Université Paris 8 - Vincennes à Saint-Denis

Licence Informatique

Mémoire de projet tuteuré
Qualification de caméras RGB-D

Yasmine BOUDJEMAÏ
Mélanie DE JESUS CORREIA

Organisme d'accueil :	Université Paris 8 Vincennes-Saint-Denis
Tuteur – Organisme d'accueil :	Farès BELHADJ

Dédicaces

À la mémoire de mon très cher frère bien-aimé ***Belkacem*** qui a contribué pleinement à ma réussite sur différents plans. Pour son grand soutien moral et intellectuel que je n'oublierai jamais . Pour tous les moments de complicité passés dans des ambiances où bonheur, joie et rire s'entremêlaient. Il était tel un troisième parent pour moi, se souciant constamment de mon bien-être. Il était également tel un frère jumeau car ressentant quand j'avais besoin de soutien et d'aide ne serait-ce que minime était-elle. Ainsi que pour toutes les autres choses qu'il a su m'apporter que la page ne pourrait citer.

Remerciements

Je tiens à remercier toutes les personnes qui m'ont apporté une quelconque contribution durant ce stage.

Je souhaiterais tout d'abord adresser mes sincères remerciements à mon maître de stage **M.Farès BELHADJ** pour m'avoir accordée sa confiance pour les missions sollicitées. Ainsi que de m'avoir aidée lors de la conception du modèle réel et virtuel et pour tous ses conseils précieux qui ont été utiles pour le bon déroulement du stage.

Par ailleurs, je voudrais remercier **Mme.Martine LE LEZ** pour m'avoir aidée pour la recherche de stage.

Enfin, je voudrais exprimer toute ma reconnaissance et toute ma gratitude envers toute ma famille. Un grand merci pour mes parents pour leur soutien moral quotidien inestimable. Je souhaite également remercier mes frères ainsi que ma belle-sœur et mon petit neveu pour leurs contributions lors de la réalisation du modèle réel.

Résumé

Dès lors de sa sortie, de nombreux domaines ont connu un essor important à l 'aide des technologies RGB-D. À titre d'exemple, on peut citer Natural-Pad [1] qui se sert de la Kinect pour la capture de mouvements pour des besoins de rééducation.

Dans des applications où la précision demeure nécessaire pour de meilleurs résultats, il est indispensable de savoir choisir le capteur qui convient le mieux aux exigences du contexte de l'application. Dans ce document, nous expliquons comment nous avons réalisé notre application pour qualifier un ensemble de caméras RGB-D. Nous exposons les résultats produits par notre application, puis discutons de de la fiabilité des résultats obtenus.

Dans le premier chapitre, nous proposons un état de l'art des technologies de la caméra RGB-D. Dans le second, nous détaillons les étapes par lesquelles nous sommes passés pour élaborer l'outil répondant à la demande du stage, ainsi que les résultats obtenus. Dans le troisième, nous montrons comment procéder pour utiliser un modèle bien précis de caméra afin de démarrer les tests avec ce dernier. Enfin, en conclusion, nous exposons succinctement le bilan du travail réalisé.

1. Entreprise, basée à Montpellier, spécialisée dans la création de jeux vidéo ludiques au service des besoins médicaux.

Table des matières

Table des figures

Introduction

La caméra RGB-D dans ses débuts, notamment la Kinect, a beaucoup servi le monde du jeux-vidéo. Cependant, depuis un certain nombre d'années, elle contribue activement et fondamentalement dans le développement de multiples applications dans diverses disciplines telles que la médecine, la robotique et plus généralement l'aide à la personne.

Ce stage, effectué au sein de l'Université Paris 8 Vincennes-Saint-Denis, a pour objet la conception d'outils pour la qualification de caméras RGB-D. Il vise à comparer différentes caméras afin de sélectionner la meilleure en fonction du besoin ciblé.

Tout d'abord, avant de passer à l'étape des qualifications, il nous faut concevoir un modèle-étalon sur lequel nous passerons nos tests de mesure. Ce dernier est fait principalement de polystyrène. Ensuite, nous fabriquons sa version virtuelle en 3D en ayant recours au logiciel *123D Design*.

Par la suite, nous exploitons certaines fonctions OpenCV telles que matchTemplate afin de détecter le modèle-étalon.

Enfin, nous opposons les données OpenGL et les données des caméras que nous avons pu réunir à partir de l'aire du modèle détecté en se servant de méthodes différentes pour le calcul du taux d'erreurs tel que le RMSE (Root Mean Square Error) [2]. L'application produit automatiquement un rapport mettant en avant ces divergences.

