

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication
(TIC)

RÉALITÉ AUGMENTÉE MOBILE DANS UN BAS À SABLE

Réalisé par

Fabien Mottier

Sous la direction de
Prof. Paul Albuquerque
À l'école HEPIA

Dr. Jean-Luc Falcone, département d'informatique, Université de Genève

Lausanne, HES-SO//Master, 2020

Accepté par la HES-SO//Master (Suisse, Lausanne) sur proposition de

Prof. Paul Albuquerque, conseiller du projet d'approfondissement

Dr. Jean-Luc Falcone, département d'informatique, Université de Genève

Lausanne, le 10 juillet 2020

Prof. Paul Albuquerque
Conseiller

Prof.
Responsable de la filière David Grünenwalrd

Table des Matières

1.1.	But de l'étude	1
1.2.	Portée et limites de l'étude	1
2.1.	Projet actuel	3
2.2.	Technologies	3
2.3.	Logiciel actuel	3
2.3.1.	Architecture	4
2.3.2.	Fonctionnement	5
2.3.3.	Calibration	5
3.1.	Nouvelle architecture	9
3.2.	Adaptation des processus de calibration	10
3.2.1.	Génération de la matrice de rotation	10
3.2.2.	Calcul de la position du projecteur	11
4.1.	Application Calibration	13
4.1.1.	Génération de la matrice de rotation	14
4.1.2.	Calcul de la position du projecteur	14
4.1.3.	Sauvegarde	15
4.2.	Application Demo	16
5.1.	Changement d'architecture	17
5.2.	Déroulement du projet	17
5.3.	Résultat	18
5.4.	Améliorations	18

Remerciements

Je remercie Monsieur Paul Albuquerque, Monsieur Pierre Kunzli, Monsieur Adrien Lescourt, ainsi que Monsieur Simon Fanetti qui m'ont soutenu durant ce projet.

Table des figures

Figure 1 - Diagramme de déploiement	4
Figure 2 - Diagramme de séquence pour afficher une image de courbe de niveau	5
Figure 3 - Projection dans un espace tridimensionnel	5
Figure 4 - Calcul de l'angle	6
Figure 5 - Rotation de l'écran projeté.....	6
Figure 6 - Calcul de la matrice	6
Figure 7 - Régression linéaire	7
Figure 8 - Calcul de la position du projecteur	7
Figure 9 - Distance la plus courte entre deux droites	7
Figure 10 – Diagramme de déploiement de la nouvelle conception	9
Figure 11 - Ancien processus de la génération de matrice de rotation	10
Figure 12 - Nouveau processus de la génération de matrice de rotation	10
Figure 13 - Ancien processus de calcul de position du projecteur	11
Figure 14 - Nouveau processus de calcul de position du projecteur	11
Figure 15 - Application de calibration.....	13
Figure 16 - Génération de la bordure.....	14
Figure 17 - Adaption de la bordure à la projection	14
Figure 18 - Détection de cercle mal placé	14
Figure 19 - Détection de cercle bien placé	14
Figure 20 - Copie du fichier de configuration	15
Figure 21 - Application Demo	16
Figure 22 - Détection des niveaux de courbes avec un objet proche	16
Figure 23 - Détection des niveaux de courbes avec un objet éloigné	16

Abstract

Ce rapport présente les modifications apportées au projet de Réalité Augmentée mobile dans un bac à sable afin d'obtenir une API réutilisable qui permette de calibrer une installation comportant un bac à sable, une caméra de profondeur Intel RealSense et un projecteur. Cela permettrait de développer différents projets avec cette API. Le projet a été fortement impacté par le confinement dû au Covid-19 ce qui a posé des problèmes de développement et a mis en évidence des faiblesses du système mobile. Une application mobile est sujette à de nombreux changements de contexte et une forte instabilité qui demande une implémentation solide. Il n'a pas été possible de résoudre ces problématiques dans le temps imparti, seulement de les observer. Le projet a toutefois permis de mettre en place une application de calibration utilisant l'API afin de créer un fichier de configuration et de créer une petite application de démonstration utilisant la même API.

1. Introduction

L'HEPIA a développé un bac à sable de réalité augmentée. Le dispositif est constitué d'un bac à sable sur lequel est projetée l'image d'un projecteur, ainsi que d'une caméra de profondeur pour analyser le bac à sable et un ordinateur sur lequel tourne l'application. L'application actuelle permet de projeter des courbes de niveaux sur le bac à sable. Cependant l'architecture logicielle actuelle ne permet pas de réutiliser du code pour développer de nouvelles applications. Il faudrait implémenter à nouveau la partie accès au matériel et la calibration du dispositif.

1.1. But de l'étude

Ce projet vise à transformer l'architecture logicielle de réalité augmentée mobile dans un bac à sable afin d'extraire une API réutilisable avec d'autres applications que celle déjà mise en place. Le résultat attendu est une bibliothèque dont la seule fonction est de calibrer et manipuler le bac à sable.

1.2. Portée et limites de l'étude

Ce projet vise à revoir l'architecture du logiciel fonctionnel afin d'obtenir une API utilisable dans différentes applications. Une simple application de démonstration basée sur le logiciel fonctionnel doit permettre de tester l'application. Les algorithmes ont été repris en appliquant uniquement les modifications nécessaires à la modification d'architecture, la conception des algorithmes n'est pas impactée par ce projet.

Le logiciel fonctionne de manière client-serveur. Il existe un client écrit en C++ et un en python. Seul le client C++ sera traité.

2. Analyse

2.1. Projet actuel

Il est composé d'un bac à sable, d'une caméra de profondeur, d'un projecteur et d'un ordinateur. Le principe est de calculer des niveaux de profondeurs du sable dans le bac et de générer ensuite une image en conséquence. Cette image générée est affichée à l'écran. Le projecteur est un duplicata de l'écran d'ordinateur pour ainsi projeter cette image sur le sable.

2.2. Technologies

La caméra de profondeur est une Intel RealSense modèle D415. Cette caméra possède des capteurs de profondeur, mais aussi une caméra conventionnelle. Le projecteur utilisé par l'HEPIA est un Sanyo modèle PLC-XU116, mais à cause du confinement dû au coronavirus, un projecteur InFocus modèle IN119HDx a été utilisé pour le développement.

L'ordinateur tourne sous Ubuntu 18 pour pouvoir utiliser les drivers et le SDK de la caméra de profondeur.

La procédure d'installation du SDK provient du site officiel du SDK 2.0 de Intel RealSense¹.

