Sujet de projet tuteuré : création d'images avec réalité augmentée

Photo originale de l'IUT

Photo augmentée

Contents

1 Sujet

- Utilisation de la bibliothèque PoLAR (polar.inria.fr)
- Utilisation basique de C++
- Taches :
  - T1 : Compiler la bibliothèque PoLAR (après Qt5.4 et OpenScengraph 3.2)
  - T2 : Charger une photo (ex : Extérieur de l'IUT)
  - T3 : Mettre un (ou plusieurs) objet 3D sur la photo avec la bonne matrice de protection (ex : une voiture)
  - T4 : Mettre un (ou plusieurs) objet fantôme pour avoir de l'ombrage (ex : le sol)
  - T5 : Mettre un (ou plusieurs) objet fantôme pour avoir de l'occlusion partielle (ex : un mur devant la voiture)
  - T6 : Ajouter des comportements dynamiques continus (ex : une balle qui rebondit de gauche à droite)
  - T7 : Ajouter des comportements évènementiels (ex : un avion qui passe lorsque une touche est appuyée)
  - T8 : Faire un gif animé montrant les différentes étapes de création de la scène
- Rendu : un rapport sur les différentes étapes de création de la scène
- Bonus : Faire la même chose avec une séquence d'images avec différents points de vue

- Difficulté mathématique : Calcul de la matrice de projection basée sur la caméra
- Difficulté informatique : installation de bibliothèques, programmation objet dans un autre language (C++), programmation événementiel

2 Aide

Il y pas mal d'aide dans le manuel disponible sur https://polar.inria.fr/files/2016/02/PoLAR_User_Manual.pdf

Pour chaque tâche :

- Pour T1, l'explication pas à pas de l'installation est p.2 pour Windows, p.1 pour Linux et p.4 pour Mac OS.
- Pour T2, le chargement d'images est expliqué p. 7.
- Pour T3, l'insertion d'objets 3D est p. 14. Le calcul de la matrice de projection est expliqué dans §4.
- Pour T4, l'insertion d'objets fantômes est p. 19.
- Pour T5, c'est la même procédure.
- Il y aura un peu de programmation pour T6, du style une boucle 'for'.
- Pour T7, c'est un peu la tâche bonus.
- T8 correspond à prendre un screenshot à chaque étape.

3 Évaluation

- Rien fait : 0 pts
- Installation de la bibliothèque et exécution des exemples : 5 pts * sous réserve d'une démo à l'enseignant
- Incrustation d'objets 3D : 5 pts * code envoyé à l'enseignant * matrice de projection correcte
- Ombrage sur objet(s) fantôme(s) : 2 pts * code envoyé à l'enseignant * ombres cohérentes
- Parties cachées : 2 pts * code envoyé à l'enseignant * parties cachées cohérentes

—— plus difficile ——

- Ajouter des comportements dynamiques continus (ex : une balle qui rebondit de gauche à droite) : 2 pts
- Ajouter des comportements événementiels (ex : un avion qui passe lorsque une touche est appuyée) : 2 pts

Autres rendus :

- Un gif animé montrant les différentes étapes de création de la scène (2pts)
- Rendu : un rapport sur les différentes étapes de création de la scène (2pts)

Le total fait 22 pts

4 Calcul d'une matrice de projection

Pour plus de renseignement, ce référer à https://www.robots.ox.ac.uk/~vgg/publications/2000/Simon00/simon00.pdf.

L'objectif est de connaitre la matrice permettant de passer du repère image au repère réel et inversement (Cf figure 1).

4.1 mesure des points sur l'image

Les points **Q1**, **Q2**, **Q3** et **Q4** peuvent être mesurés sur l'image grâce à l'utilitaire de PoLAR `bin/calib`.

