle complet nous le ferons à l'aide d'un programme se trouvant dans librairie PCL mais sous le langage C++. Donc nous devons pouvoir faire une simulation d'un terminal pour lancer ce programme avec les paramètres que nous voulons lui soumettre. Nous avons fait le choix de faire ceci, car dans la version actuelle la librairie cette fonction de conversion n'est pas disponible et nous n'avions pas le temps de recréer un programme permettant de le faire. Ce programme permet de convertir un fichier 3D en OBJ, STL, VTK, PLY, PCD

Ce programme se trouve à cette adresse :

Simulation d'un terminal :

Concernant la simulation du terminal nous utiliserons la librairie OS, vu qu'elle permet de lancer des commandes comme-ci nous nous trouvions dans un terminal. Pour ce faire nous utiliserons en Python les commandes suivantes :

```
import os          #Importation de la librairie permettant l'exécution

cmd ='./pcl_converter "Fichier à convertier" "Fichier de sortie de
conversion"'       #Création de la commande à exécuter

os.popen(cmd)      #Exécution de la commande
```

Code du fichier C++ :
https://github.com/PointCloudLibrary/pcl/blob/master/io/tools/converter.cpp

Lecture et écriture des données contenu dans le fichier 3D

Pour la lecture et l'écriture dans un fichier, nous utiliserons la librairie OS. Cette librairie permet de lire et d'écrire dans n'importe quel type de fichier. Cet aspect nous arrange car le type de fichier avec lequel nous allons travailler est le type PCD. Ce mode change selon un paramètre de la fonction d'ouverture qui est :

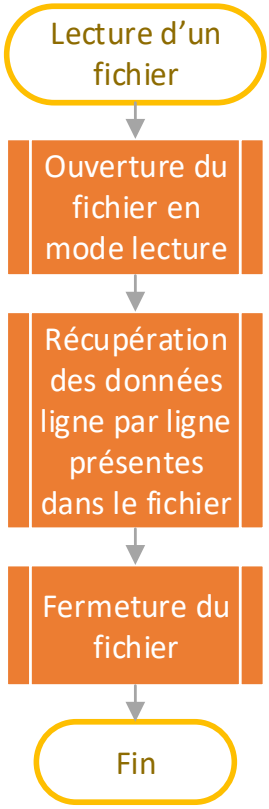
```
var = open("Fichier que l'on souhaite utilisé", "mode d'ouverture")
```

Il existe trois modes d'ouverture de fichier.

Lecture d'un fichier :

La lecture d'un fichier se fait assez facilement avec cette librairie. Elle permet de soit lire le fichier caractère par caractère ou ligne par ligne. Dans notre cas nous utiliserons la lecture ligne par ligne. Chaque ligne est en fait stockés dans une case d'une liste et une ligne se termine à la lecture d'un caractère '\n' signifiant saut de ligne.

Algorigramme :



Cette fonction permet l'ouverture en lecture d'un fichier, où nous lisons ligne par ligne, le contenu présent dans celui-ci. Chaque ligne est stockée dans un élément d'une liste. Une fois la lecture terminée, nous refermons le fichier pour qu'il soit réutilisable.

Code exemple :

```
import os #Importation de la librairie OS
#Ouverture du fichier en mode lecture
#file_read contient l'intégralité du fichier
file_read = open("Fichier a lire", "r")
#Stockage du contenu du fichier ligne par ligne
# dans une liste
contenu = file_read.readlines()
#Fermeture du fichier lu
file_read.close()
```

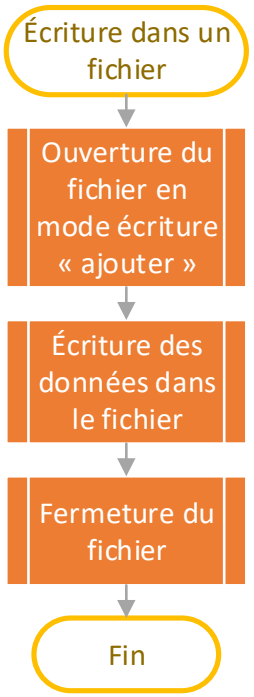
Ecriture d'un fichier :

Il existe deux moyens d'écriture dans la librairie OS, l'un permet de rajouter à la suite dans le fichier et l'autre efface l'intégralité du fichier pour pouvoir écrire dedans.

Écriture avec ajout à la fin du fichier :

Dans ce mode le fichier dans lequel nous écrivons n'est pas écrasé à l'écriture. Ce mode d'écriture peut être utilisé pour la complétion d'un fichier.

Algorigramme :



Cette fonction permet l'ouverture d'un fichier en mode d'écriture « ajouter », nous pouvons écrire dans le fichier et chaque élément sera ajouter à la suite des autres. Une fois l'écriture terminé, nous n'avons plus qu'à refermer le fichier.
Pour faire un saut à ligne et retour au début de la ligne, nous avons juste à lui écrire le caractère '\n'. Nous pouvons écrire lettre par lettre, mot par mot ou encore un texte complet dedans cela ne change rien pour lui tant qu'il s'agit d'une chaine de caractère.

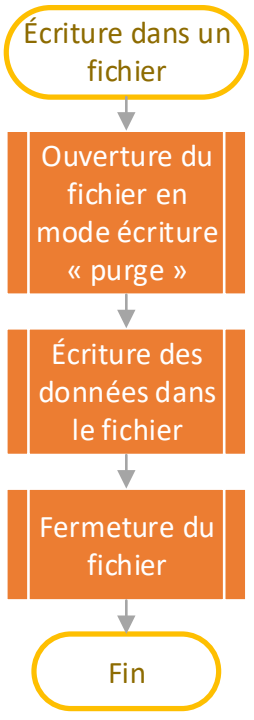
Code exemple :

```
import os #Importation de la librairie OS
#Ouverture du fichier en mode écriture "ajouter"
#file_write est le fichier dans lequel nous allons ecrire
file_write = open("Fichier ou nous voulons ecrire", "a")
#Ecriture du contenu
file_write.write("Ecriture du contenu souhaite")
#Fermeture du fichier ecrit
file_write.close()
```

Écriture purgente :

Dans ce mode le fichier dans lequel nous écrivons est écrasé à l'écriture. Ce mode d'écriture peut être utilisé pour effacer le contenu d'un fichier.

Algorigramme :



Cette fonction permet l'utilisation d'un fichier en mode d'écriture « purge ». A chaque fois que nous écrirons dans le fichier, tout le contenu présent dans celui-ci sera effacé.

Code exemple :

```
import os #Importation de la librairie OS
#Ouverture du fichier en mode écriture "purge"
#file_write est le fichier dans lequel nous allons ecrire
file_write = open("Fichier ou nous voulons ecrire", "w")
#Ecriture du contenu
file_write.write("Ecriture du contenu souhaite")
#Fermeture du fichier ecrit
file_write.close()
```

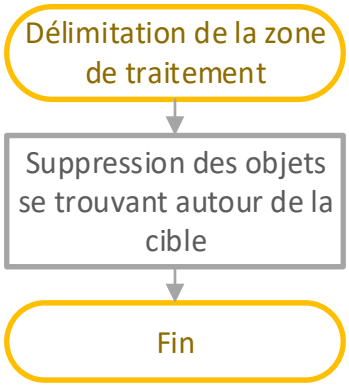
Limitation de la zone de traitement

Les coordonnées des points de la caméra sont données en mètres donc pour avoir une limitation de la zone de traitement optimal, l'utilisateur devra donner plusieurs paramètres pour avoir un traitement de la cible optimale. Cette partie permettra qu'une capture puissent s'effectuer dans des conditions moins bruts

Délimitation de la zone

En premier lieu, nous devons connaitre la distance qui sépare le centre de la plateforme et la caméra, la largeur maximum de la cible ainsi que sa hauteur, et nous devrons aussi connaitre la hauteur de la caméra. Puis à l'aide de condition losr de la lecture du modèle, les parties du modèle se trouvant en dehors de la délimitation ne seront pas pris en compte pour la suite du traitement.

Algorigramme

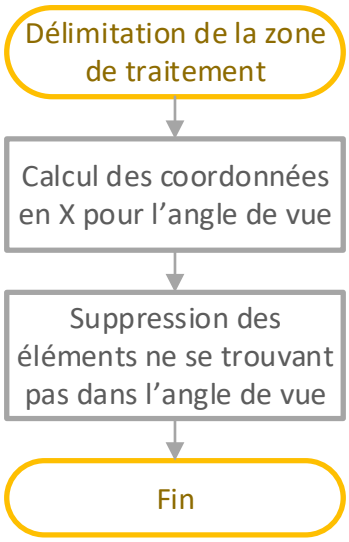


Réduction l'angle de vue

Lors de la prise des images 3D, l'angle de vue de la caméra est de 85.2°. Cet angle ne nous arrange pas pour la création de l'objet 3D complet. Nous devons donc réduire cet angle pour qu'il convienne à la rotation que va effectuer la plateforme. Pour ce faire nous allons utiliser un calcul basique de la trigonométrie qui est le suivant :

$$Nouvel\_angle = distance\_centre * \tan\left(\frac{angle\_rotation}{10}\right)$$

Algorigramme



Rotation

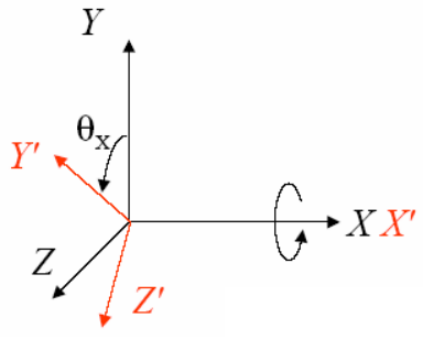
La plateforme effectuant une rotation, nous avons besoins pour la création d'un modèle 3d complet de prendre cette rotation en compte. Pour ce faire à chaque nouveau modèle 3D prit par la caméra, nous devons effectuer un calcul ayant exactement la même valeur de la rotation que la plateforme. Pour ce faire nous allons utiliser des matrices de rotations. Cette matrice sera une matrice 3 * 3.

Rotation sur l'axe X :

Lorsque nous effectuons une rotation sur l'axe X, les coordonnées des points par rapport à l'axe X ne sont pas changer. Par contre les coordonnées des points par rapports aux axes Y et Z, eux se trouvent changés par cette rotation.

Matrice de rotation sur l'axe X :

Rotation à angle positif :



$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta x & -\sin\theta x \\ 0 & \sin\theta x & \cos\theta x \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$x' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = x * 1 + y * 0 + z * 0 = x$$

$$y' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 0 \\ \cos \theta x \\ \sin \theta x \end{pmatrix} = x * 0 + y * \cos \theta x + z * \sin \theta x = y * \cos \theta x + z * \sin \theta x$$

$$z' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 0 \\ -\sin \theta x \\ \cos \theta x \end{pmatrix} = x * 0 + y * -\sin \theta x + z * \cos \theta x = y * -\sin \theta x + z * \cos \theta x$$

Rotation à angle négatif :

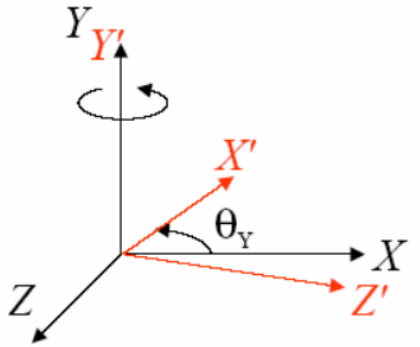
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta x & \sin \theta x \\ 0 & -\sin \theta x & \cos \theta x \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Rotation sur l'axe Y :

Lorsque nous effectuons une rotation sur l'axe Y, les coordonnées des points par rapport à l'axe Y ne sont pas changer. Cependant les coordonnées des points par rapports aux axes X et Z, eux se trouvent changés par cette rotation.