- librealsense2:amd64/bionic 2.34.0-0~realsense0.2251
 - librealsense2-dkms:all 1.3.13-0ubuntu1
 - librealsense2-utils:amd64/bionic 2.34.0-0~realsense0.2251
 - librealsense2-dev:amd64/bionic 2.34.0-0~realsense0.2251
 - librealsense2-dbg:amd64/bionic 2.34.0-0~realsense0.2251

OpenCV a été utilisé pour manipuler les frames. Il est important d'installer OpenCV avec « apt-get ». La version d'OpenCV à installer depuis Github est passablement différente et ne fonctionne pas dans ce projet.

- libopencv-dev:amd64/bionic-security 3.2.0+dfsg-4ubuntu0.1
- libopencv-dev:amd64/bionic-security 3.2.0+dfsg-4ubuntu0.1

```
sudo apt-get install libopencv-dev python-opencv
```

Qt Creator a été utilisé pour faciliter la mise en place d'interface graphique. L'installation depuis l'installateur fourni sur le site de Qt peut poser problème, certains chemins de bibliothèques sont différents.

- qtcreator:amd64/bionic 4.5.2-3ubuntu2
- build-essential:amd64/bionic 12.4ubuntu1

```
sudo apt-get -y install qtcreator build-essential
```

2.3. Logiciel actuel

Le logiciel actuel fonctionne en mode client-serveur qui communiquent par TCP. Le client contient la logique de l'application et communique avec le serveur grâce au client TCP de l'API. Il va demander des frames au serveur qu'il va ensuite traiter et renvoyer une frame au serveur afin qu'il l'affiche à l'écran. Le serveur utilise une bibliothèque afin de calibrer la caméra de profondeur et le projecteur. Toujours grâce à cette bibliothèque, il capture les frames à l'aide de la caméra de profondeur et affiche les frames du client à l'écran. Le projecteur projette cet écran sur le bac à sable.

¹ <https://www.intelrealsense.com/sdk-2/>

2.3.1. Architecture

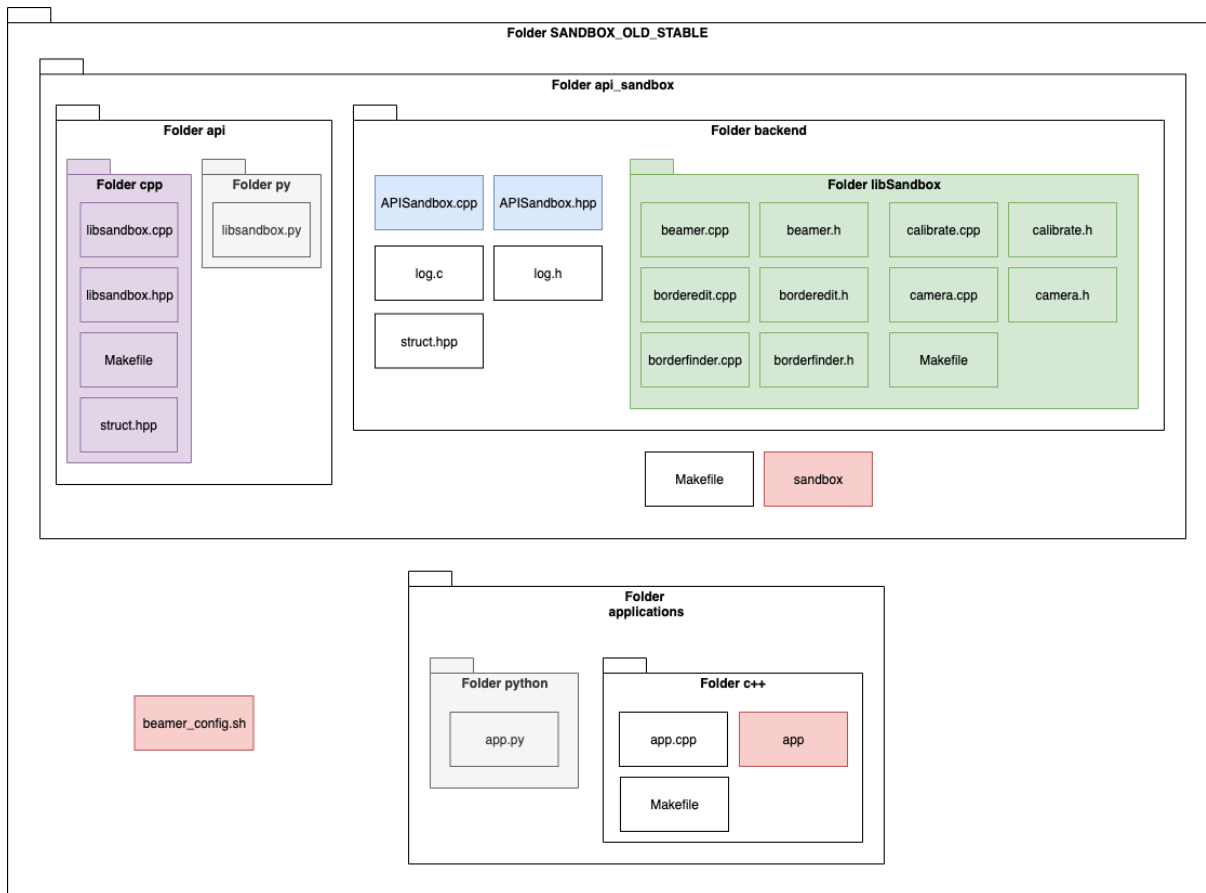


Figure 1 - Diagramme de déploiement

La figure ci-dessus représente le diagramme de déploiement global. Le script « `beamer_config.sh` » pour adapter la résolution de l'écran à celui du projecteur se trouve à la racine. Le dossier « applications » contient le client qui exécute la logique de l'application, une version en python et une version en C++. Cependant ce client ne contient pas de code pour communiquer avec le serveur uniquement la logique de l'application. Pour communiquer avec le serveur, elle doit faire appel au client TCP de l'API qui se trouve dans le dossier « api » (de couleur violette). Le client TCP contient uniquement la logique pour communiquer avec le serveur qui se trouve dans le dossier « backend » (de couleur bleu). Ce serveur utilise la bibliothèque « libsandbox » (de couleur verte) pour la gestion des interfaces graphiques et l'utilisation de la caméra de profondeur et du projecteur.

2.3.2. Fonctionnement

Les artefacts en rouge dans la figure 1 sont les artefacts à exécuter. L'utilisateur commence par exécuter le script « beamer_config.sh » pour adapter sa résolution d'écran à celui du projecteur. Ensuite, il lance « sandbox » afin d'effectuer la calibration et de démarrer le serveur. Lorsque le serveur est démarré, l'utilisateur peut lancer « app » pour démarrer la partie client.