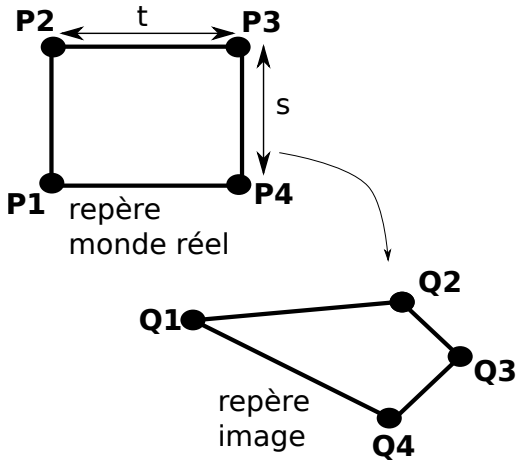
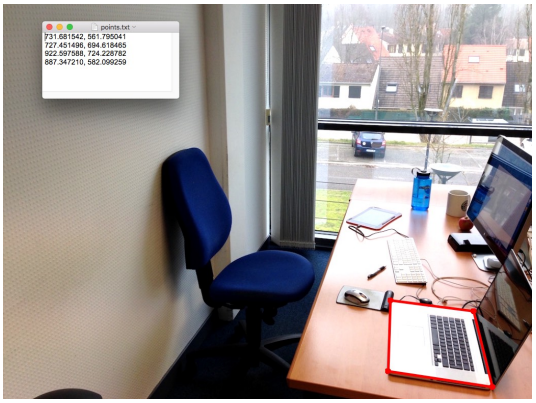


Figure 1: Passage d'un repère image à un repère dans le monde réel



Utilisation :

- 'p' pour activer la sélection des quatre coins du rectangles.
- Bouton du milieu pour ajouter des points, commencer en bas à gauche et aller dans le sens des aiguilles d'une montre.
- 'k' pour avoir une couleur plus visible.
- ctrl (ou cmd sur Mac) + souris pour pan et zoom.
- Bouton de gauche de la souris pour déplacer des points.
- 's' pour enregistrer dans 'points.txt' à la racine de PoLAR.

Figure 2: Dessin des points sur l'image avec `bin/calib cheminVersImage.png`

4.2 Calcul de la distance focale

Si la taille de l'image est $h \times w$ (pour *height* et *width*), le centre de l'image est de coordonnée : $(h/2, w/2)$.

Les points doivent alors être centrés :

$$Q1' = \begin{bmatrix} Q1_x - w/2 \\ Q1_y - h2 \\ 1 \end{bmatrix}; Q2' = \begin{bmatrix} Q2_x - w/2 \\ Q2_y - h2 \\ 1 \end{bmatrix}; Q3' = \begin{bmatrix} Q3_x - w/2 \\ Q3_y - h2 \\ 1 \end{bmatrix}; Q4' = \begin{bmatrix} Q4_x - w/2 \\ Q4_y - h2 \\ 1 \end{bmatrix} \quad (1)$$

Pour calculer la distance focale, il faut connaitre les points de fuite sur l'image ($\mathbf{V}$, $\mathbf{W}$). Ils peuvent être calculés comme illustrés sur la figure 3.

Dans la suite $\otimes$ désigne le produit vectoriel.

Les lignes de fuites $l1$, $l2$, $l3$ et $l4$ sont données par:

$$l1 = \begin{bmatrix} Q1'_x \\ Q1'_y \\ 1 \end{bmatrix} \otimes \begin{bmatrix} Q2'_x \\ Q2'_y \\ 1 \end{bmatrix}; l2 = \begin{bmatrix} Q4'_x \\ Q4'_y \\ 1 \end{bmatrix} \otimes \begin{bmatrix} Q3'_x \\ Q3'_y \\ 1 \end{bmatrix}; l3 = \begin{bmatrix} Q2'_x \\ Q2'_y \\ 1 \end{bmatrix} \otimes \begin{bmatrix} Q3'_x \\ Q3'_y \\ 1 \end{bmatrix}; l4 = \begin{bmatrix} Q1'_x \\ Q1'_y \\ 1 \end{bmatrix} \otimes \begin{bmatrix} Q4'_x \\ Q4'_y \\ 1 \end{bmatrix} \quad (2)$$