Matrice de rotation sur l'axe Y :

Rotation à angle positif :



$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos \theta y & 0 & \sin \theta y \\ 0 & 1 & 0 \\ -\sin \theta y & 0 & \cos \theta y \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$y' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = x * 0 + y * 1 + z * 0 = y$$

$$x' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} \cos \theta y \\ 0 \\ -\sin \theta y \end{pmatrix} = x * \cos \theta y + y * 0 + z * -\sin \theta y$$

$$= x * \cos \theta y + y * 0 + z * -\sin \theta y$$

$$z' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 0 \\ -\sin \theta x \\ \cos \theta x \end{pmatrix} = x * 0 + y * -\sin \theta x + z * \cos \theta x = y * -\sin \theta x + z * \cos \theta x$$

Rotation à angle négatif :

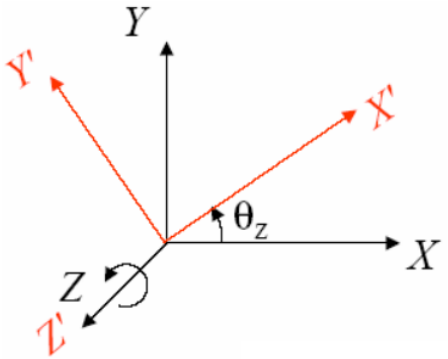
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos \theta y & 0 & -\sin \theta y \\ 0 & 1 & 0 \\ \sin \theta y & 0 & \cos \theta y \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Rotation sur l'axe Z :

Lorsque nous effectuons une rotation sur l'axe Z, les coordonnées des points par rapport à l'axe Z ne sont pas changer. En revanche les coordonnées des points par rapports aux axes X et Y, eux se trouvent changés par cette rotation.

Matrice de rotation sur l'axe Z :

Rotation à angle positif :



$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos \theta x & -\sin \theta x & 0 \\ \sin \theta x & \cos \theta x & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$x' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = x * 1 + y * 0 + z * 0 = x$$

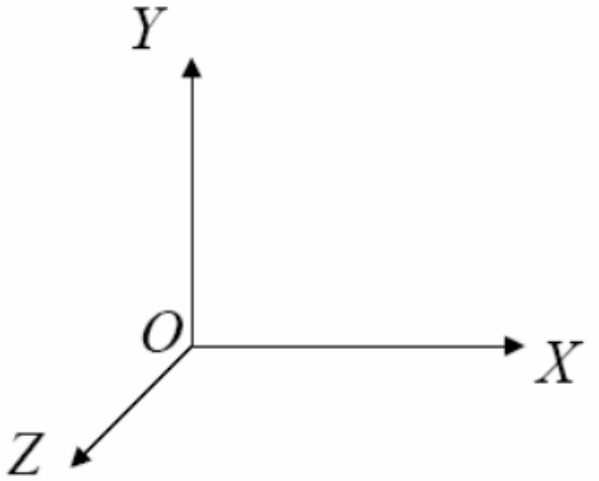
$$y' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} \cos \theta x \\ \sin \theta x \\ 0 \end{pmatrix} = x * \cos \theta x + y * \sin \theta x + z * 0 = x * \cos \theta x + y * \sin \theta x$$

$$z' = \begin{pmatrix} x \\ y \\ z \end{pmatrix} * \begin{pmatrix} 0 \\ -\sin \theta x \\ \cos \theta x \end{pmatrix} = x * 0 + y * -\sin \theta x + z * \cos \theta x = y * -\sin \theta x + z * \cos \theta x$$

Rotation à angle négatif :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos \theta x & -\sin \theta x & 0 \\ \sin \theta x & \cos \theta x & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Représentation du repère de la caméra :

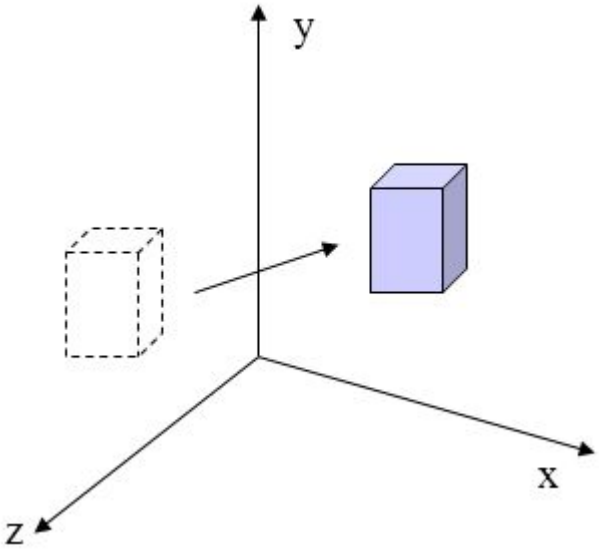


Comme dans la représentation ci-contre, nous pouvons voir que l'axe représentant la hauteur est l'axe Y. Nous ferons donc tourner le modèle sur l'axe Y lors de la rotation.

Translation

Pour la création du modèle complet, nous devons effectuer une translation. Cette translation nous permettra de recentrer l'objet 3D. Nous effectuons une translation par le calcul

Calcul translation :



$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} Px \\ Py \\ Pz \end{pmatrix} + \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$x' = Px + x$$

$$y' = Py + y$$

$$z' = Pzz$$

Matrice Homogène

Dans les calculs matriciels, il existe un moyen d'effectuer une rotation et une translation dans le même calcul, cette matrice se nomme matrice homogène.

$$P' = \begin{pmatrix} R & T \\ 0^t & 1 \end{pmatrix} P$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} R & T \\ 0^t & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T \\ 0 \begin{pmatrix} x \\ y \\ z \end{pmatrix} + 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Le problème de cette matrice est le fait que la translation s'effectue après la rotation ce qui fait que le modèle n'est centré lors de sa rotation. Dans notre cas, nous voulons effectuer le centrage avant la rotation. Dans le cas, où nous utiliserons cette matrice notre résultat serait faussé.

Création d'un fichier PCD

Pour pouvoir effectuer la création du modèle 3D complet, nous avons besoin de recréer un fichier 3D qui contiendra toutes les coordonnées des points sur lesquelles nous avons effectué les rotations et le recentrage. Pour ce faire il faut que nous soyons capables de reproduire le Header d'un fichier PCD.

Header d'un fichier PCD :