2. Méthode permettant de calculer l'écart-type des erreurs enregistrées. Se reférer à cette page web pour davantage de renseignements : `https://www.statisticshowto.datasciencecentral.com/rmse/`

Chapitre 1

État de l'art

Dans ce chapitre, nous allons définir ce qu'est une caméra RGB-D, nous énumérons certains des différents modèles existants que nous comptons utiliser pour ce projet, nous abordons brièvement les différentes techniques utilisées pour la récupération de la profondeur par une caméra. Enfin, nous discutons les différences recensées sur la `kinect V2` et `V3`.

1.1 Description d'une caméra RGB-D

La caméra RGB-D, aussi appelée capteur RGB-D, comprend un capteur RGB et un capteur de profondeur (D pour Depth). Ces derniers capturent des images couleur ainsi que des informations de profondeur par pixel. C'est principalement ces dernières qui vont nous intéresser tout au long des qualifications.

1.2 Modèles existants

Il existe différents modèles de caméras RGB-D. Parmi elles, nous pouvons citer la Kinect et ses différentes versions, Asus Xtion Pro Live, BlasterX Senz3D, Orbbec, Intel RealSense D415, ...

`Kinect` a fait son apparition en septembre 2008. Elle a été conçue par Microsoft et était destinée pour les consoles de jeux XBox 360. Elle permettait aux utilisateurs d'interagir avec la console à l'aide d'une NUI[1] en utilisant les mouvements gestuels et une reconnaissance vocale. Elle sera plus tard

1. Natural User Interface (Interface Utilisateur Naturelle), se réfère à une interface utilisateur invisible.

utilisée dans les domaines de la recherche et du développement pour différents secteurs comme le domaine de la médecine, l'industrie automobile, la robotique, l'éducation, ...

`Asus Xtion Pro Live` est le modèle de référence que nous utilisons afin d'effectuer les qualifications. Elle utilise la technologie PrimeSense [2] pour la détection de mouvements.

1.3 Mesure d'erreurs

La mesure d'erreur est un processus qui permet d'évaluer l'écart entre la valeur mesurée et la valeur de référence qui est soit exacte ou connue. L'objectif de la mesure d'erreurs est de jauger à quel degré les deux valeurs sont proches. Dans cette section, nous faisons un détour sur les moyens théoriques et physiques employés à cet effet.

1.3.1 Types et sources d'erreurs

Il existe deux types d'erreurs : aléatoires et systématiques.
Les erreurs aléatoires sont principalement dues aux restrictions de précisions de l'outil de mesure.
Les erreurs dites systématiques sont des erreurs reproductibles mais difficilement détectables même en augmentant le nombre d'observations contrairement aux erreurs aléatoires.

Ces erreurs peuvent provenir de différentes sources ; elles peuvent être dues à des facteurs environnementaux, à la résolution de l'instrument utilisé, au calibrage de l'appareil employé ou même à des erreurs humaines.

2. Connu principalement pour sa licence de conception matérielle et de puce employée dans le mécanisme de détection de mouvements de la Kinect XBox360. Pour plus d'informations, le lecteur peut se référer à ce lien `https://www.crunchbase.com/organization/primesense#section-web-traffic-by-similarweb`.

1.3.2 Mesure d'erreurs à l'aide de matériels

1.3.3 Mesure d'erreurs à l'aide de moyens théoriques

1.4 Technologies utilisées pour la depth d'une caméra RGB-D

Des méthodes employées pour la récupération de la depth, on peut en citer deux principales.

1.4.1 Infrarouge

Le projecteur infrarouge de certaines caméras le possédant qu'on peut voir sur la figure 1.1 projette un spectre infrarouge sur la scène captée. Le motif produit sur cette dernière sera capté par la caméra infrarouge et sera par la suite comparé à la base de données de motifs de référence stockés au préalable dans la caméra. Ces derniers seront indispensables pour la mesure de la profondeur de chaque pixel.