L'application est prête à être utilisée. Le client a actuellement une fonctionnalité stable qui est l'affichage de courbe de niveau. Pour ce faire le client demande une frame qui passe par le client de l'API, puis est envoyé au serveur qui utilise la bibliothèque pour générer une frame avec la caméra de profondeur et la retourner. Ensuite il génère une image de courbe de niveau qu'il transmet au serveur à travers le client de l'API afin qu'il l'affiche à l'écran que projette le projecteur grâce à la bibliothèque.

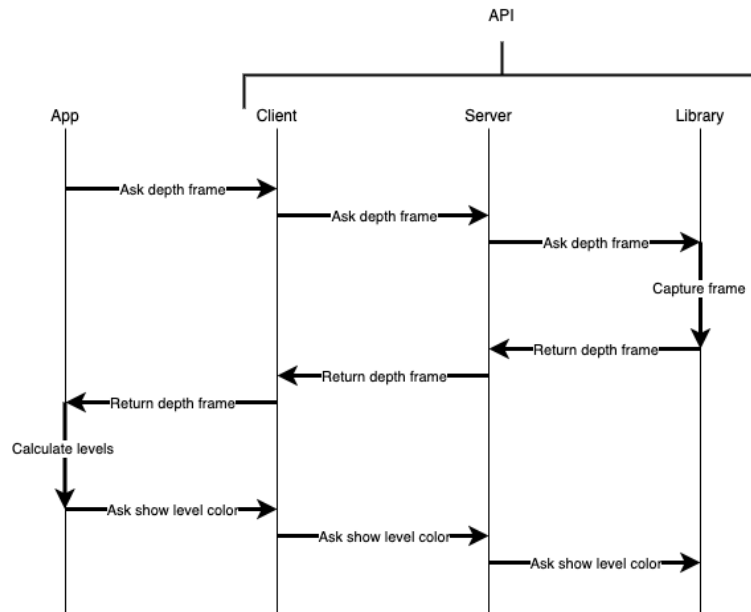


Figure 2 - Diagramme de séquence pour afficher une image de courbe de niveau

2.3.3. Calibration

Normalement un projecteur projette une image sur une surface plane et non sur une surface tridimensionnelle comme un bac à sable. Il faut par conséquent déformer l'image projetée pour l'adapter au bac à sable.

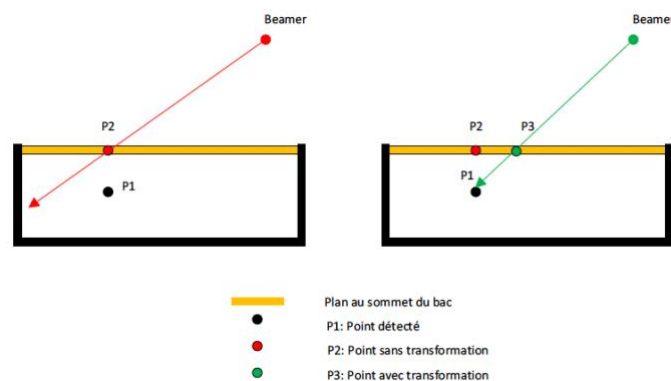


Figure 3 - Projection dans un espace tridimensionnel

Le point P2 est une projection normale d'un pixel, sans transformation, sur une surface 2D. Le point P1 est l'endroit où on souhaite projeter le pixel déterminé par la détection de profondeur. Il faut donc appliquer une transformation pour que le pixel soit projeté sur P3 plutôt que P2 afin d'atteindre le point P1.

L'objectif de la calibration est de définir tous les paramètres nécessaires à cette déformation. La calibration consiste à définir à quelle distance du bac à sable se trouve la caméra de profondeur, la position du projecteur par rapport à la caméra de profondeur et une matrice de rotation entre l'image projetée par le projecteur et celle récupérée par la caméra de profondeur.

Pour commencer, une planche plate est déposée sur le bac à sable afin de travailler sur une plane pour les premières étapes. La première étape est de déterminer la distance entre le bac à sable et la caméra de profondeur. Une moyenne est calculée à partir des distances d'un ensemble de points au centre de la caméra.

La seconde étape est d'identifier les contours du bac à sable. Pour ce faire, la caméra capture une frame RGB et le logiciel ajoute un rectangle par-dessus. L'utilisateur adapte manuellement le rectangle pour l'adapter au contour de l'image projeté par le projecteur. La caméra n'est pas forcément alignée au projecteur ce qui peut provoquer une légère rotation à compenser. Pour ce faire, nous calculons l'angle de rotation et générons une matrice de rotation grâce à celui-ci.

$$angle = \tan^{-1} \left(\frac{P3.y - P0.y}{P3.x - P0.x} \right)$$

Figure 4 - Calcul de l'angle

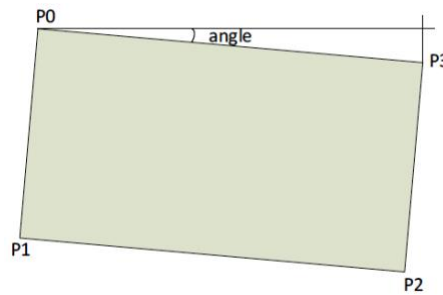


Figure 5 - Rotation de l'écran projeté

$$\begin{bmatrix} \alpha & \beta & (1 - \alpha) \cdot \text{center.x} - \beta \cdot \text{center.y} \\ -\beta & \alpha & \beta \cdot \text{center.x} + (1 - \alpha) \cdot \text{center.y} \end{bmatrix}$$

Avec :

$$\begin{aligned} \alpha &= \text{scale} \cdot \cos \text{angle}, \\ \beta &= \text{scale} \cdot \sin \text{angle} \end{aligned}$$

Figure 6 - Calcul de la matrice

La troisième étape est de déterminer la position du projecteur. Le concept est d'utiliser trois vecteurs de projections des pixels afin de retrouver leurs points d'origine, le projecteur. Pour déterminer un vecteur de projection, la surface plane est retirée et trois points sont mesurés à différentes hauteurs à partir d'un même pixel au sein de l'espace 3D du bac à sable. Ensuite une régression linéaire de ces points permet de calculer ce vecteur.

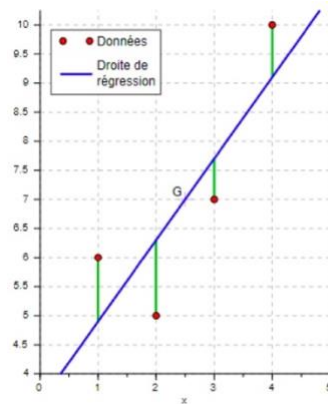


Figure 7 - Régression linéaire

Une fois les trois vecteurs obtenus, le croisement de ces vecteurs devraient théoriquement être la position du projecteur. Cependant dans la réalité des choses, les trois vecteurs ne se croisent pas en un unique point. Deux vecteurs peuvent ne pas se toucher du tout. Pour pallier ce problème, les deux points déterminant la distance la plus courte entre deux droites pour chaque paire de droites sont utilisés afin de calculer un centre de gravité. Ce centre de gravité est une bonne approximation de la position du projecteur.