A faire : 4 produits vectoriels

Les intersections V et W sont données par:

$$V^H = l1 \otimes l2; W^H = l3 \otimes l4; V = \begin{bmatrix} V_x^H / V_z^H \\ V_y^H / V_z^H \\ 1 \end{bmatrix}; W = \begin{bmatrix} W_x^H / W_z^H \\ W_y^H / W_z^H \\ 1 \end{bmatrix} \quad (3)$$

A faire : 2 produits vectoriels + 4 opérations

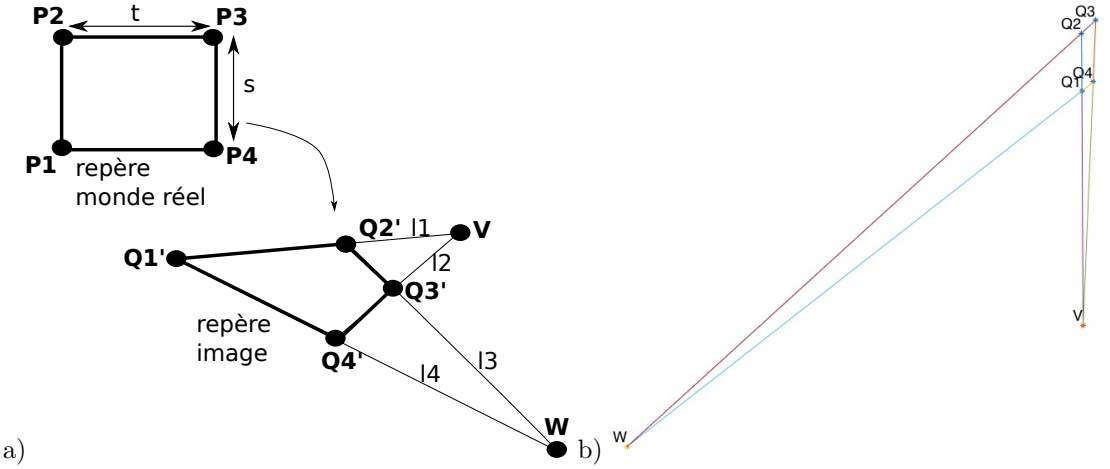


Figure 3: Point de fuite d'un rectangle sélectionné. a) théorie b) exemple

La distance focale f est alors donnée par :

$$f = \sqrt{-(V_x.W_x + V_y.W_y)} \tag{4}$$

A faire : 1 racine carrée d'une opération
Si $(V_x.W_x + V_y.W_y) > 0$ alors il y a une impossibilité physique, reprendre alors les points sur l'image.
La distance focale permet de définir la **matrice intrinsèque K** qui est indépendante du point de vue.

$$K = \begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

avec l'hypothèse d'avoir des pixels homogènes et d'avoir le distance focale appliquée au centre de l'image.

On peut alors vérifier que les points **V** et **W** sont bien aux intersections des droites **(Q1'Q2')** et **(Q3'Q4')** pour **V** et **(Q2'Q3')** et **(Q1'Q4')** pour **W**. La figure 3.b) donne un exemple de validation.

4.3 Calcul de la matrice d'homographie

P1 de la figure 1 est considéré comme l'origine du repère monde.

P1, **P2**, **P3** et **P4** doivent être dans le plan $z = 0$.

Les coordonnée des points dans le monde réel deviennent alors :

$$P1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}; P2 = \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix}; P3 = \begin{bmatrix} s \\ t \\ 0 \end{bmatrix}; P4 = \begin{bmatrix} 0 \\ t \\ 0 \end{bmatrix} \tag{6}$$

les coordonnées mesurées sur l'image Qi recentrées Qi' sont :

$$Qi' = \begin{bmatrix} Qi_x - w/2 \\ Qi_y - h/2 \\ 0 \end{bmatrix} \tag{7}$$

Soit la matrice **A** et **Qi'** définies par :

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & -Q2'_x & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & -Q2'_y & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -Q3'_x & -Q3'_x \\ 0 & 0 & 0 & 1 & 1 & 1 & -Q3'_y & -Q3'_y \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & -Q4'_x \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & -Q4'_y \end{bmatrix}; Qi' = \begin{bmatrix} Q1'_x \\ Q1'_y \\ Q2'_x \\ Q2'_y \\ Q3'_x \\ Q3'_y \\ Q4'_x \\ Q4'_y \end{bmatrix} \tag{8}$$