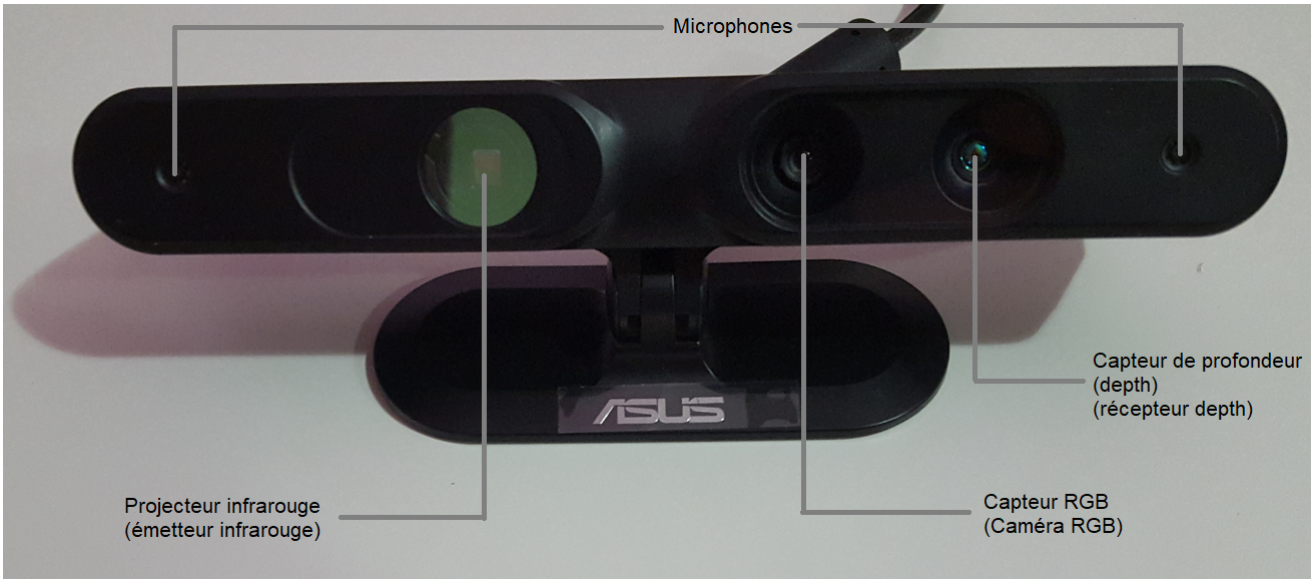


FIGURE 1.1 – Composants caméra RGB-D (Ici Asus Xtion Pro Live).

Par la suite, les valeurs obtenues seront corrélées à un capteur RGB qu'on peut apercevoir sur la figure. Ces données pourront être représentées par un

nuage de points [3].

Toutefois, ce type de caméras possède certaines restrictions. Parmi ces dernières, on peut citer :

— Distances de mesure limitées.

— Problèmes de calculs des informations de profondeur à l'encontre de surfaces brillantes, très mates, transparentes, réfléchissantes ou encore envers des objets absorbants.

— Interférence des motifs (patterns) infrarouges si présence de plusieurs caméras RGB-D de même type. En effet, chaque capteur visualisera ses propres motifs ainsi que ceux des autres caméras présentes et ne saura distinguer les siens des autres qui se chevauchent. De cela découle une perte considérable d'informations de profondeur comme nous pouvons l'observer sur la figure 1.2.

FIGURE 1.2 – (De gauche à droite) image renvoyée par la caméra RGB, image IR, depth map (une seule caméra), depth map (deux caméras). (figure tirée de l'article de *F.Alhwarin, A.Ferrein* et *I.Scholl* ayant comme titre *IR Stereo Kinect : Improving Depth Images by Combining StructuredLight with IR Stereo*).

Cependant, des travaux de recherches ont été conduits afin d'y remédier. Parmi eux, on peut citer le travail réalisé par l'équipe de *Rafibakhsh* ([**RAFIBAKHSH15**]) qui recommande de laisser un angle de 35° entre deux caméras suspendues à la même hauteur en considérant les scènes captées dans de bonnes conditions et une interférence très faible. *Maimone* et *Fuchs* [**MaimoneFuchs15**] proposent un algorithme de remplissage et de lissage en modifiant le filtre médian aux zones trouées à l'exception des bords. Quant à *F. Kenton Musgrave, Craig E.* et *Robert S. Mace* [**KentonCraigMace12**], ils appliquent une certaine quantité minime de mouvements (en utilisant des composants matériels supplémentaires) à certains capteurs de sorte que chacun puisse voir son propre motif infrarouge de façon nette et une version floue des motifs de ses voisins.

3. Est une représentation des points de coordonnées tridimensionnelles dont chacun peut avoir des attributs qui lui ont sont propres.

1.4.2 Stéréo-vision

Une caméra stéréoscopique est un appareil qui contient deux voire plus de capteurs d'images. Ceci nous rappelle la vision binoculaire humaine. En effet, ce mécanisme permet au système nerveux central de percevoir simultanément les images issus de chaque œil envoyées sous forme de signaux. Ainsi, il sera en mesure de se servir de ces différences (entre les deux images) pour permettre une vision stéréoscopique pour la perception de relief et une mesure des distances en utilisant la triangulation [4]. Le concept que nous venons d'expliquer est appelé disparité stéréoscopique [5].