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 693395
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 693395
DATA ascii
```

Fields : Disposition des coordonnées

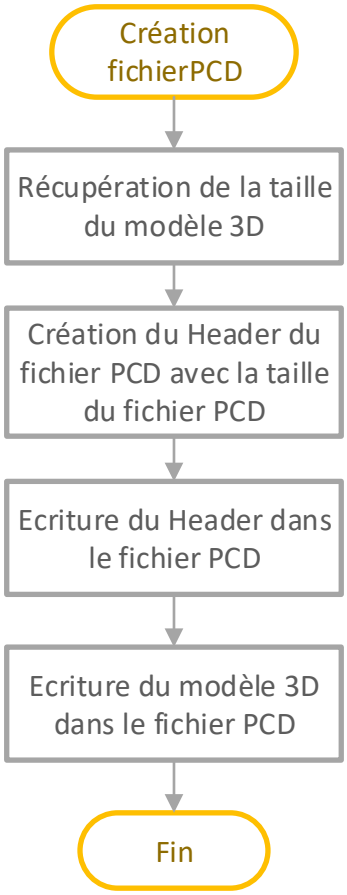
Type : Signifiant que les données sont de types flottants

Points et Width : correspond au nombre de points présent dans le fichier

Data : Signifit que les données sont stockées sous le format ASCII et non binaire

Nous devrons donc recréer ce type d'en tête pour le fichier 3D final. Dans notre cas, nous modifierons un seul paramètre qui se trouve être le nombre de point qui devra correspondre aux nombres de points se trouvant dans le modèle final.

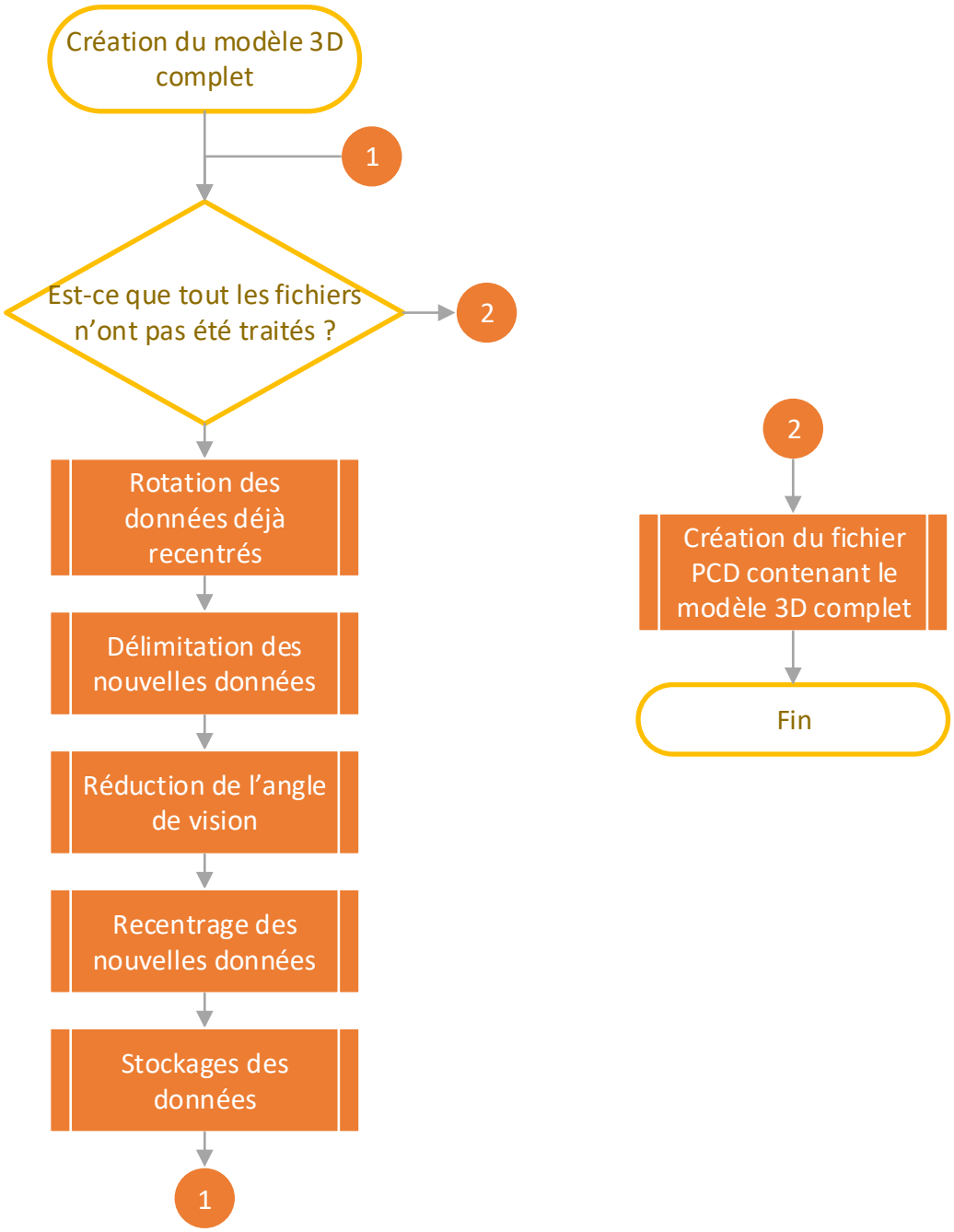
Algorigramme :



Création du modèle 3D complet

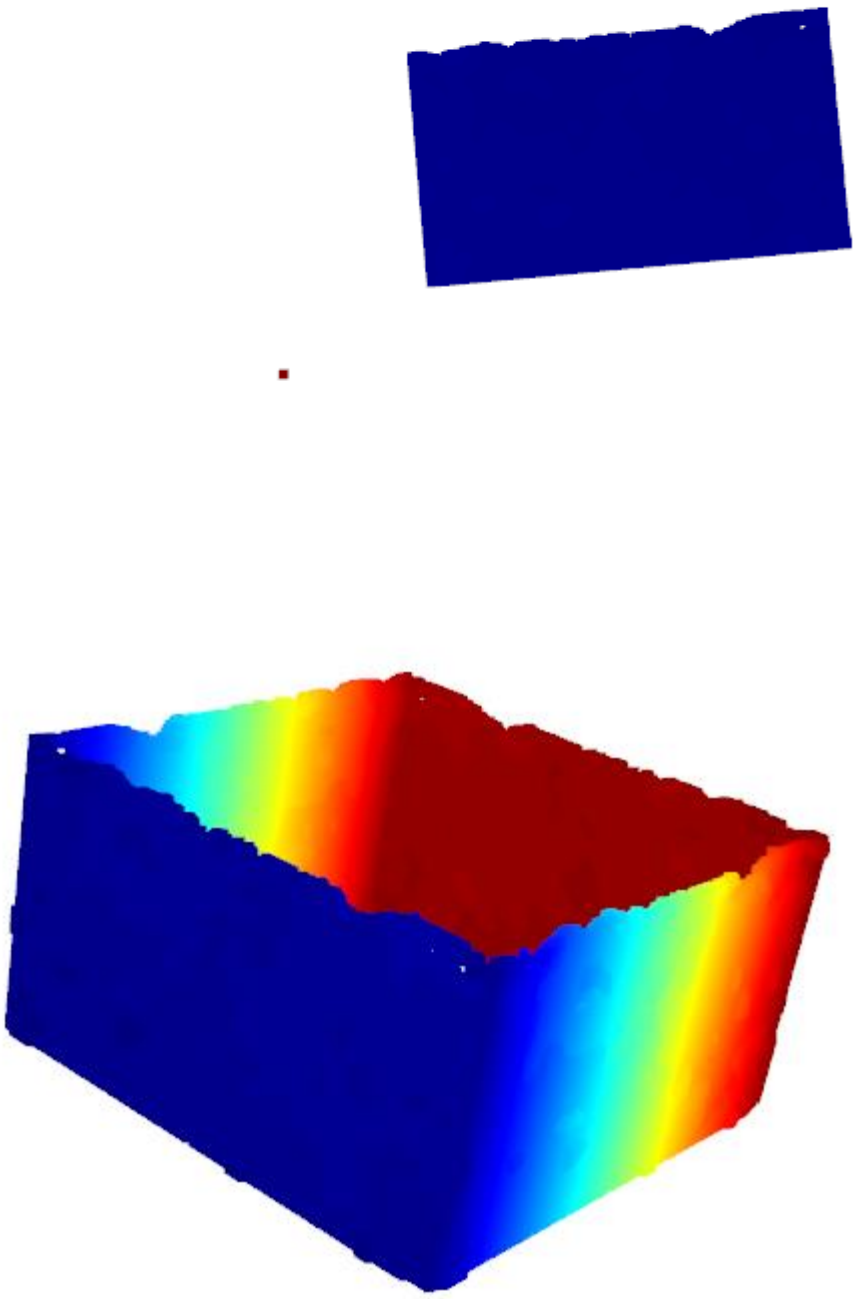
Pour la création du modèle 3D complet, nous allons rassembler les différentes parties que nous venons de vous présenter. En premier lieu nous devrons convertir le fichier capturer par la caméra en format PCD. Ensuite sur ce nouveau fichier, nous délimiterons la zone de capture, après nous ferons une rotation sur l'axe des Y pour les différents que nous avons déjà recentré, une fois que cette rotation a été effectué nous recentrons les nouveaux points que nous devons rajouter au modèle. Une fois que le modèle a été compléter, nous stockerons tous ces points dans un fichier PCD.

Algorigramme :



Test

Pour tester cette partie, nous avons capturé la face d'une boîte pour recréer un carré avec cette face.



Plateforme

Pour la plateforme, au tout début du projet nous devions être capable de scanner un être humain. Donc nous devions avoir une plateforme capable de tourner, ainsi que de supporter le poids d'un Homme. Cet objectif a été revu à la baisse, à cause de différents soucis que nous avons encourus lors du déroulement du projet (Départ de deux étudiants du projet). Le nouvel objectif a été déterminé avec Mr Aubry, maintenant nous être capable de scanner une figurine et non plus un être humain. Ce changement d'objectif a permis de retirer une bonne partie du travail que nous devions faire sur la plateforme, comme pour le choix du matériel de la plateforme comme elle n'a plus à supporter un humain, et le choix du moteur pour les mêmes raisons.

Présentation de la plateforme :

Pour la conception de la plateforme n'ayant pas le temps, de la concevoir par mes propres moyens via l'aide d'un outil comme SolidWorks, je me suis dirigé vers des projets de scanner 3D Open Source, pour récupérer une plateforme rotative.

Ciclop :

Je me suis donc dirigé vers le projet ciclop, que nous avons vu pendant l'état de l'art. Cette plateforme étant Open Source, et les personnes l'ayant faite fournissant toutes les pièces en format 3D, pouvant être modifié et imprimé.

A la base cette plateforme est faite pour les scans 3D par Laser. Dans notre cas, nous allons utiliser une caméra 3D donc la partie support de celle-ci doit être modifié en conséquence. Cette partie est actuellement en cours de conception par Romain Lux--Quach

Présentation des différents composants :

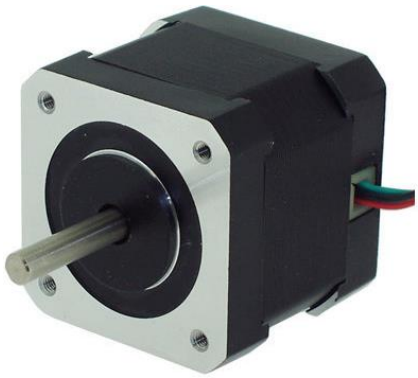
Pour assurer le bon fonctionnement de la plateforme, nous avons besoin de différents composants.

Moteur Pas à Pas :

Nous avons fait le choix de prendre un moteur pas à pas, pour plusieurs raisons. La première était la compatibilité avec la plateforme qui demandait une taille d'arbre pour le moteur précise et ainsi ne pas à avoir à modifier la pièce 3D. La seconde était pour l'angle de rotation car un moteur pas à pas à un fonctionnement très précis, qu'il fait une rotation par étape dans à chaque étape, son angle change d'une valeur précise.

Moteur pas à pas choisi :

Nous avons choisi le moteur 17HS15-0404S, ce moteur à un couple de 4Kg.cm, et une taille de l'axe Nema 17 (5 * 22 mm). Il possède un nombre de pas de 200, soit 1,8° par pas. Il s'alimente à 12V et avec un courant de 0.4A. Il fait une dimension de 42 x 42 x 42 mm et un poids de 240g.

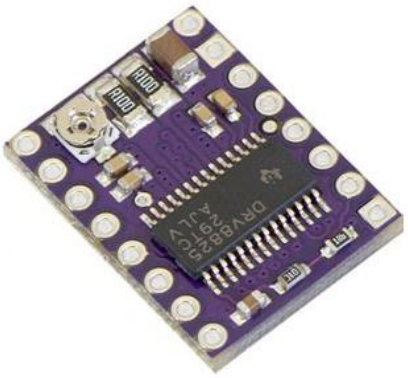


Carte de commande de moteur pas à pas :

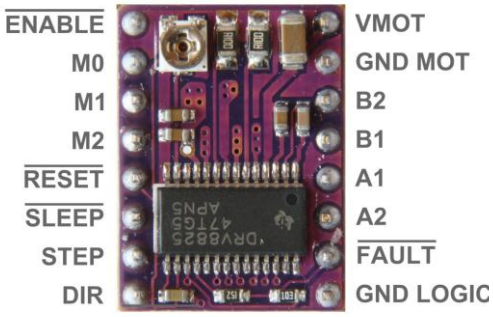
Une carte de commande de moteur pas à pas permet de contrôler un moteur pas à pas à l'aide d'un signal PWM.

Commande moteur choisi :

Nous avons choisi la carte de commande moteur DRV8825 2133. Cette carte accepte une alimentation pour la partie logique de 2.5 à 5.25V et pour la partie moteur de 8.2 à 45V avec un courant de sortie de 1.5A par phase. Cette carte permet de choisir le sens de rotation du moteur, la division du pas en sortie, de régler l'intensité de sortie via un potentiomètre et de d'activer ou désactivé la rotation du moteur via une broche. Elle est faite pour les moteur pas à pas bipolaire. Elle nous permet de ramener à la bonne tension et de le traiter pour la bipolarité notre signal PWM.



Fonctionnement de la carte de commande moteur



- La broche ENABLE permet d'activer ou désactiver la rotation du moteur, elle fonctionne en logique inverse donc quand elle est à 0, le moteur est activé
- M0, M1, M2 sont les broches qui permettent le micro-Stepping (division du pas)

M0	M1	M2	Résolution Micro-Stepping
Low	Low	Low	Full step (pas complet)
High	Low	Low	Demi pas