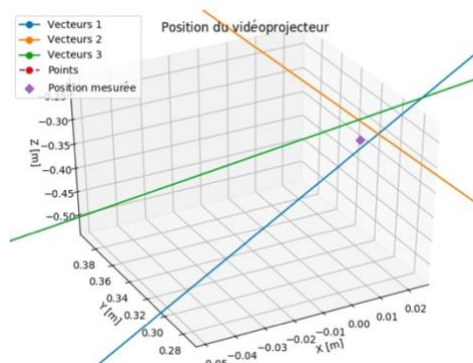


Figure 8 - Calcul de la position du projecteur

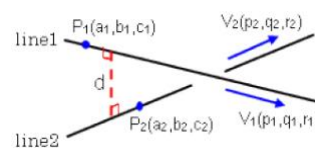


Figure 9 - Distance la plus courte entre deux droites

3. Conception

Le problème de la première version est que la bibliothèque a trop de responsabilités. Elle gère les accès au matériel, la logique de l'application et les interfaces graphiques. L'objectif est donc de limiter les responsabilités de l'API. La partie client-serveur qui permet de découpler les contraintes de langage entre l'API et le serveur a été abandonnée et la logique et la gestion d'interfaces graphiques ont été déplacées dans une application Qt dédiée.

3.1. Nouvelle architecture

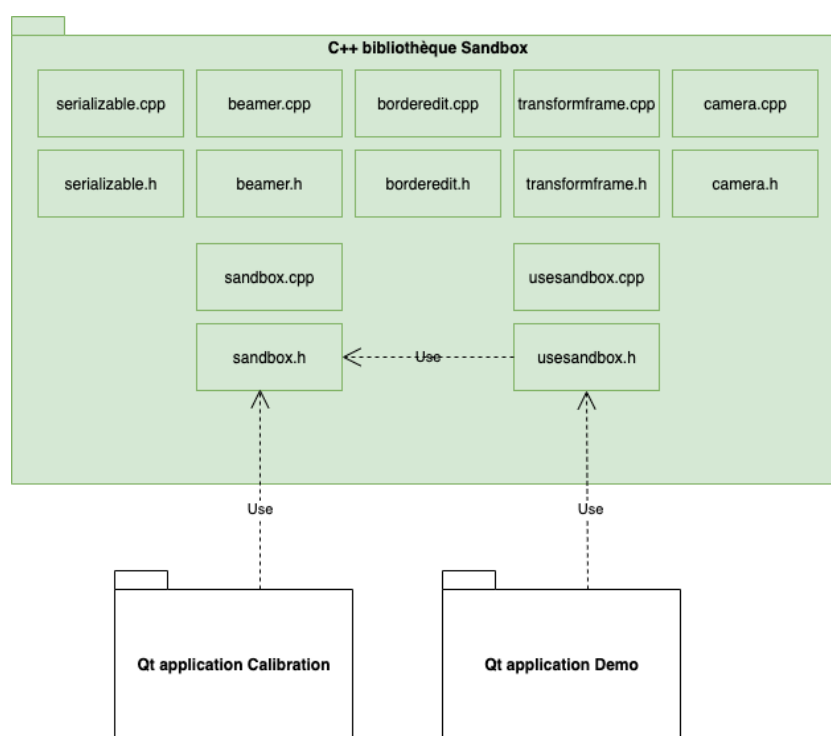


Figure 10 – Diagramme de déploiement de la nouvelle conception

La figure ci-dessus représente le diagramme de déploiement pour la nouvelle architecture centrée sur l'API. Elle ne contient plus que les méthodes pour calibrer et utiliser la caméra de profondeur et le projecteur. La partie logique et gestion des interfaces graphiques ont été déplacées dans des applications Qt. L'application Calibration gère la logique de calibration et permet de la sauvegarder. Ensuite cette calibration est importée dans d'autres applications telles que l'application Demo qui contient la logique applicative du bac à sable.

Il est à noter que l'application Calibration utilise la classe Sandbox qui sert d'interface à l'API tandis que l'application Demo utilise la classe UseSandbox. La classe UseSandbox fournit uniquement l'accès aux méthodes de la classe Sandbox nécessaires à l'utilisation de l'API, elle ne contient pas les méthodes pour la calibration.

3.2. Adaptation des processus de calibration

Il y a deux processus pour calibrer l'application : génération de la matrice de rotation et détection de la position du projecteur. Le calcul de la distance entre le bac à sable et la caméra de profondeur est simple et intégré dans le processus de génération de la matrice de rotation. Ils ont dû être adaptés pour que les interactions utilisateurs et l'affichage soient extraits de l'API et soient gérés par l'application Qt.