La stéréo-vision sert principalement à reconstituer la scène observée sous forme de modèle 3D.

Carte de disparités

Le principe de la disparité est une approche du mécanisme humain vu précédemment. Il consiste en la différence entre les coordonnées pixels d'un point bidimensionnel d'une image et celles de son correspondant (présent sur une autre image prise au même moment). Ainsi, en appliquant le même traitement sur tous les pixels correspondants, on obtient la carte des disparités. Une carte est dite éparse lorsqu'une disparité est associée à quelques pixels et lorsque cette dernière est associée à chaque pixel, on dit d'elle qu'elle est dense.
Une formule pour calculer la profondeur en fonction de la disparité s'obtient comme suit :

$$z = \frac{B.f}{d} \tag{1.1}$$

Avec :

z représentant la profondeur.
B *Baseline* représente la distance séparant les deux capteurs.
f La distance focale en pixels calculée comme montré ci-après en nous servant de la figure 1.3 .

4. Approche géométrique permettant une mesure des distances. Le lecteur peut consulter cette page web pour de plus amples informations : `https://fr.wikipedia.org/wiki/Triangulation` .

5. Différence dans la localisation d'un objet perçu par l'œil gauche et l'œil droit résultant de la séparation horizontale des yeux dont certaines caméras essaient de s'approprier la technique afin de récupérer la profondeur.

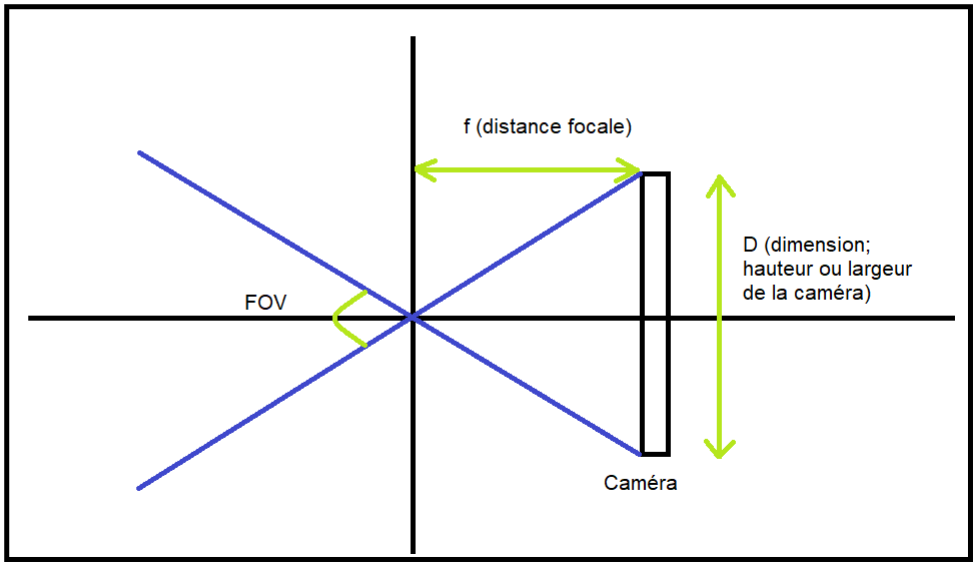


FIGURE 1.3 – Schéma illustrant la relation entre la distance focale et le champ de vision (fov).

$$\tan\left(\frac{fov}{2}\right) = \frac{D}{2.f} \Leftrightarrow f = \frac{D}{2.\tan\left(\frac{fov}{2}\right)} \tag{1.2}$$

Avec :

D dimension qui peut représenter soit la largeur soit la hauteur en fonction du fov de la caméra employé.
fov (Field Of View) représente l'angle du champ de vision (soit horizontal ou vertical).
f représente la distance focale.

1.5 Kinect V2 et Kinect V3

1.5.1 Kinect V2

`Kinect V2` se sert de la méthode TOF (ie : Time Of Flight ou autrement dit Temps De Vol) pour générer la carte de profondeur. Cette technique se base sur la différence de temps entre l'émission d'un faisceau lumineux et son retour après réflexion sur un objet.
La distance est calculée comme suit :
$d = c.\frac{\Delta t}{2}$, avec c la vitesse de la lumière dans l'air.

Elle permet une bien meilleure précision même dans le noir que sa version précédente (`Kinect V1`).