Low	High	Low	1/4 de pas
High	High	Low	1/8 de pas
Low	Low	High	1/16 de pas
High	Low	High	1/32 de pas
Low	High	High	1/32 de pas
High	High	High	1/32 de pas

- RESET et SLEEP doivent être branché ensemble
- La broche STEP est la broche où nous enverrons le signal PWM pour effectuer la rotation du moteur
- La broche DIR permet de sélectionner le sens dans lequel tournera le moteur
- VMOT est la broche où nous mettrons l'alimentation de notre moteur
- B2, B1, A2, A1 correspondent aux broches où nous devrons connecter le moteur
- Fault sera branché à la tension logique de la carte microcontrôleur

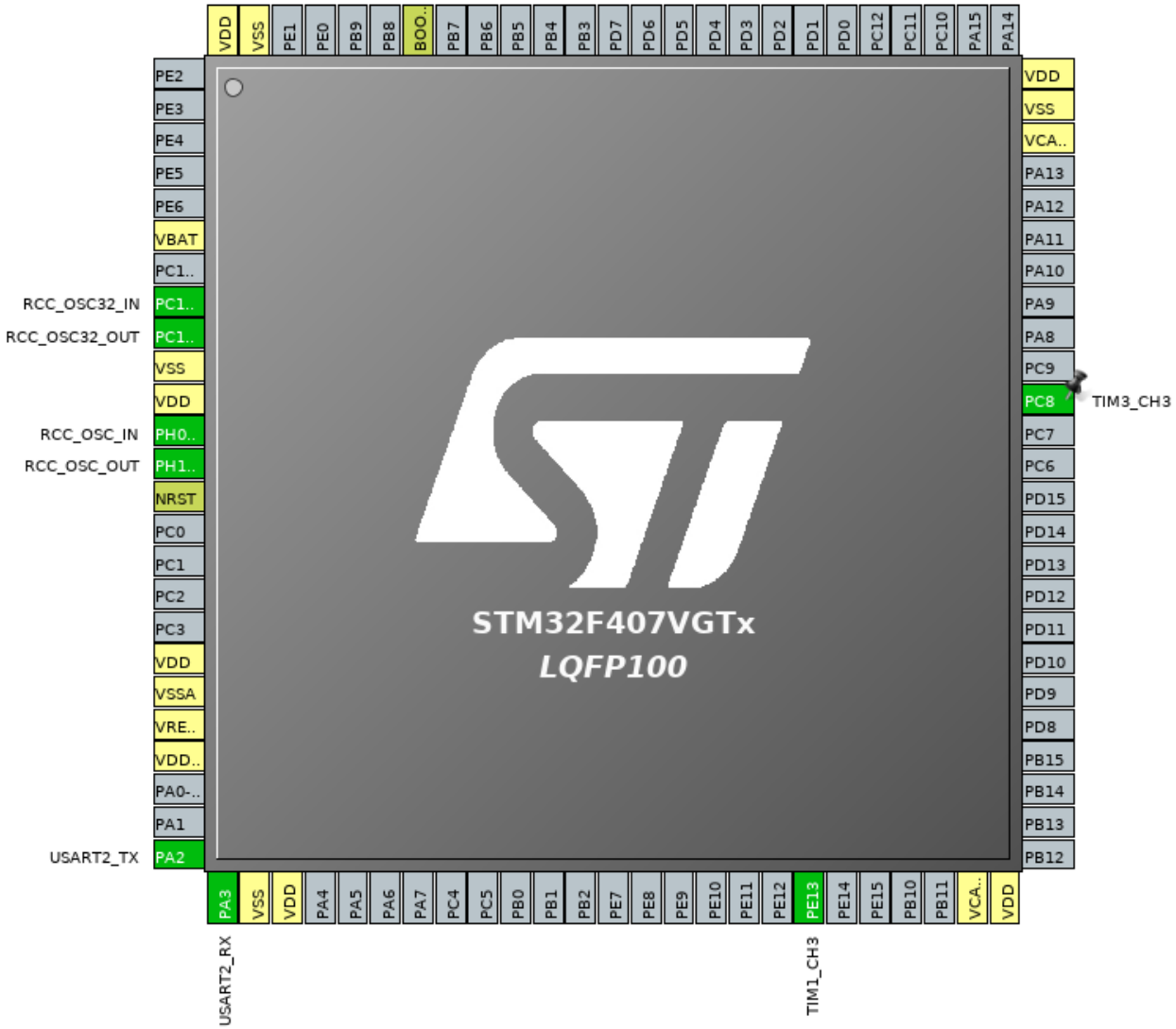
Microcontrôleur :

Nous avons besoin d'une carte microcontrôleur pour assurer le bon fonctionnement de la plateforme et ainsi assurer le bon fonctionnement du scanner 3D.

Microcontrôleur choisi :

Nous avons choisi la carte STM32-Discovery comme carte microcontrôleur, ce choix a été fait par simplicité car c'est une carte que nous avions déjà étudié au préalable. Elle dispose d'un logiciel qui est STM32CubeMx qui permet de la configurer plus facilement. Elle permet via une carte d'extension la communication série via USB et des branchement Mikrobus de MikroElektronika pour brancher la carte de commande moteur.

Pin de la STM32 :



Roulement à bille :

Pour assurer une bonne rotation de la plateforme, un roulement a été ajouté pour effectuer ceci. Nous avons pris comme roulement à bille, un roulement faisant une taille de 70*110*13mm. Il sera positionné sur une pièce de la plateforme qui est prévu pour.

Fonctionnement de la plateforme :

La plateforme est présente pour effectuer une rotation de la pièce que nous pouvons contrôler. Pour effectuer nous ceci nous utiliserons une PWM, ainsi qu'une InputCapture et d'autre GPIO pour assurer le bon fonctionnent de celle-ci. Le signal PWM sera envoyé à la carte moteur, pour assurer la rotation du moteur, une autre broche permettra d'activer ou de

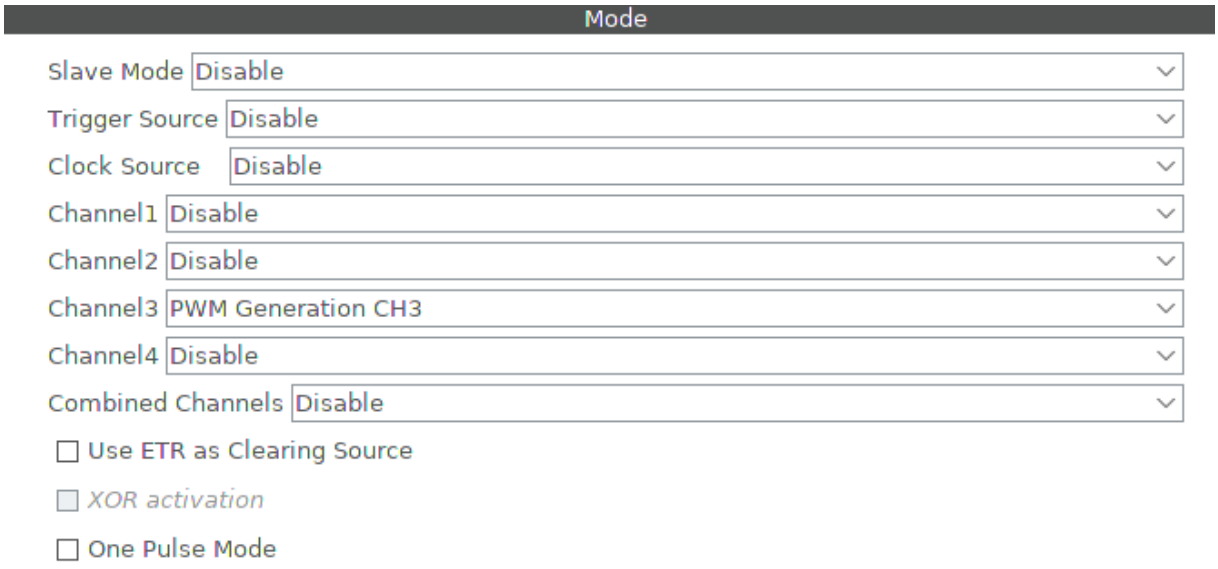
désactiver la rotation et une autre le sens. Nous utiliserons l'InputCapture pour compter le nombre de fois où le signal PWM a été émis et ainsi savoir le nombre de pas que nous avons effectué. L'inputCapture sera réglait en interruption pour effectuer cette tâche.

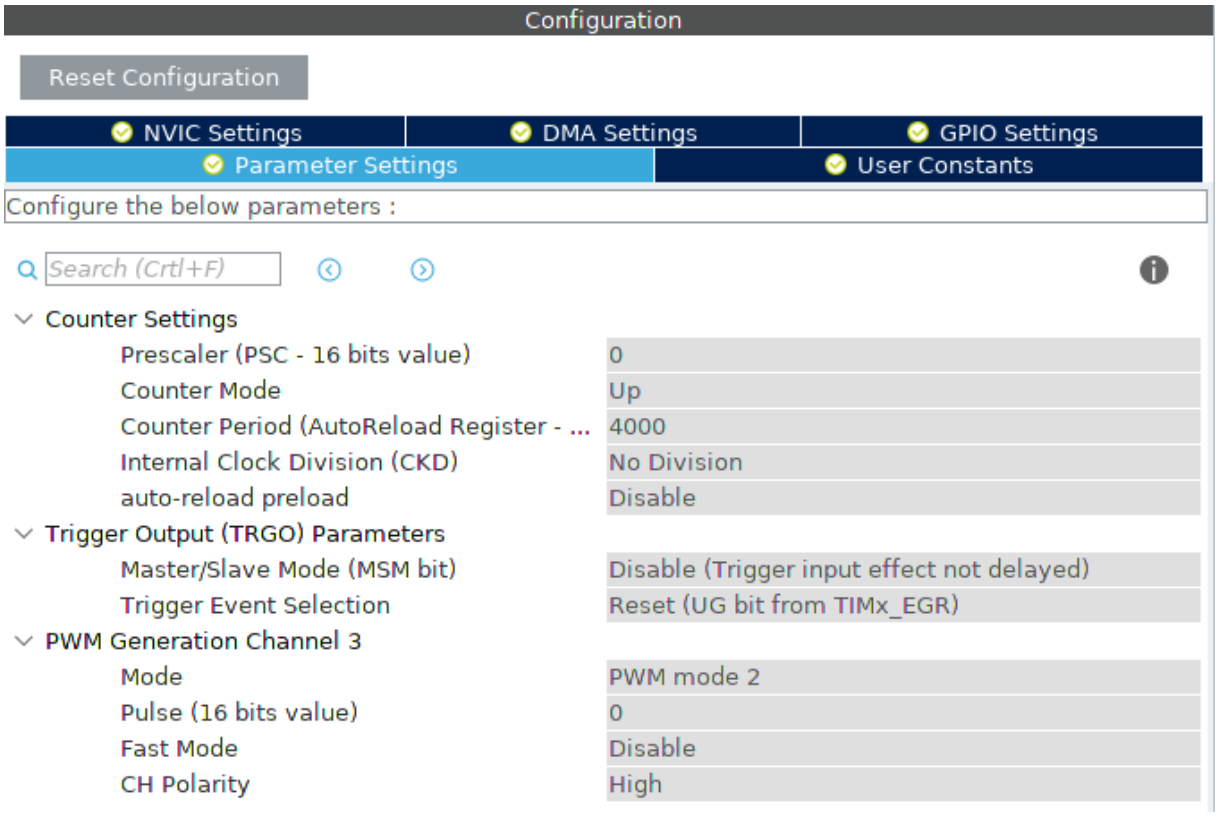
Configuration :

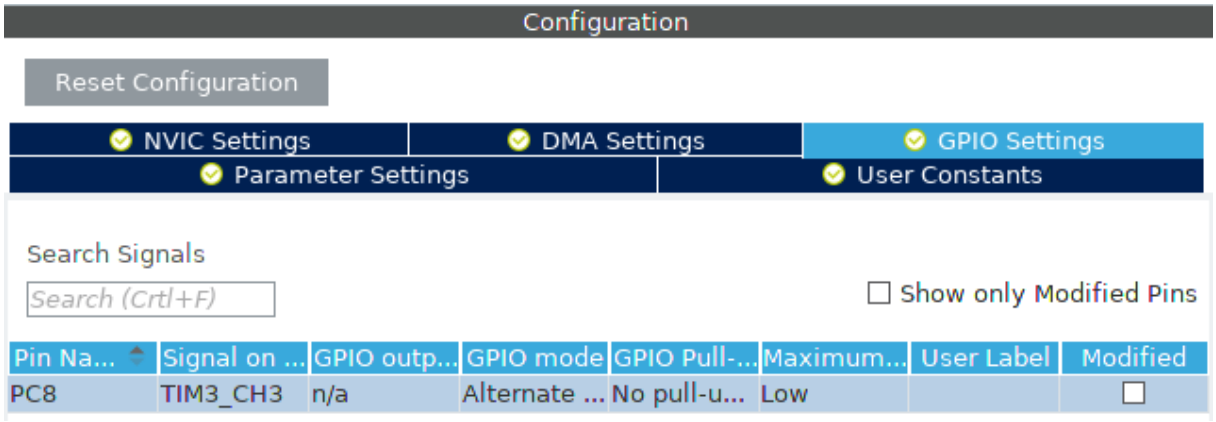
Pour la configuration de la carte STM32, celle-ci a été faite via STM32CubeMX. Il s'agit d'un logiciel permettant la configuration des cartes STM32. Nous pouvons y configurer ce qui est matériel comme UART, les Timers, la gestion des pins, etc. Nous pouvons aussi y configurer des parties logicielles comme freeRTOS.

PWM :

Pour la PWM, nous travaillerons sur le Timer3 et plus précisément sur la GPIO PC8.







Input Capture :

Pour l'InputCapture, nous travaillerons sur le Timer1 et plus précisément sur la GPIO PE13.

TIM1 Mode and Configuration

Mode

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Channel1	Disable
Channel2	Disable
Channel3	Input Capture direct mode
Channel4	Disable
Combined Channels	Disable

- [ ] *Activate-Break-Input*
- [ ] *Use ETR as Clearing Source*
- [ ] *XOR activation*
- [ ] One Pulse Mode

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings

Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+F)

Counter Settings	
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - ...	4000
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Input Capture Channel 3	
Polarity Selection	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter (4 bits value)	0

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings

Parameter Settings | User Constants

NVIC Interrupt Table	Enabl...	Preemption Prior...	Sub Prior...
TIM1 break interrupt and TIM9 global interrupt	☐	0	0
TIM1 update interrupt and TIM10 global interrupt	☐	0	0
TIM1 trigger and commutation interrupts and TIM11 global inte...	☐	0	0
TIM1 capture compare interrupt	☑	0	0

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings

Parameter Settings | User Constants

Search Signals

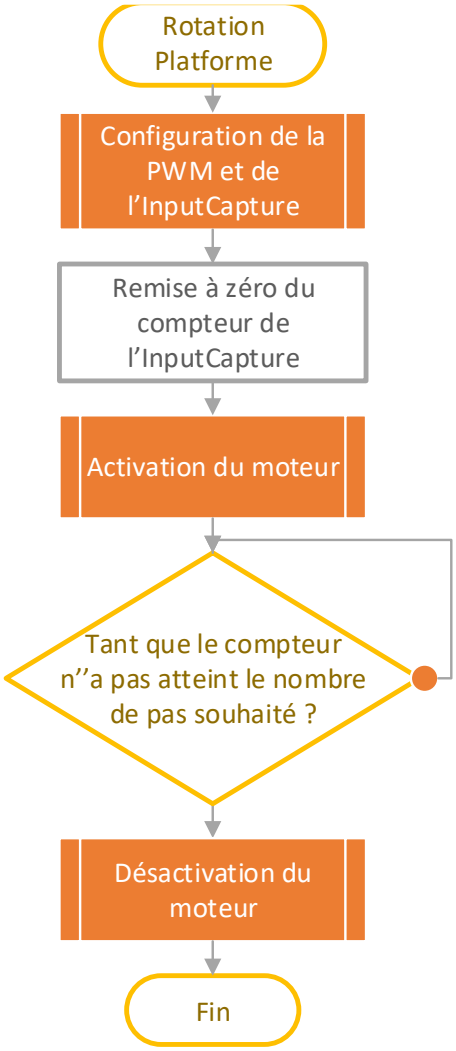
Search (Crtl+F)

☐ Show only Modified Pins

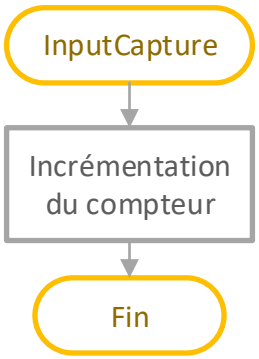
Pin Na...	Signal on ...	GPIO outp...	GPIO mode	GPIO Pull-...	Maximum ...	User Label	Modified
PE13	TIM1_CH3	n/a	Alternate ...	No pull-up...	Low		☑

Algorigramme :

Programme Principal :



InputCapture en interruption :



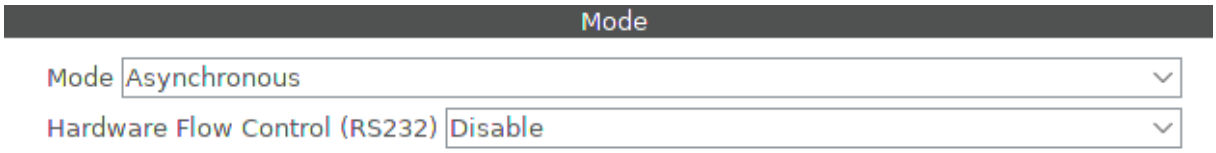
Communication série

Pour la communication entre la plateforme et le scanner 3D, nous passerons par une communication série. Cette communication sert à ce que le scanner demande une rotation à la plateforme est uniquement à cela pour le moment.

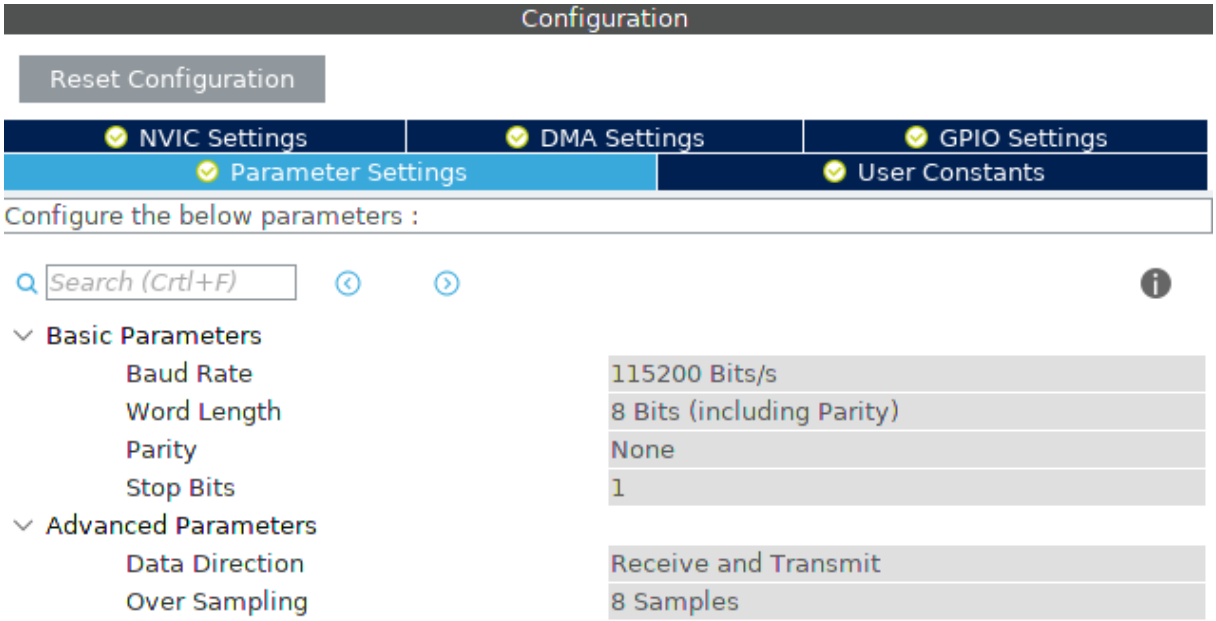
Plateforme

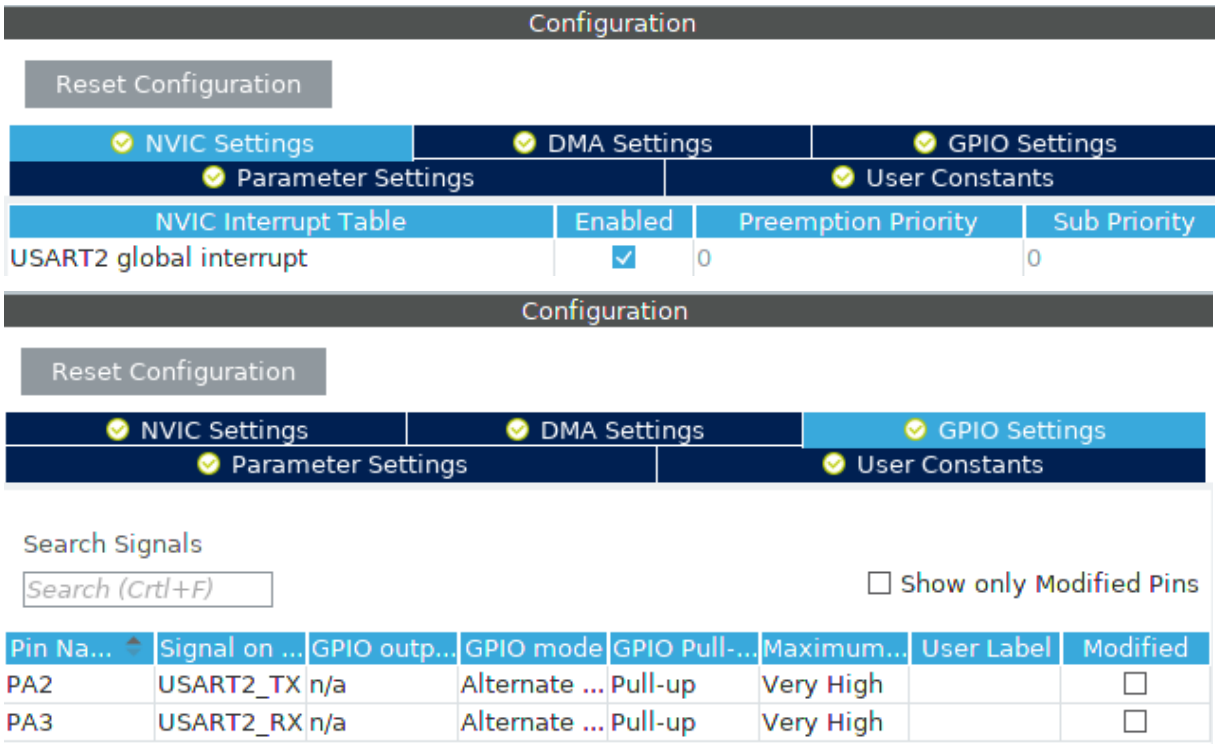
Configuration de la carte

Configuration du mode de l'UART :



Configuration de l'UART :





Algorigramme

Concernant la réception de la donnée, nous la ferons en interruption. Et pour l'envoie de la donnée nous la ferons directement dans le programme via le programme.

Ordinateur

Librairie utilisée

Pour la communication série, nous avons utilisé la librairie Python pyserial. Cette librairie s'installe via la ligne suivante :