3.2.1. Génération de la matrice de rotation

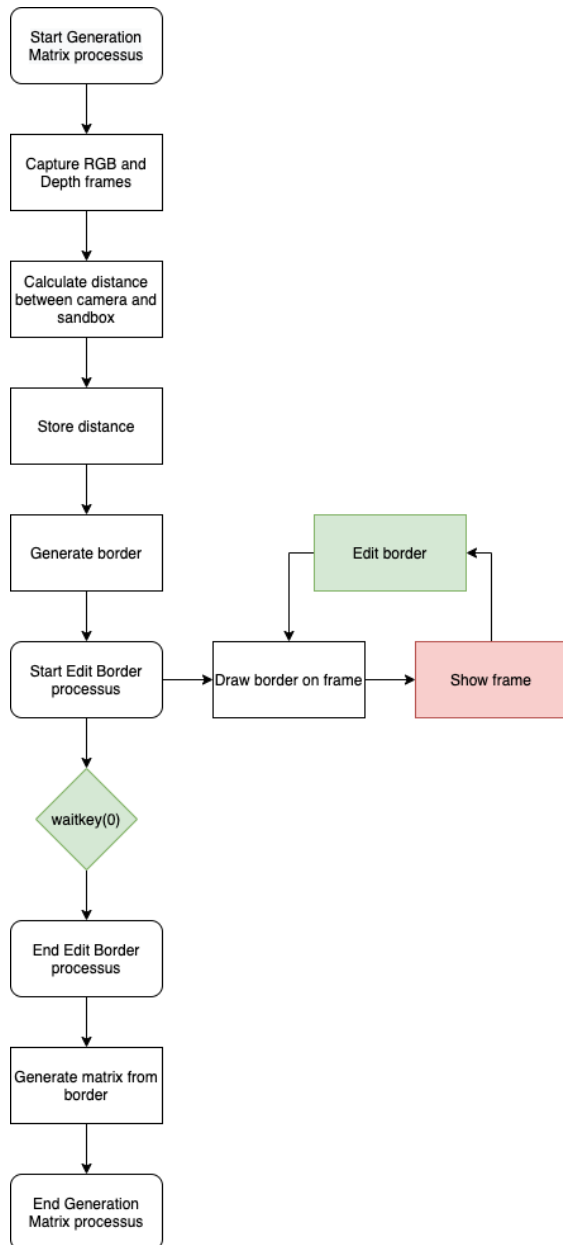


Figure 11 - Ancien processus de la génération de matrice de rotation

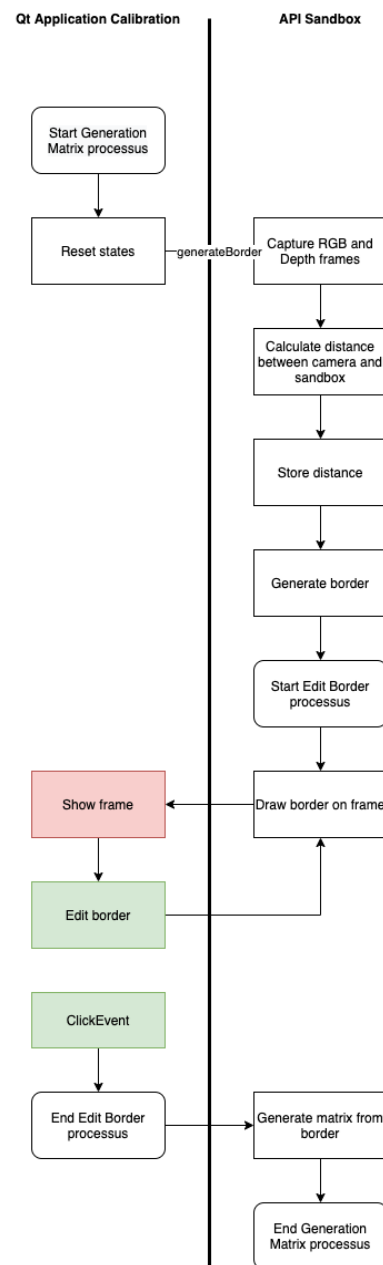


Figure 12 - Nouveau processus de la génération de matrice de rotation

On peut observer en vert les interactions utilisateurs et en rouge l'affichage des frames. Ainsi le contrôle et la logique sont extraits de l'API et se situent dans l'application Qt. Dans l'application Qt apparaît une étape un peu spéciale « Reset states ». Comme le processus est découpé en plusieurs parties qui passent d'un côté à l'autre, il a été décidé de stocker des états afin de limiter le nombre de paramètres à transmettre.

3.2.2. Calcul de la position du projecteur



Figure 13 - Ancien processus de calcul de position du projecteur

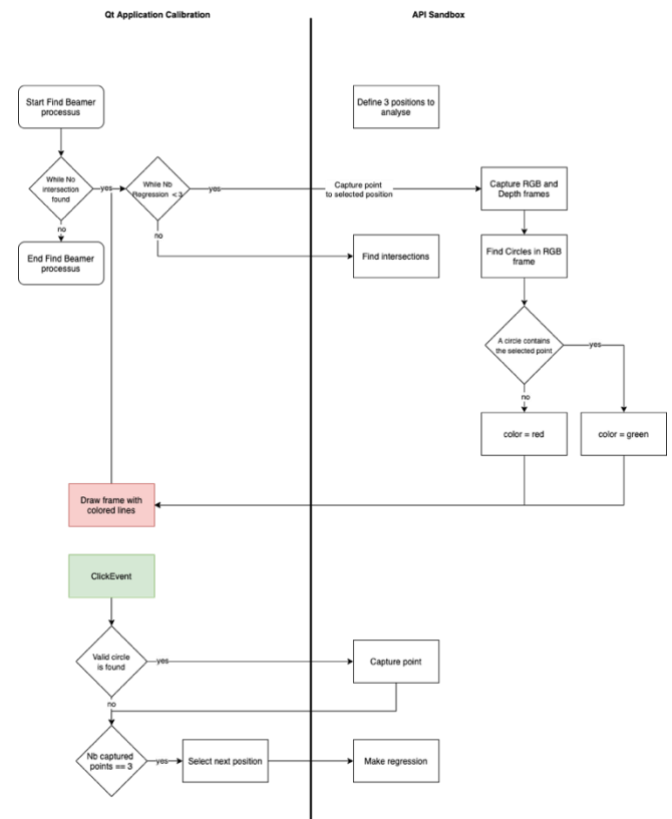


Figure 14 - Nouveau processus de calcul de position du projecteur

Comme pour la génération de matrice, les interactions utilisateurs et l'affichage sont attribués à l'application Qt. Il n'y a pas de réinitialisation d'état, car ce processus suit directement le processus de la génération de matrice et donc les états sont déjà réinitialisés.

4. Résultats

A cause du confinement, la réalisation de ce projet n'a pas pu être faite avec un bac à sable à disposition et le matériel adéquat en position fixe. La mise en place devait être faite à chaque utilisation. La projection se faisait simplement sur un mur. La distance entre le mur et les appareils n'était pas optimale. Cela impacte la cohérence des différentes images qui vont être montrées dans ce chapitre ainsi que la qualité du résultat. Cela sera traité dans le chapitre Discussion.

4.1. Application Calibration

Dans un environnement de production, l'installation est fixe. Par conséquent la calibration ne change pas et n'est à réaliser qu'une seule fois. L'application permet donc de faire cette unique calibration et la stocke dans un fichier texte.