1.5.2 Kinect V3

`Kinect V3`, baptisée `Microsoft Azure Kinect V3`, tout comme la précédente version, se base aussi sur la technologie TOF. Elle comprend un capteur RGB de 12 Mp, un capteur de depth de 1Mp avec un fov réglable en large ou réduit ainsi que 7 microphones intégrés.

Kinect V2 Vs Kinect V3

`Kinect V3` est beaucoup plus légère et plus petite que la `V2`. Elle possède entre autre plus de microphones que la précédente. Elle a été conçue afin d'être principalement utilisée avec Azure , le service cloud de `Microsoft`. C'est sans doute ce qui la démarque le plus de sa prédécesseure. En effet, ce service lui permet d'effectuer une partie des calculs. En outre, elle bénéficie des Cognitive Services, autrement dit de l'intelligence artificielle pourra être incluse dans les applications créées.

Chapitre 2

Notre outil pour la qualification de caméras RGB-D

Dans ce passage, nous allons décrire notre outil conçu pour qualifier les caméras par rapport à notre modèle-étalon montré dans l'image ci-dessous.

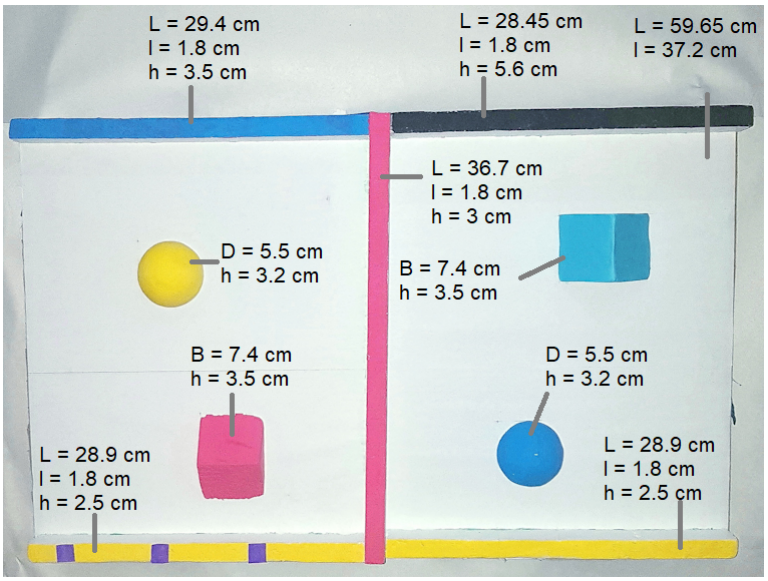


FIGURE 2.1 – Modèle-étalon.

2.1 Description de l'application

Notre application est composée d'une fenêtre qui comprend une partie affichage de la scène filmée en temps réel ainsi qu'une partie interface gra-

phique qui nous permet de communiquer avec le programme. Nous parlerons un peu plus en détail de ces parties dans les sous-sections à venir.

2.1.1 Première partie : Affichage de la scène captée

Pour cet effet, nous sommes partis d'un programme fourni par *M. Farès BELHADJ* auquel nous avons rajouté des choses pour la conception de l'outil. Il comprend l'utilisation de la bibliothèque OpenGL, GL4Dummies ainsi que la bibliothèque OpenNI2 pour établir la connexion avec la caméra RGB-D pour nous fournir par la suite la carte de couleurs et la carte de profondeurs.

2.1.2 Seconde partie : Interface graphique

Pour l'interface graphique, nous avons exploité la bibliothèque Dear ImGui disponible sur GitHub sur le lien suivant `https://github.com/ocornut/imgui`. C'est une bibliothèque assez facile d'utilisation une fois l'étape de l'intégration dans le contexte OpenGL effectué. Cette interface contient un certain nombre de composants. Parmi les composants principaux, nous pouvons citer une liste déroulante qui nous permet de sélectionner le modèle de caméra sur laquelle nous souhaitons réaliser les tests de mesures. Ainsi, une fois le choix de caméra effectué, le programme met automatiquement à jour tous les paramètres présents dans les champs suivants la liste en récupérant les données à partir d'un fichier que nous avons nommé *infoCameras.txt*. Ainsi, ce dernier pourra être exploité pour tout rajout de nouvelles caméras et/ou modification sur les données uniquement. Autres les composants cités ci-dessus, nous noterons la présence de deux cases à cocher comme montré dans la figure 2.2 La première permet une détection du modèle et l'autre de passer les tests de mesures.

Détection du modèle réel

Pour cette partie, une fonction *detectObject* est prévue à cet effet afin de renvoyer la localisation du modèle réel ainsi que celles des éléments le composant (demi-sphères, "pyramides", ...), ce qui fait un total de 10 objets à identifier. Une base de données d'images est créée pour servir de templates (images modèles) afin de pouvoir les distinguer.