```
pip install pyserial
```

Utilisation :

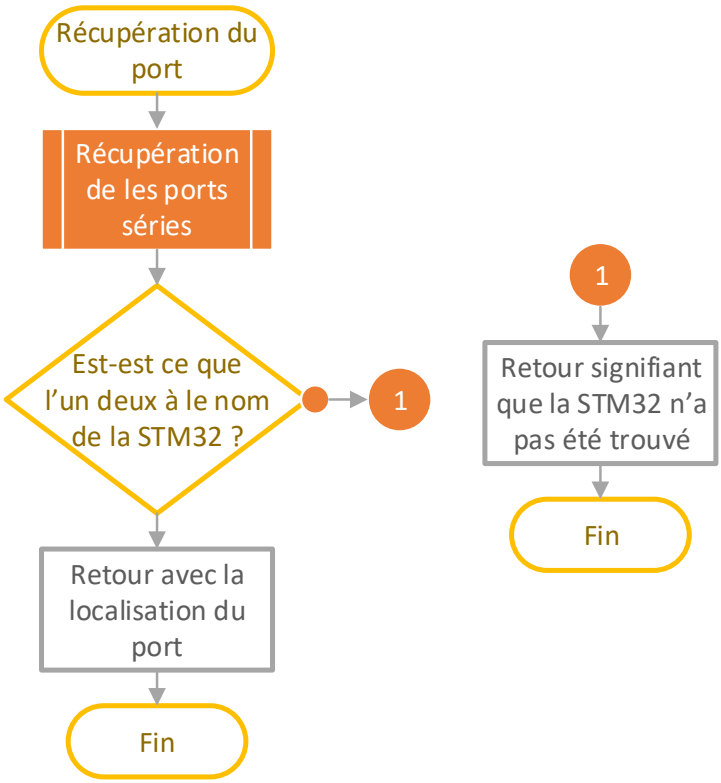
Sous linux, pour pouvoir avoir accès à un port série il faut que soit nous donnions les droits en lançant le programme Python avec la commande sudo ou que nous octroyions les droits via différentes commandes qui sont les suivantes :