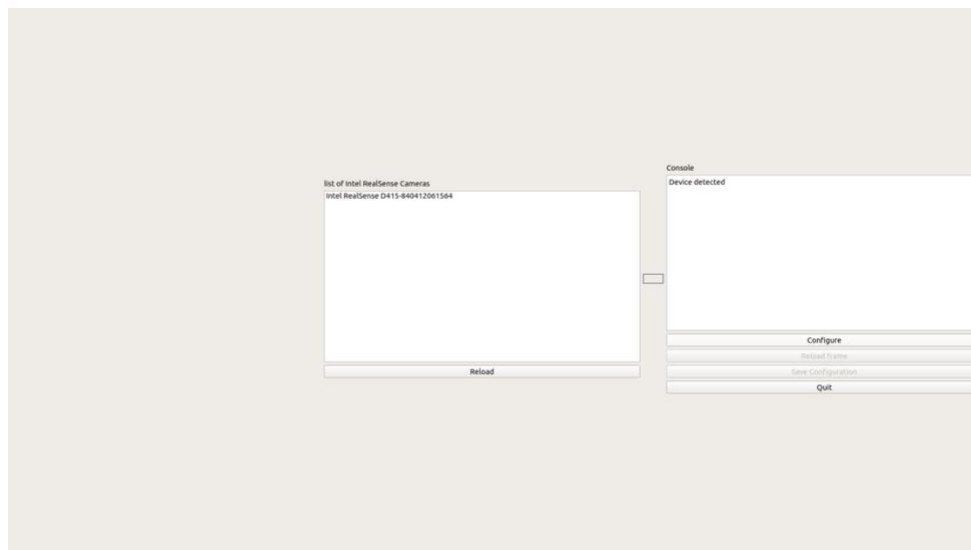


Figure 15 - Application de calibration

L'application liste les périphériques Intel RealSense et permet de sélectionner la caméra. Cela permet de contrôler que la caméra est bien connectée et accessible. Le petit rectangle est un label qui va s'adapter et permettre l'affichage des frames. La console à droite fournit un retour utilisateur sur les étapes effectuées.

Le système fonctionne comme une machine à états en passant d'étape en étape lors de la calibration. Les deux premiers boutons vont changer de fonctions en fonction de l'étape en cours. Ceci afin d'éviter de surcharger l'interface avec une interface trop lourde et complexe. La sauvegarde n'est accessible qu'une fois le processus terminé. Si la calibration est relancée, la sauvegarde est à nouveau inaccessible.

4.1.1. Génération de la matrice de rotation



Figure 16 - Génération de la bordure

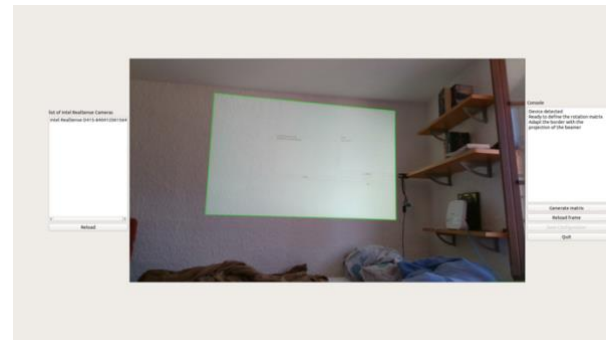


Figure 17 - Adaption de la bordure à la projection

La frame remplit le label mis en place à cet effet. La classe QLabel a été surchargée afin de permettre l'interaction de la souris avec la bordure. L'utilisateur aligne la bordure avec la projection puis lance la génération de matrice de rotation. Dans l'exemple ci-dessus, on voit une forte déformation entre la projection et d'un rectangle et ce que récupère la caméra. L'impact de cette déformation est traité dans le chapitre Discussion.

4.1.2. Calcul de la position du projecteur

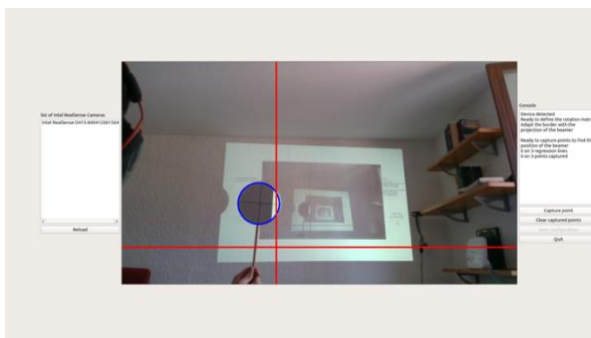


Figure 18 - Détection de cercle mal placé

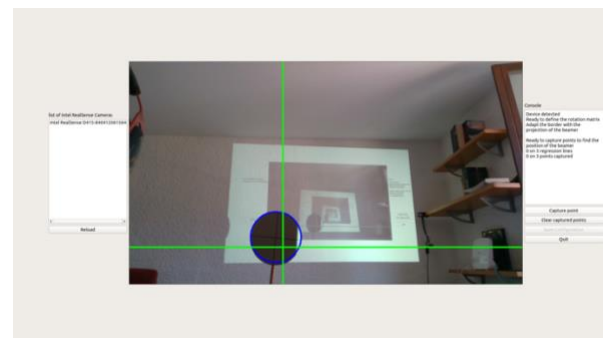


Figure 19 - Détection de cercle bien placé

Le centre de la croix représente la position à laquelle capturer les points pour la régression linéaire. Dans le premier exemple, le cercle est bien détecté, mais mal positionné. Dans le deuxième exemple, la distance entre la position et le centre du cercle est inférieure au rayon du cercle ce qui indique que la détection de profondeur se fait bien et les lignes deviennent vertes. La détection de plusieurs cercles ne pose pas de problème, car seul celui assez proche de la position est pris en compte.

Une fois les lignes devenues vertes, Le point peut être capturé. Trois points doivent être capturés à différentes profondeurs pour appliquer une régression linéaire et détecter un vecteur de projection. Le processus est reproduit trois fois à trois positions différentes.

4.1.3. Sauvegarde

Une fois le processus terminé, le bouton de sauvegarde est activé. La sauvegarde génère un fichier nommé « device » qui contient la configuration. Ce fichier contenant la configuration est à déplacer dans le dossier d'exécution de l'application de réalité augmentée utilisant le bac à sable.

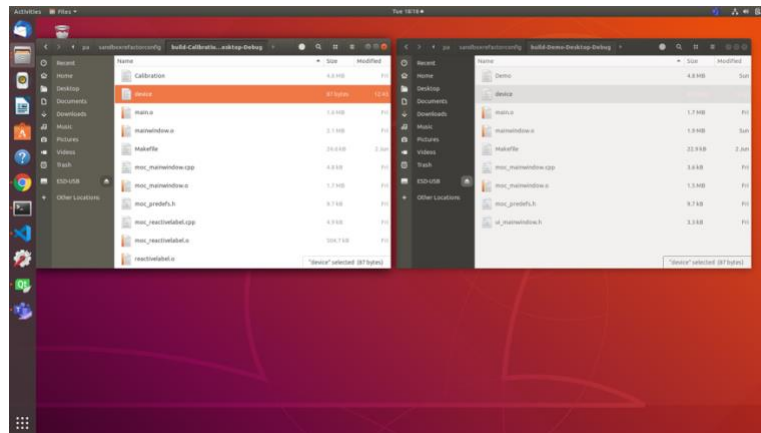


Figure 20 - Copie du fichier de configuration

4.2. Application Demo

L'application Demo sert uniquement à confirmer le fonctionnement de l'application. Elle n'a donc pas été grandement travaillée.