Pour chacun d'entre eux, un appel à la fonction *callMatchTemplate* est effectué, qui, comme son nom l'indique, appelle la fonction *matchTemplate*[1].

1. Fonction OpenCV permettant de trouver à partir d'une image source les zones de correspondances avec l'image *template* (image modèle). La documentation sur cette fonction est disponible sur le lien suivant : `https://docs.opencv.org/2.4/doc/tutorials/`

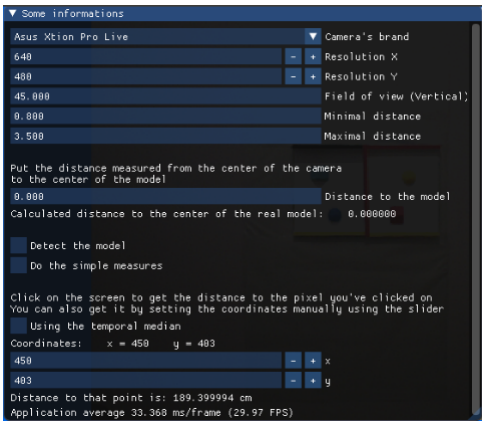


FIGURE 2.2 – Interface graphique du programme.

Dans cette dernière, nous transformons notre image source (image récupérée de la scène captée par la caméra RGB-D) ainsi que notre image template en niveaux de gris et ensuite nous les passons sous le filtre Canny afin de rehausser les contours pour une meilleure détection [2]. Ce processus qui comprend l'utilisation des filtres est appliqué uniquement pour la détection du modèle-étalon en entier et non sur ses composants. En effet, la couleur importe pour ces derniers puisque la forme n'est pas unique (présence de deux demi-sphères, ...).

Cette étape est assez conséquente en terme de temps. En effet, suivant la distance à laquelle est éloignée la caméra du modèle réel, le programme doit redimensionner la scène filmée avec une certaine échelle et avec un nombre précis de fois avant d'appeler la fonction *matchTemplate*, qui, combinée avec *minMaxLoc* [3], nous permet de renvoyer la localisation du modèle-étalon (coordonnées en x et y ainsi que la largeur et la hauteur de l'aire détectée). De ce fait, une fois les coordonnées récupérées, nous pouvons positionner notre modèle virtuel dont nous discuterons les détails de conception un peu plus loin dans ce mémoire. Ces coordonnées seront naturellement converties en coordonnées OpenGL. La formule employée à cet effet (permettant une conversion d'un système de coordonnées vers un autre) est comme suit :

`imgproc/histograms/template_matching/template_matching.html`

2. Nous nous sommes fortement inspirés du travail réalisé par Adrian ROSEBROCK, Ph.D en vision par ordinateur, disponible sur le lien suivant : `https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/`.

3. Fonction OpenCV qui renvoie les valeurs minimales et maximales d'une *Mat*. Cette dernière est une sorte de vecteur ou plus communément une matrice.

$$value_1 = \frac{value - min}{max - min}.(MAX - MIN) + MIN$$

tel que $value \in [min; max]$ (intervalle d'entrée) et $value_1 \in [MIN; MAX]$ (intervalle de sortie)

FIGURE 2.3 – Positionnement du modèle 3D sur le modèle réel.

Tests de mesures et rédaction de rapport

Pour celle-ci, nous récupérons les valeurs de la carte de profondeur d'OpenGL ainsi que les valeurs de profondeur de la caméra RGB-D. les valeurs de la depth OpenGL seront par la suite linéarisées [4] et ensuite normalisées [5]. Celles renvoyées par la caméra seront uniquement normalisées. Les valeurs des deux côtés étant normalisées, les tests de mesures peuvent débuter.

Pour mesurer le taux d'erreur entre les valeurs des deux profondeurs, nous avons opté pour trois méthodes distinctes ; RMSE, MAPE [6] et enfin Biais [7]. Une fois ces valeurs calculées pour chacun des éléments du modèle ainsi que le modèle lui-même, nous les sauvegardons dans un tableau, tableau qui sera plus tard utilisé pour le rapport.

Par la suite, nous nous servons de Gnuplot [8] afin de tracer les graphes montrant les valeurs de depth des deux côtés en faisant attention à ne pas écraser un éventuel graphe déjà présent.