La matrice d'homographie h est alors donnée par $h = A^{-1}.Qi'$ avec le changement de forme suivant :

$$\begin{bmatrix} h11 \\ h12 \\ h13 \\ h21 \\ h22 \\ h23 \\ h31 \\ h32 \end{bmatrix} => \begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & 1 \end{bmatrix} \tag{9}$$

A faire : Opérations avec des matrices

4.4 Calcul de la matrice de projection

La matrice de projection est une composition de la matrice intrinsèque **K** précédemment calculée avec la matrice extrinsèque **E** qui comprend les transformations de rotations (**R1**, **R2** et **R3**) et de translation (**T**).

Calculer d'abors la matrice B:

$$B = \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} .h = \begin{bmatrix} \vdots & \vdots & \vdots \\ B1 & B2 & B3 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{10}$$

La matrice de projection est **M=K.E** avec :

$$E = \begin{bmatrix} \vdots & \vdots & \vdots & \\ R1 & R2 & R3 & T \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{11}$$

avec les valeurs suivantes :

$$\begin{array}{lcl} R1 & = & B1/\|B1\| \\ R3 & = & R1 \otimes B2/(\|R1 \otimes B2\|) \\ R2 & = & R3 \otimes R1 \\ \lambda & = & \|B1\|/s \\ t_{calcule} & = & \|B2\|/\lambda \\ T & = & B3/\lambda \end{array} \tag{12}$$

A faire : Opérations avec des matrices

On peut alors vérifier que $t_{calcule}$ est suffisement proche du t mesuré.

Pour vérifier la projection, on peut utiliser l'utilitaire `bin/svisu3` (Cf Figure 4).

Figure 4: Vérification de laprojection avec `bin/svisu3 cheminVersImage.png cheminVersProjection.txt cheminVersObjet3D.obj`

5 Création d'un nouveau projet utilisant PoLAR

S'inspirer de l'exemple `runPolar` disponible dans `polar/Applications` du répertoire GIT.
Modifier particulièrement le fichier `CMakeLists.txt` :

- remplacer 'runPolar' par le nom de votre projet
- enlever les appels à OpenCV:

```
find_package(OpenCV REQUIRED core imgproc highgui calib3d xfeatures2d)

set(OPEN_CV_LIBS
  ${OpenCV_LIBS}
  #opencv_core
  #opencv_features2d
  #opencv_highgui
  #opencv_calib3d
  #opencv_xfeatures2d
  #opencv_imgproc
  #opencv_imgcodecs
  #stdc++

include_directories(${OpenCV_INCLUDE_DIRS})}
```

- lister vos propres fichiers sources à la place de ceux de runPoLAR :

```
set(SRCS
  LoadImageDialog.cpp
  LoadImageDialog.h
  PoseViewer.cpp
  PoseViewer.h
  Tracker.cpp
  Tracker.h
  VideoViewer.cpp
  VideoViewer.h
  Interface.cpp
  Interface.h
)
```

- créer la variable d'environnement `PoLAR_SOURCE_DIR` :

```
export PoLAR_SOURCE_DIR=/Users/chemin_qui_va_bien/polar
```

- Utiliser l'application CMake pour construire un projet pour votre application :

```
mkdir build && cd build && cmake ..
```

6 Propriété intellectuelle

- Uniquement utiliser une photo originale (qui ne vient pas, par exemple, d'internet). La photo devra avoir un objet rectangulaire **le plus grand possible** (pour le calcul de la matrice de projection) ainsi que des **plans orthogonaux** (pour faciliter le placement d'objets virtuels).
- Uniquement utiliser des modèles 3D faits à partir de Blender ou étant sous licence 'Creative Commons' (ex: http://resources.blogscopia.com/category/models/

Le rapport devra mentionner où a été prise la photo ainsi que d'où viennent les modèles 3D.