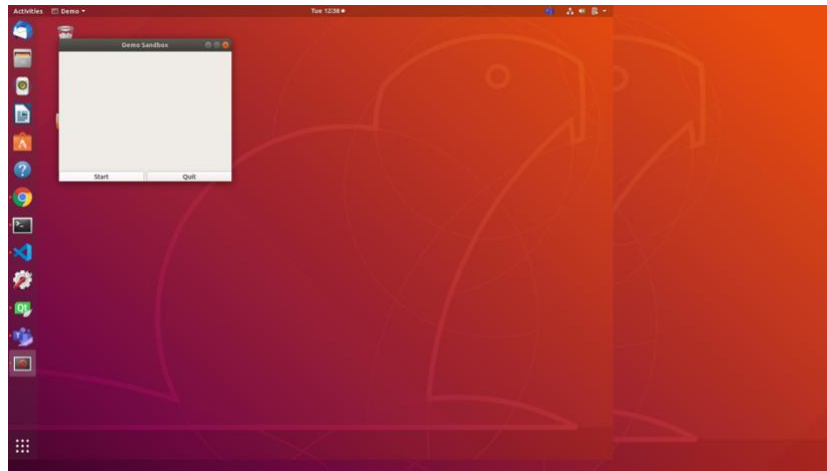


Figure 21 - Application Demo

Un simple bouton permet de lire le fichier de configuration et calibrer le système. Ensuite l'application calcule les niveaux de courbes du bac sable.

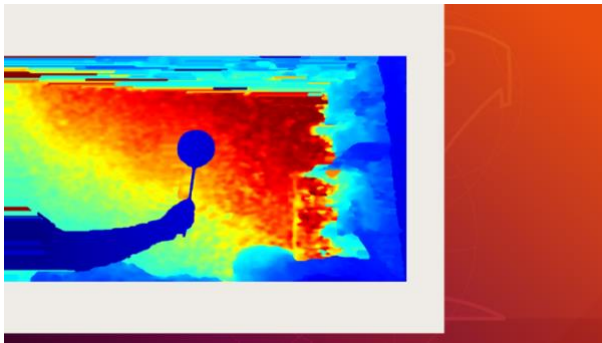


Figure 22 - Détection des niveaux de courbes avec un objet proche

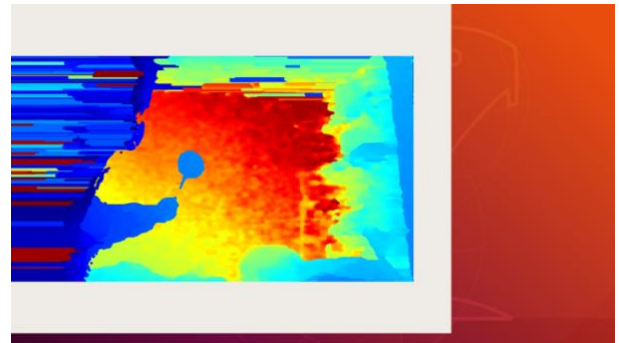


Figure 23 - Détection des niveaux de courbes avec un objet éloigné

Les objets les plus proches sont en bleu foncé et les éloignés en rouge avec un dégradé entre les deux passant par le bleu clair, puis le jaune. On peut l'observer avec le petit cercle. L'imprécision du système provient de plusieurs choses. Les objets ne sont pas à une distance optimale de la caméra. Le mur est trop éloigné. Sur la figure n°22, les lignes sur la gauche proviennent du fait que la personne est trop proche. Il y a de nombreux artefacts à cause de la problématique de l'utilisation du système dans le cadre du développement et une mauvaise calibration. Cet aspect sera traité dans le chapitre Discussion.

5. Discussion

5.1. Changement d'architecture

L'architecture de la première version offrait une bonne interopérabilité grâce au client-serveur. Les problèmes du projet originel pour une bonne réutilisabilité sont la répartition des responsabilités et les codes fantômes.

Certains codes sont dupliqués ou non-utilisés. La structure devient obscure et mal commentée. Les méthodes deviennent lourdes et ont plusieurs fonctions. Le tout rend confus l'utilisation et l'adaptation du code source.

La bibliothèque gère les accès au matériel, une partie de la logique, les interfaces graphiques et les interactions utilisateurs. Cela empêche la création de nouvelles interfaces graphiques, de nouvelles logiques et donc de nouvelles applications.

La nouvelle version s'est concentrée sur la création d'une API qui gère uniquement l'accès au matériel et la manipulation des frames. La gestion des interfaces graphiques a été déplacée dans l'application cliente. Cela permet de créer de nouvelles applications facilement. Il est aisé de réintégrer la technologie client-serveur entre la bibliothèque et l'application cliente pour retrouver l'interopérabilité. Le serveur et le client TCP servent de pont pour que n'importe quel client utilise la méthode de l'API.

5.2. Déroulement du projet

Le projet a été fortement impacté par le confinement durant le développement. Le matériel ne correspond plus à celui de l'environnement utilisé pour la première version qui potentiellement deviendrait un environnement de production. La disposition n'était pas optimale. Cela a conduit à des problèmes de précision de la caméra de profondeur qui fonctionne dans une plage de profondeur précise. L'angle entre la projection et l'image recueillie est beaucoup plus importante ce qui implique de plus fortes transformations et conduit à plus d'imprécisions. Le matériel n'étant pas fixé, il avait tendance à se déplacer légèrement ce qui fausse la calibration.

La phase de calibration a été complexifiée par la nouvelle installation. Une deuxième personne était nécessaire pour calibrer le matériel, il était impossible d'appuyer sur le bouton tout en plaçant le disque pour la capture de point. Le problème de luminosité provenant du projecteur et de l'extérieur rendait difficile la détection du disque. L'absence de bac rendait difficile le placement du disque sur une même plage de profondeur. L'ensemble de ces problèmes a provoqué de grosses imprécisions humaines.

Le changement de matériel a aussi posé un problème de résolution d'écran. Le script permet d'adapter la résolution de l'écran pour convenir au projecteur. Cependant le nouvel ordinateur modifie différemment la résolution. Au lieu d'adapter l'écran à la résolution, l'ensemble des éléments sont redimensionnés et astreints à une zone correspondant à la résolution ce qui rend visible un débordement observable dans différentes figures telle que la n°22. Dans cette figure, le fond d'écran est visible alors que l'application est en pleine écran. Cependant dans le cas de la configuration l'application prend tous l'écran comme dans la figure n°16 ce qui implique que l'application n'est pas projetée en entier sur le mur, l'application n'est pas redimensionnée pour la résolution du projecteur ce qui génère des erreurs lors de la calibration.

