Repérage dans l'espace

Afin de pouvoir proceder à l'integration d'un objet dans une scène, il est necessaire de pouvoir positionner le périphérique d'acquisition par rapport à la scène. Ces informations de positionnement sont necessaires non seulement au positionnement de l'objet virtuel mais également à la reconstruction de l'evironnement lumineux.

Partie 1.1

Hiérarchie de référentiels

Afin de mettre en place les différents mécanismes de repérage, il a été nécessaire de hiérarchiser les repères relatifs aux différents référentiels considérés.

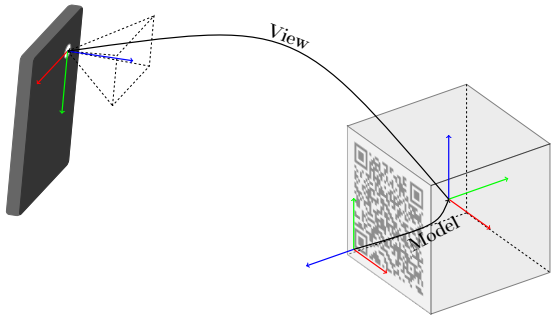
Le référentiel principal est le référentiel du monde. Il est par définition fixe au cours du temps et est défini par rapport à la mire utilisée pour l'acquisition. Ce repère est placé au centre géométrique de la mire et servira aussi bien à positionner l'objet à afficher qu'à définir le référentiel de base pour l'envmap.

Les différentes faces de la mire sont elles-mêmes fixes dans le repère du monde car liées à l'objet physique (la mire) qui le défini. Les transformations entre le repère de chacune des faces de la mire et le repère du monde sont codées dans les marqueurs présents sur les faces (matrice *model*).

Ainsi, ayant identifié une des faces à l'aide du marqueur présent sur cette dernière, il est possible, en appliquant la transformation codée par ce marqueur, de reconstruire le repère du monde.

L'interface d'acquisition est elle aussi positionnée vis-à-vis du repère principal. La transformation entre ces deux repères est codée par la matrice *view* et changera au cours du temps, l'utilisateur se déplaçant par rapport à la mire.

Figure 1.1 Les différents référentiels



Les différentes caméras présentes sur l'interface d'acquisition seront à leur tour positionnées relativement à l'interface d'acquisition. Cette dernière transformation est disponible dans les fichiers de configuration de caméra chargés au démarrage.

Table 1.1 Hiérarchie des matrices de transformations

	QRcode	
	⇅	
	Face	
$model^{-1}$ ↱		↳ $model$
	Monde	
$view^{-1}$ ↱		↳ $view$
	Smartphone	
$orientation^{-1}$ ↱		↳ $orientation$
	Caméra	
$cvToGl^{-1}$ ↱		↳ $cvToGl$
	Vue OpenGL	
$projection^{-1}$ ↱		↳ $projection$
	Image rendue	

Partie 1.2

Utilisation de marqueurs

L'évaluation des transformations, qui est nécessaire au positionnement dans la scène et donc à la reconstruction de l'environnement, nécessite la reconnaissance de marqueurs. L'identification de ces marqueurs, fixes dans l'espace du monde, associée à la connaissance de la matrice *model* correspondante permettent de calculer l'orientation au regard de la scène.

Afin de maximiser la reconnaissance de la scène dans sa globalité, il est nécessaire de construire une mire complète, disposant de plusieurs faces parmi lesquelles au moins une devra, pour toute orientation, être lisible. Les marqueurs constituants cette mire doivent par ailleurs être capables de porter les informations caractéristiques de la matrice *model* associée.

Il est donc nécessaire de choisir un modèle de marqueurs permettant à la fois une localisation précise d'au moins trois points (nécessaires au repérage) et portant des informations propres.

Parmi les nombreux modèles de marqueurs répondant

au pré-requis, le choix d'une implémentation s'est rapidement porté sur un modèle utilisant des QR codes [1]. L'adaptation à un autre modèle est particulièrement aisée puisqu'elle ne demande que le développement d'un module de scanner reconnaissant les marqueurs (voir section **??**, page **??**).