```
cd /dev
chmod 333 ttyUSB0
```

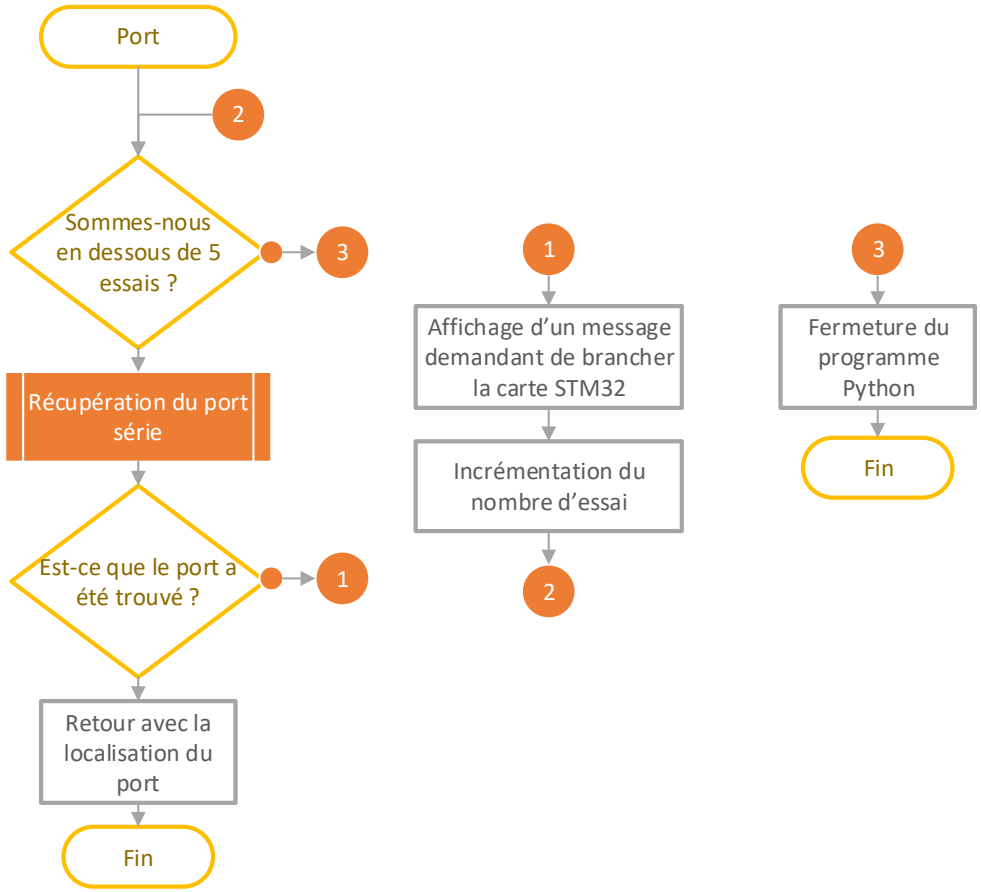
Algorigramme

Fonction de récupération du port série

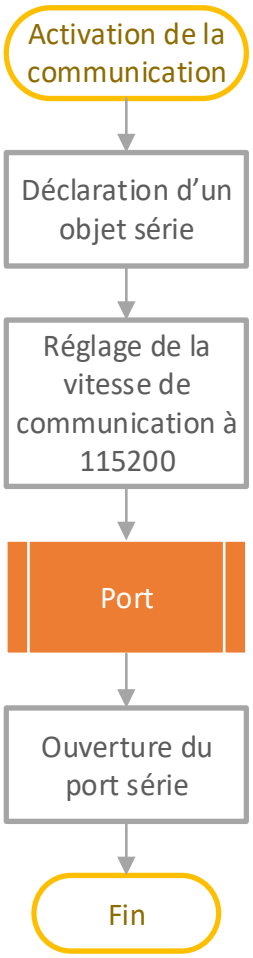
Une fonction pour la récupération du port série a été faite. Cette fonction permet que l'utilisateur n'ai pas à sélectionner le port série sur lequel se trouve la carte STM32.



Cette fonction permet de récupérer la localisation du port série de la STM32.


Port
2
Sommes-nous en dessous de 5 essais ?
3
Récupération du port série
Est-ce que le port a été trouvé ?
1
Retour avec la localisation du port
Fin
1
Affichage d'un message demandant de brancher la carte STM32
Incrémentation du nombre d'essai
2
3
Fermeture du programme Python
Fin

Activation de la communication série :