5.3. Résultat

L'ensemble des problèmes évoqués dans le sous-chapitre Déroulement du projet expliquent le manque de qualité de la détection de courbe de niveau. Cependant l'objectif du projet était d'extraire une API qui ne s'occupait que de l'accès matériel et de la manipulation de frame. L'objectif a été pleinement atteint et la nouvelle version est pleinement opérationnelle. Avec une installation convenable, la qualité du logiciel devrait en toute logique correspondre à celle de l'ancienne version, mais avant de la mettre en production, il est nécessaire de le confirmer avec des tests.

5.4. Améliorations

La phase de détection des cercles a démontré quelques problèmes de détection avec des RGB frames à cause de la luminosité. Il serait préférable d'utiliser la caméra de profondeur pour détecter les cercles, car moins dépendant de la luminosité.

La méthode de régression linéaire et du calcul de centre de gravité est fortement liée à la précision des mesures qui dépendent de la précision humaine. Si la situation n'est pas optimale, il peut être compliqué d'obtenir une qualité suffisante. Il pourrait être bon de changer de méthode pour pallier ces problèmes. Une solution serait d'analyser les coins de la projection. De par les caractéristiques du projecteur, il est possible de déterminer l'angle de projection entre une surface plane et le projecteur au niveau des coins. Il est possible de faire de même pour la caméra profondeur. Une fois les angles à disposition, il est possible de retrouver la position du projecteur en fonction de la surface plane et donc de calculer cette position avec comme origine la caméra de profondeur.

Il serait utile de développer un système de logs pour s'assurer du fonctionnement du logiciel.

6. Conclusion

Malgré le retard et les problèmes dû au confinement et un environnement de développement inadapté, le projet a atteint son objectif principal qui est d'extraire une API réutilisable pour accéder au matériel et manipuler les frames. Cependant, tous les objectifs secondaires ont dû être mis de côté et les applications ont été développées sous forme de prototype. L'API est utilisable dans un projet d'application de réalité augmentée mobile dans un bac à sable. L'application de calibration nécessite un travail sur l'ergonomie et le retour utilisateur.

Yverdon-les-Bains, le 10 juillet 2020.

Fabien Mottier

7. Références

- [1] Adow, S. (2019). *Réalité augmentée dans un bac à sable pour l'aménagement du territoire*. Genève: HEPIA.
- [2] Intel. (2020, 07 07). *Intel RealSense SDK 2,0*. Retrieved from www.intelrealsense.com: <https://www.intelrealsense.com/sdk-2/>
- [3] team, o. d. (2019, 12 31). Retrieved from OpenCV documentation: <https://docs.opencv.org/2.4/index.html>

8. Appendices

- Les PV du projet

Appendix I

PV n°1

Date : 18.02.2020

Participants :

- Paul Albuquerque
- Fabien Mottier
- Adrien Lescourt
- Pierre Kunzli

Objectifs :

- Rencontre des intervenants
- Présentation du projet

L'objectif premier du projet est de récupérer l'ancien projet et de l'adapter pour obtenir une API qui permette l'abstraction de la programmation hardware. Dans un second temps, une démo sera développée pour valider l'API.

Etat du projet :

- calibrage de l'appareil est bon
- pas d'abstraction au moyen d'une API
- il manque un système de sauvegarde des configurations
- mode client-serveur

PV n°2

Date : 24.02.2020

Participants :

- Fabien Mottier
- Simon Fanetti
- Pierre Kunzli

Objectifs :

- Démonstration du projet
- Explication de la structure du projet

Monsieur Fanetti m'a expliqué la structure globale du projet. Il y a une app(client) qui appelle un client sandbox(dans le dossier api) qui fait de multiples appels au serveur(sandbox).

PV n°3

Date : 03.03.2020

Participants :

- Paul Albuquerque
- Adrien Lescourt
- Pierre Kunzli
- Fabien Mottier

Objectifs :

- Faire le point entre le projet de Simon Fanetti et le mien
- Réévaluation des premières tâches

Les deux projets sont indépendants. Chacun développe de son côté, mais partage les informations.

Suite à des soucis de calibrage, il a été supposé que cela venait de la différence de position entre la caméra RGB et celle de profondeur. Il a donc été convenu que je fasse une sorte d'audit du calibrage.

PV n°4

Date : 10.04.2020

Participants :

- Paul Albuquerque
- Adrien Lescourt
- Pierre Kunzli
- Fabien Mottier

Objectifs :

- Réévaluer le projet à cause du confinement

J'avais des problèmes d'installation du matériel qui m'ont fait presque prendre un mois de retard, que j'ai cependant pu régler la veille de l'entretien. Le point a été sur l'avancée du projet. Messieurs Albuquerque, Lescourt et Kunzli sont relativement satisfait de la direction prise par le projet.

Il a été toutefois convenu qu'au vu de la situation et du retard déjà pris de diminuer le cahier des charges. L'objectif est uniquement d'obtenir une API réutilisable qui permet de faire la calibration et d'accéder au matériel et de gérer les frames. Les contraintes étant que l'API ne soient pas dépendantes de Qt et que l'affichage ne soit pas géré par l'API.

PV n°5

Date : 12.05.2020

Participants :

- Paul Albuquerque
- Adrien Lescourt
- Pierre Kunzli
- Fabien Mottier

Objectifs :

- Problème d'utilisation de la caméra
- Faire le point sur l'état du projet

Lorsque je récupérais les RGB frames de la caméra, j'avais de nombreuses taches noires qui apparaissaient à l'écran. Ce qui générait de nombreux cercles et rendait le processus inutilisable. Je ne comprenais pas d'où venait le problème. Grâce à messieurs Albuquerque, Lescourt et Kunzli, nous avons découvert qu'il s'avérait que c'était dû à un mauvais alignement des frames RGB et de profondeur.

J'avais toutefois avancé le projet sans pouvoir le tester. Une fois ce problème résolu, le test du projet a démontré une avancée convenable qui les a satisfaits.

PV n°6

date: 26.05.2020

participants:

- Paul Albuquerque
- Adrien Lescourt
- Pierre Kunzli
- Fabien Mottier

objectif:

- faire le point sur l'état du projet
- redéfinition de la date de rendu

Le projet est quasiment fonctionnel et a atteint ses objectifs. Il a été décidé de garder un code léger et de ne pas trop effectuer de tests. Il faut plutôt écrire un bon README d'utilisation. C'est une technique de programmation de type par contrat plutôt que défensive.

Au vu des problèmes de confinement et des informations fournies par le rectorat, il a été décidé de repousser le rendu au 10 juillet 2020.