4. Technique employée pour « approcher par une fonction linéaire ».
5. Standardiser les données afin qu'elles appartiennent à l'intervalle [0; 1].
6. Mean Absolute Percentage Error est une méthode permettant une mesure des erreurs tout comme le RMSE vu plus haut. Consulter cette page web pour un maximum d'informations : `https://www.statisticshowto.datasciencecentral.com/mean-absolute-percentage-error-mape/`
7. Tout comme les deux méthodes vu précédemment, elle permet un calcul du taux d'erreur. Le lecteur peut consulter ce lien la concernant `http://www.chups.jussieu.fr/polys/biostats/poly/POLY.Chp.10.html`
8. Un logiciel multiplateforme gratuit qui offre la possibilité de tracer des graphes en deux ou trois dimensions.

De plus, un rapport est rédigé en LaTeX en mettant en avant les informations relatives à la caméra (comme le champ de vision, la résolution, ...). A chaque fois qu'on coche sur la case *Do the simple measures* de l'interface graphique, le programme écrit ou réécrit sur un fichier que nous nommons *Report.tex* situé dans un dossier *Report* créé à cet effet. Si la caméra a déjà été testée, le programme rajoute dans la section concernée (dans le rapport) les nouveaux résultats incluant le graphe des valeurs de la profondeur, la distance calculée par le programme entre la caméra et le centre du modèle détecté ainsi qu'un tableau exposant les valeurs obtenues du taux d'erreurs pour les modèles et ses composants avec les trois méthodes citées plus haut. Et ce, si et seulement si la caméra n'a pas été d'ores et déjà testée avec cette distance calculée par le programme.

2.2 Conception du modèle réel et du modèle virtuel

2.2.1 Conception du modèle réel

Pour fabriquer le modèle réel, nous avons décidé d'opter pour le polystyrène pour les composants du modèle et du carton pour sa base. Cette dernière sera par la suite recouverte de papier blanc. Les composants seront peints de couleurs distinctes suivant leur hauteur (les éléments qui possèdent une même hauteur seront peints de la même couleur plus particulièrement les rebords du modèle) sauf les demi-sphères et "pyramides".

Nous avons opté pour le polystyrène car c'est un matériau facilement maniable et que l'on peut trouver assez facilement.

2.2.2 Conception du modèle virtuel

La réalisation du modèle virtuel s'est faite à l'aide du logiciel 123D Design qui est assez facile d'utilisation. Chaque élément du modèle est conçu en fonction des vraies mesures faites sur le modèle réel.

2.3 Comparaison de la depth OpenGL avec la depth de la caméra RGB-D

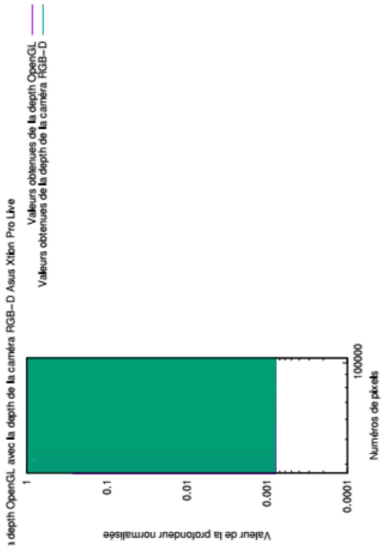
Dans ce passage, nous exposons certains des résultats obtenus ainsi que les résultats attendus avec la caméra Asus.

0.1 Asus Xtion Pro Live

Resolution X: 640
Resolution Y: 480
Field of view (vertical): 45°
Distance of use: between 0.8m and 3.5m

Distance between the camera and the center of the real model:

- real distance: not precised
- calculated distance: 0.959000m



1

object	RMSE	Bias	MAPE
Whole model	1.17644	1.51711	1.49895
Blue piece	0.286128	0.0931709	0.0860401
Black piece	0.238564	0.0649469	0.0621372
Vertical magenta piece	0.442908	0.281794	-0.281794
Yellow piece 1	0.200927	0.044062	0.0434231
Yellow piece 2	0.186558	0.0379655	0.0376244
Yellow half-sphere	0.090412	0.0107841	-0.0107841
Magenta pyramid	0.321306	0.120943	0.102204
Blue pyramid	0.179682	0.0370594	0.0348795
Blue half-sphere	0.278156	0.0913352	0.0725286

The estimated errors rate Distance between the camera and the center o
real model:

- real distance: 0.000000m
- calculated distance: 0.867000m

2

FIGURE 2.4 – Partie du rapport généré par le programme.