Sont disponibles en annexe les patrons des mires cubiques (figure **??**, page **??**) et hémisphériques octogonales (figure **??**, page **??**) construites à base de QR codes.

Partie 1.3

Évaluation des transformations

Compte tenu de la hiérarchie de référentiel décrite dans la section précédente, il est nécessaire, pour décrire complètement le système, de décrire chacune des matrices de transformation.

1.3.1 Calibrage de la caméra

L'étape de calibrage de caméra permet de calculer les différents termes de la matrice *projection* propre à l'interface d'acquisition. Ici l'interface est simplement modélisée suivant le modèle communément admis de sténopé (ce qui revient à assimiler l'objectif à un simple point), dont la calibration détermine les caractéristique physiques (longueur focale, définition du capteur).

La matrice *projection* est de la forme suivante :

$$\begin{pmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.3.1.1}$$

avec

$$\alpha_x = f * m_x$$
$$\alpha_y = f * m_y$$

où f est la longueur focale, m est la résolution du capteur selon les axes $\vec{x}$ et $\vec{y}$, u_0 et v_0 sont les coordonnées du centre optique sur le capteur et γ est un coefficient qui décrit la non orthogonalité des axe principaux.

Le calcul de cette matrice est résolu par la minimisation d'un système linéaire sur de nombreuses prises de vue du marqueur.

Cette étape de calibration est implémentée par la bibliothèque OpenCV [2].

Une fois déterminés, les paramètres relatifs à la caméra sont sauvegardés dans un fichier `xml`.

1.3.2 Positionnement à partir de marqueurs

Une fois la caméra calibrée, c'est-à-dire une fois les paramètres intrinsèques déterminés il est possible d'utiliser la connaissance de la matrice *projection* et des dimensions du marqueur pour reconstruire, à partir des coordonnées du marqueur dans l'espace image, la matrice de transformation *extrinsic* correspondant au passage de l'espace du QR code (face de la mire) à l'espace de la caméra.

Les matrices *model* et *orientation* étant connues, la première grâce aux informations écrites sur le QR code et la seconde étant pré-enregistrée (fixe, liée à la géométrie du téléphone), il est possible d'en déduire la matrice *view* qui caractérise le positionnement dans la scène :

$$\begin{aligned} view = {} & orientation^{-1} \\ & \times extrinsic \\ & \times model^{-1} \end{aligned} \tag{1.3.2.1}$$

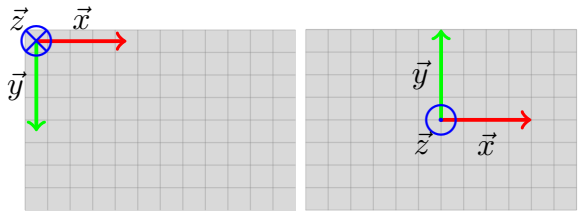
1.3.3 Models OpenCV / OpenGL

Les différentes bibliothèques ont chacune leurs approches qui inpliquent des différences de conventions.

OpenCV considère les images comme des matrices. À ce titre l'origine est placée en haut à gauche, avec le $\vec{x}$ pointant vers la gauche et le vecteur $\vec{y}$ pointant vers le bas. Ainsi regarder une image revient à regarder selon l'axe des $\vec{z}$ croissants.

À l'inverse OpenGL s'intéresse à des espaces $3D$ complets et oriente donc naturellement le vecteur $\vec{y}$ vers le haut. L'utilisateur regarde donc selon l'axe des $\vec{z}$ décroissants.

Figure 1.2 Conventions d'orientations



(a) OpenCV **(b)** OpenGL

Afin d'utiliser conjointement les deux bibliothèques, il convient de faire les transformations adéquates.

$$cvToGl = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{1.3.3.1}$$

1. **QRCode :** http://www.qrcode.com/
2. **OpenCV :** http://opencv.org/

Bibliographie