```
ser = serial.Serial()
ser.baudrate = 115200
ser.port = ports()
ser.open()
```

Écriture et Lecture :

Ecriture

```
ser.write([data])
```

Lecture

```
out = ser.read()
```

Logiciel

Pour le moment, l'interface Homme Machine du Scanner 3D est un terminal, il est prévu de la faire soit avec ImGUI ou QT par le futur mais par manque de temps ceci n'a pas pu être fait.

Terminal

L'interface sur le terminal permet de contrôler, le système. Il s'agit d'une interface Homme Machine de test, qui n'est pas définitive mais lors du projet, nous avons préféré travailler sur d'autres tâches que la partie IHM car celle-ci permet juste de faire la connexion entre l'être humain et le programme, donc nous nous sommes plus concentrés sur la partie fonctionnement du scanner 3D.

Cette interface marche sous la forme d'un menu qui représente les différentes étapes de la conception d'un modèle 3D. En premier lieu nous pourrons effectuer une visualisation du point de vue de la caméra. En second il y aura la délimitation de la zone de traitement, pour cela l'utilisateur devra rentrer les différentes informations que nous avons vu comme la distance entre la caméra et la plateforme, la largeur de la cible, etc. Nous trouverons après, la partie de capture et donc de scan de la cible, l'utilisateur pourra donner un nom au fichier qu'il souhaite avoir en sortie. Ensuite nous aurons le lancement de la partie création du modèle 3D. Après cela, l'utilisateur pourra choisir d'afficher le modèle qui vient d'être créé. La prochaine étape permet que l'utilisateur choisissent le format de sortie qu'il souhaite. Et enfin la sortie du programme. Par exemple si à la visualisation de la pièce 3D, celle-ci se trouve être fausse l'utilisateur peut changer la zone de délimitation et relancer la conception de sa pièce. Ou s'il souhaite avoir modèle en plusieurs format en sortie c'est possible.

Nous savons qu'il ne s'agit pas de la meilleure interface pour l'expérience utilisateur mais nous avons dû faire avec différents problèmes au niveau de celle-ci.

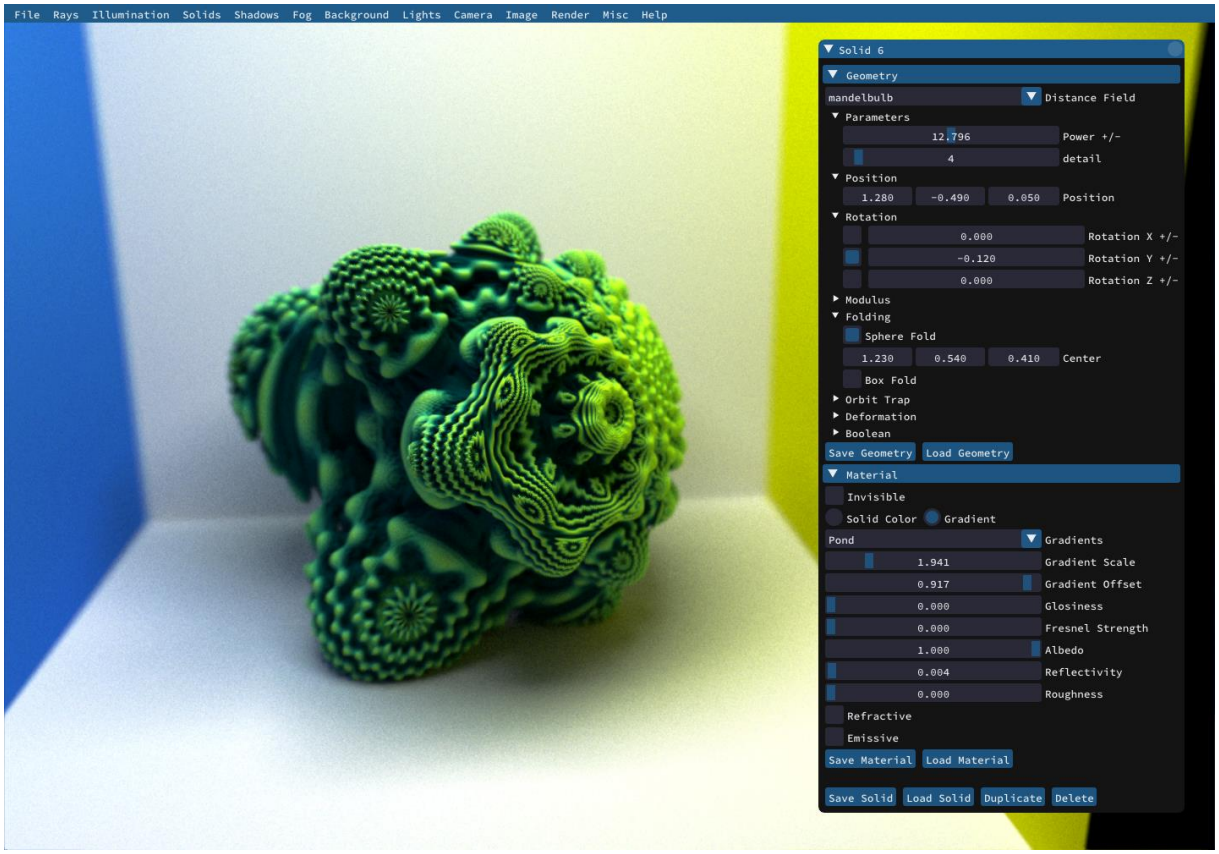
IMAGE

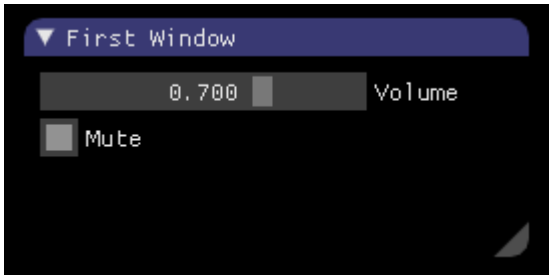
ImGUI

ImGUI est librairie permettant la conception d'interface graphique. Elle disponible pour les langages Python et C++. Avec celle-ci nous pouvons utiliser différentes API 3D comme OpenGL, Vulkan et Direct3D. Cette librairie est utilisée la plupart pour le dessin de widget tout comme Qt.

Le plus gros avantage que nous avons trouvé à cette librairie est le fait que nous n'avons pas à gérer la partie connexion de Qt. Car pour tester si un objet que nous avons

créé a été utiliser nous avons juste à regarder l'état de celui-ci. Ceci permet d'avoir une programmation simplifiée de l'interface.



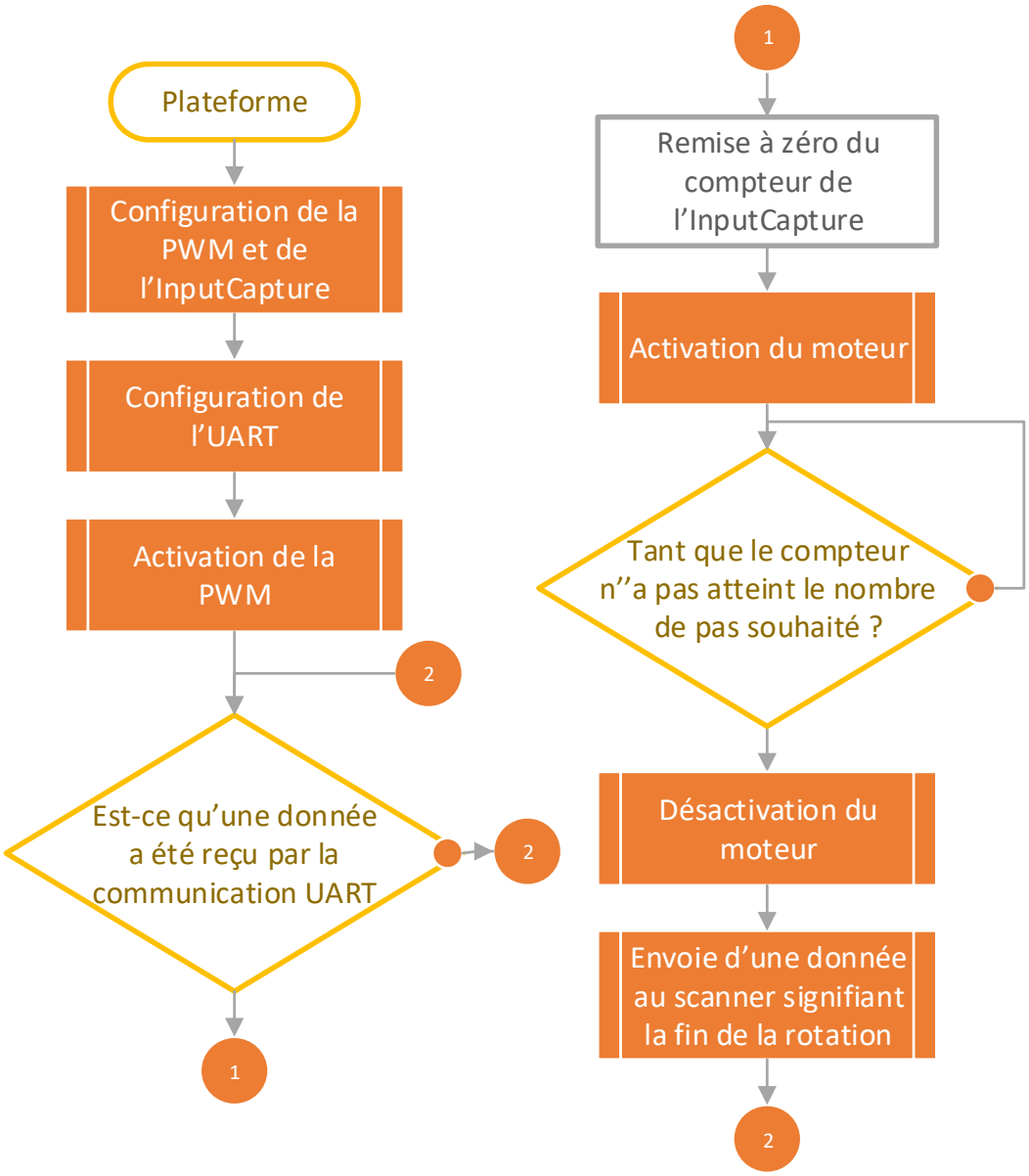


Qt

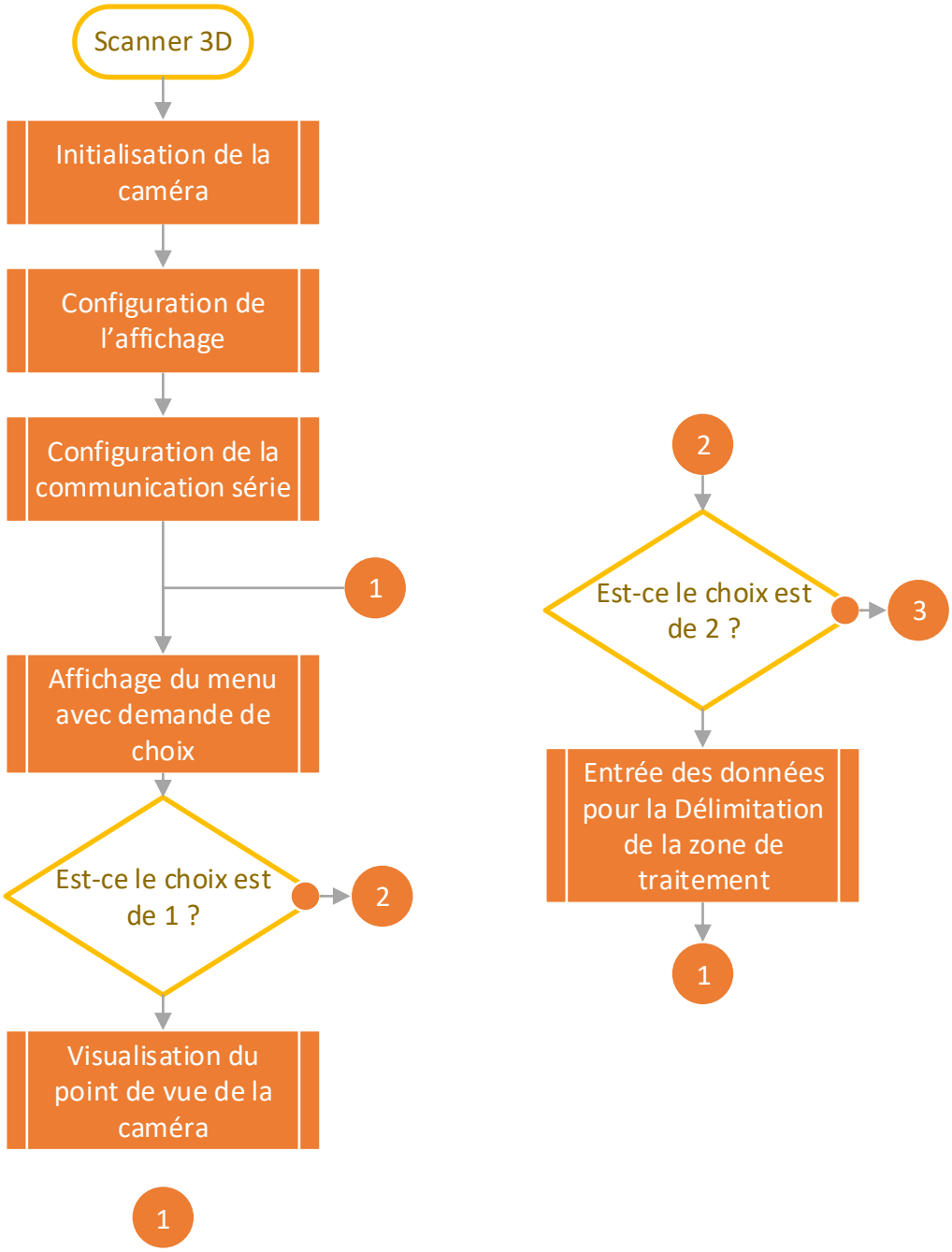
QT est un framework de C++ qui permet de créer des applications fenêtrées appelées GUI. QT est fourni avec un environnement de développement intégré intitulé "QT Creator", avec lequel vous pourrez écrire vos codes sources directement dessus.

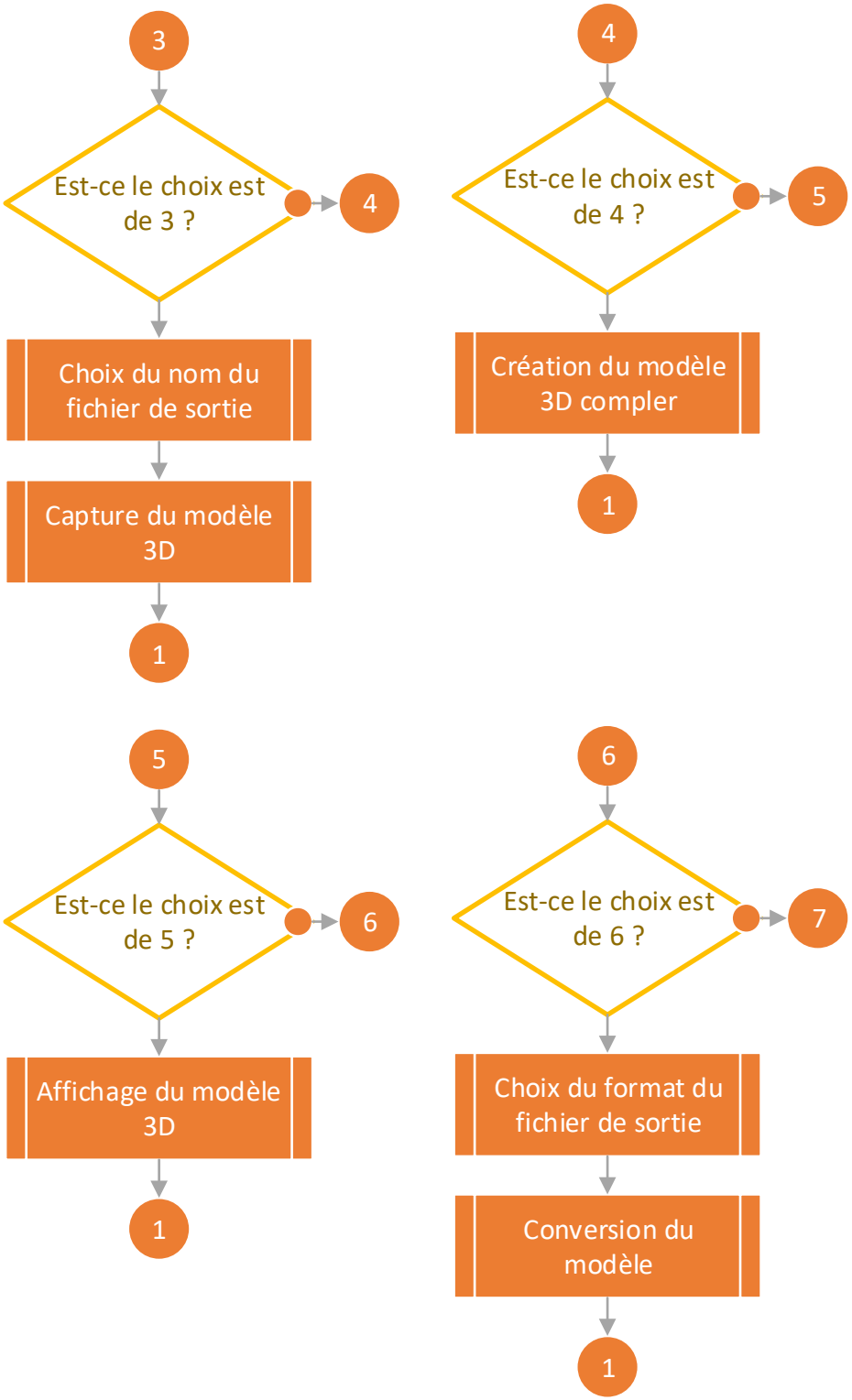
Rendu complet

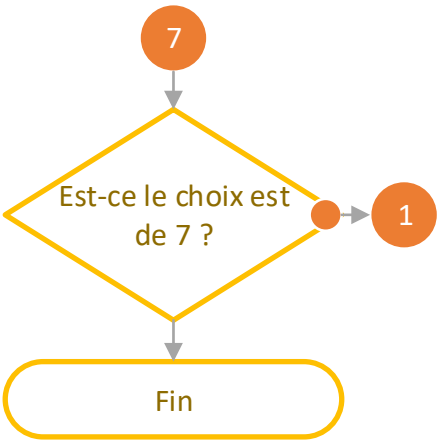
Algorigramme Carte



Algorigramme PC

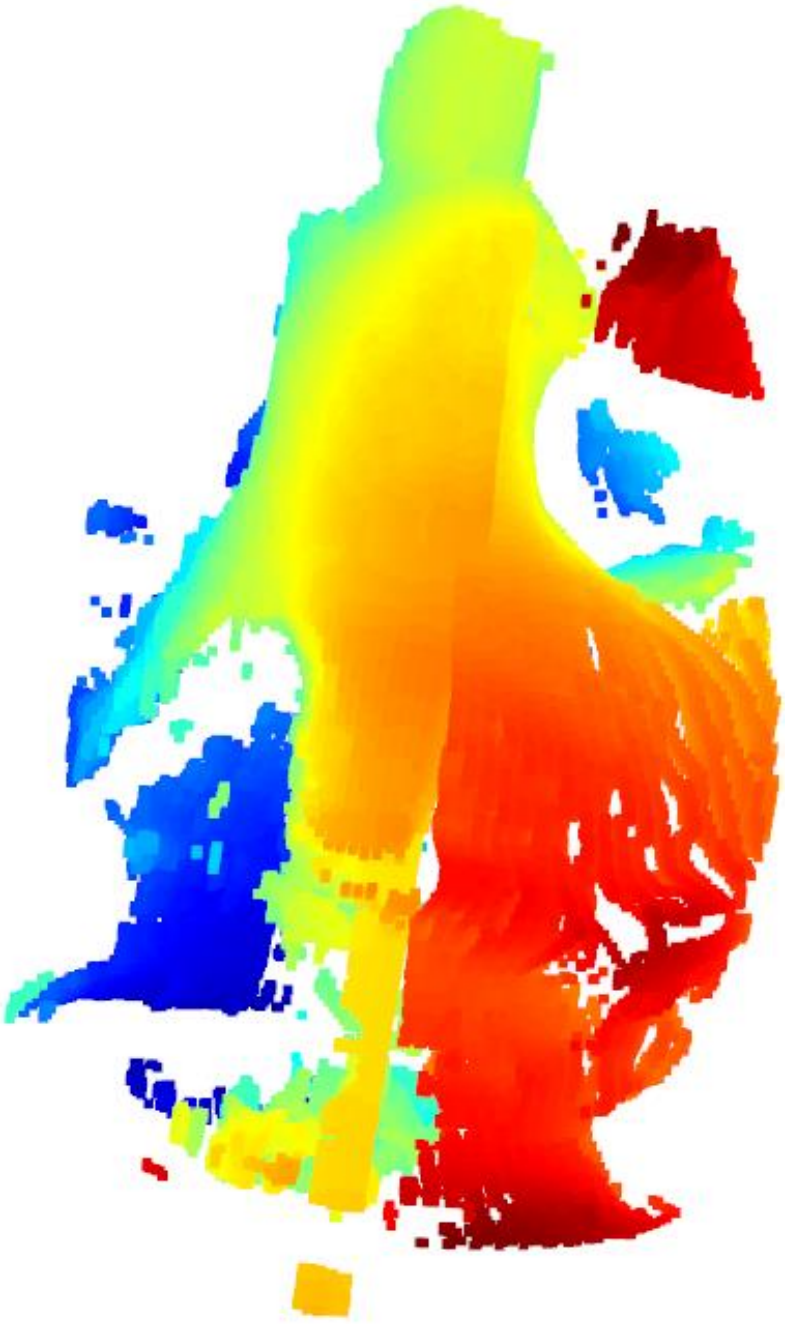


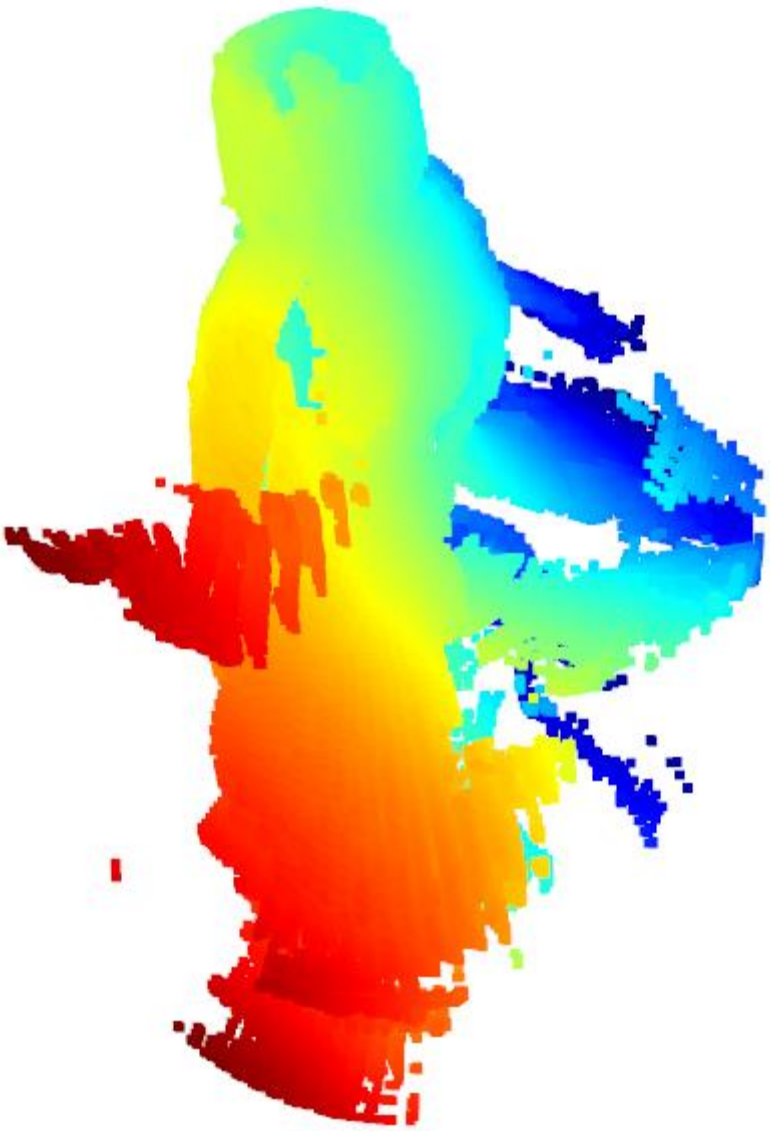

3
Est-ce le choix est de 3 ?
4
Choix du nom du fichier de sortie
Capture du modèle 3D
1
4
Est-ce le choix est de 4 ?
5
Création du modèle 3D compler
1
5
Est-ce le choix est de 5 ?
6
Affichage du modèle 3D
1
6
Est-ce le choix est de 6 ?
7
Choix du format du fichier de sortie
Conversion du modèle
1



Test

Nous avons effectué une conception du modèle sur ce type de figurine. La conception du modèle 3D a duré 3 minutes et 48secondes. Nous avons eu plusieurs problèmes lors de la conception, nous nous sommes trompés dans la distance entre le centre de la plateforme et la caméra, ce qui a faussé le résultat. L'autre problème est le fait que la figurine soit trop complexe pour être scanner sans défauts.