2.3.1 Résultats obtenus

Ces résultats sont extraits du rapport rédigé par le programme après réalisation des tests avec la caméra Asus Xtion Pro Live avec une distance de 1.936m (distance entre la caméra et le modèle). Nous avons créé la base de données d'images templates suivant cette distance.

object	**RMSE**	**Bias**	**MAPE**
Whole model	0.645805	0.464625	0.445257
Blue piece	0.163108	0.0310855	0.0294527
Black piece	0.140734	0.0231826	0.0215494
Vertical magenta piece	0.301483	0.130255	-0.130255
Yellow piece 1	0.132736	0.019909	0.018747
Yellow piece 2	0.122078	0.0168138	0.0156804
Yellow half-sphere	0.12617	0.0215568	-0.0215568
Magenta pyramid	0.12583	0.0232983	-0.0232983
Blue pyramid	0.136497	0.0254308	-0.0254308
Blue half-sphere	0.110225	0.0178525	-0.0178525

Ci-dessous le tableau résumant le taux d'erreur approximatif obtenu sur des distances différentes, et ce, en prenant en considération que la valeur RMSE calculée.

	Distances (m)							
object	**1.936**	**1.843**	**1.748**	**1.639**	**1.550**	**1.445**	**1.359**	**1.259**
Whole model	65%	69%	71%	76%	80%	84%	89%	94%
Blue piece	16%	17%	17%	18%	18%	21%	21%	22%
Black piece	14%	15%	15%	14%	17%	16%	18%	18%
Vertical magenta piece	30%	30%	30%	31%	31%	31%	32%	38%
Yellow piece 1	13%	13%	13%	13%	13%	13%	13%	13%
Yellow piece 2	12%	12%	12%	13%	13%	14%	12%	15%
Yellow half-sphere	13%	12%	12%	14%	15%	14%	13%	15%
Magenta pyramid	13%	14%	14%	14%	14%	15%	16%	16%
Blue pyramid	14%	14%	13%	13%	14%	16%	16%	17%
Blue half-sphere	11%	11%	12%	12%	13%	14%	15%	14%
Average rate	**20.1%**	**20.7%**	**20.9%**	**21.8%**	**22.8%**	**23.8%**	**24.5%**	**26.2%**

On peut observer aisément un taux d'erreur de plus en plus élevé pour le modèle en entier. Pour les autres éléments, le taux reste pratiquement constant.

2.3.2 Résultats attendus

Comme nous doutons de la méthode employée pour extraire les données OpenGL correspondantes aux données de la depth de la caméra (données extraites aux mêmes pixels), nous pensons qu'il est tout à fait normal que le taux soit si élevé. Toutefois, comme après chaque changement de distance (ie : la caméra se rapproche du modèle observé), le modèle 3D subit un changement d'échelle afin qu'il soit superposé au modèle réel. Ainsi, la valeur de la coordonnée z de chaque pixel du modèle 3D subit pareillement ce changement (devient de plus en plus grande).

Chapitre 3

Cas pratiques de qualification

Dans ce chapitre, nous montrons comment utiliser un modèle bien particulier de caméra avec notre application.

3.1 Asus Xtion Pro Live

Dans cette section, nous montrons comment nous avons fait pour configurer la caméra afin que le programme puisse effectuer les tests avec cette dernière. Tout d'abord, nous avons édité le fichier *infoCameras.txt* cité dans le chapitre précédent afin de pouvoir y ajouter les informations relatives à cette caméra.

Par la suite, nous n'avons pas eu besoin de rajouter des lignes de code pour la reconnaissance de la caméra (initialisation et récupération de la carte de couleurs et de la carte de profondeurs) car ce modèle de caméra est supporté par la bibliothèque OpenNI2.

Enfin, il nous suffira de sélectionner, à partir de l'interface de l'application, le modèle du capteur branché et cocher sur les cases *Detect the model* pour une détection du modèle-étalon et *Do the simple measures* pour démarrer les tests de mesures et rédiger le rapport comme expliqué précédemment.

Chapitre 4

Conclusion et Perspectives

Dans ce rapport, nous avons présenté notre outil pour la qualification de caméras RGB-D. Comme nous avons pu le constater, cet outil recense des erreurs au niveau des mesures. En effet, des données OpenGL qui ne sont pas forcément fiables, particulièrement lors de la mise à l'échelle, influent considérablement sur les résultats que nous avons vus lors du précédent chapitre.

En perspectives, nous souhaiterions vérifier de manière plus pointilleuse les données OpenGL extraites et améliorer la mise à l'échelle du modèle 3D de manière à ne pas impacter le taux d'erreur. Dans ce cas, la qualification pourra être effectuée dans des conditions permettant une meilleure comparaison entre les différents modèles de caméras.