Conclusion

Ce projet nous a appris à travailler la 3D que ce soit l'affichage ou le traitement. Surtout ce projet m'a appris à gérer mon temps pour effectuer les différentes tâches que j'avais à faire. Mais le projet n'est pas tout à fait abouti, nous pouvons améliorer la partie logicielle que ce soit le traitement ou la partie interface. Le projet se trouve toujours sous test donc il n'y a pas d'application exécutable à part le fichier Python et le dossier pour le programme de la STM32.

Amélioration possible pour le projet

Nous pouvons faire plusieurs améliorations à ce projet.

Changer la méthode de calcul de la rotation et de la translation pour la création de l'objet 3D complet. Pour avoir de meilleure performance pour cette partie, nous pouvons passer soit par le langage Cuda ou par Vulkan ce qui permettra d'effectuer les différents calculs 3D directement sur la carte graphique.

Nous pouvons aussi passer l'entièreté de la partie Scanner 3D sur une carte embarqué de type i.mx8, nous pouvons faire ceci car le projet RealSense de base prévue par Intel est pour le monde de la robotique. Intel fournit donc une meta pour yocto pour la caméra que nous pouvons retrouver à l'adresse suivante : https://github.com/IntelRealSense/meta-intel-realsense. Grâce à cela la recette Yocto pour la caméra est déjà faites et nous n'avons plus qu'à adapter le code que nous avons fait pour une carte embarquée.

Maintenant que les plus grosses parties sont faites, nous pouvons continuer le projet pour que celui-ci fasse ce qui était prévu de base à savoir le scan d'un être humain. Pour ce faire nous avons besoin de retravaillé la plateforme pour que celle-ci puissent supporter le poids d'un être humain et aussi que la plateforme a assez de force de rotation pour celui-ci. Nous pouvons aussi passer à un autre modèle de caméra comme l'Intel RealSense d435i qui contient un IMU pour que le scan puisse s'effectuer dans plateforme. Donc le scanner ne sera plus sous la forme d'une manette de Wii avec la caméra, un petit écran pour l'IHM et des boutons pour contrôler le tout.

Gantt réel

Projet Scanner 3D

Nom des tâches	Exécutant	Status	Date de début	Date de fin
Recherche de solution pour le scanner	AxelJACQUOT	Done	2018-12-01	2018-12-30
Recherche de solution pour la plateforme	AxelJACQUOT	Done	2018-12-01	2018-12-30
Programmation C++ OpenGL	AxelJACQUOT	Done	2019-01-07	2019-01-28
Test caméra via langage C++	AxelJACQUOT	Done	2019-01-28	2019-02-18
Programmation réception données de la caméra	AxelJACQUOT	Done	2019-02-18	2019-03-25
Programmation de l'IHM	AxelJACQUOT	Done	2019-03-11	2019-05-29
Choix de la plateforme	AxelJACQUOT	Done	2019-03-25	2019-04-01
Programmation des traitements 3D	AxelJACQUOT	Done	2019-03-25	2019-05-20
Recherche pour la conversion des formats 3D	AxelJACQUOT	Done	2019-03-25	2019-04-29
Programmation de la plateforme	AxelJACQUOT	Done	2019-04-15	2019-04-29
Intégrations des différentes parties	AxelJACQUOT	Done	2019-04-29	2019-05-29
Amélioration du programme	AxelJACQUOT	Working on it	2019-05-20	2019-06-07

Remerciements

Nous remercions énormément, Clément Calliau, pour son aide apportée au projet.

Annexe

Github du projet avec programme de test :

https://github.com/AxelJacquot/Project_Scanner_3D/tree/Affichage_3D

Github du projet :

https://github.com/AxelJacquot/Project_Scanner_3D/tree/Affichage_3D

Le projet logiciel se nomme scanner.py. Le projet plateforme est le dossier plateforme pour le faire fonctionner, il faut le lancer via un logiciel permettant de programmer les cartes STM32 comme System WorkBench.

Tuturiel OpenGL :

https://opengl.developpez.com/tutoriels/apprendre-opengl/