POLYTECH ANGERS

# Estimation automatique de la matrice extrinsèque : recalage automatique & calcul de distances. (2 séances)

L'objectif de cette séance est d'estimer automatiquement la matrice extrinsèque. Dans un second temps, on pourra utiliser cette matrice pour recaler un objet virtuel (application typique de réalité augmentée) et également pour mesurer la distance "en mm" entre le damier et la caméra.

## 1 Présentation générale

Afin d'estimer la matrice extrinsèque pour chacune des poses fournies, nous allons utiliser la fonction opencv `solvePnP` (voir documentation en ligne).

La méthode `solvePnP` estime la matrice extrinsèque en s'appuyant sur les connaissances suivantes :

i) Les coordonnées 3D (`pts_3D_ref`) des coins des cases du damier dans l'image de référence sont connues.

ii) Les coordonnées pixels2D de coins des cases du damier dans l'image courante sont obtenues via la fonction (`findChessboardCorners`).

iii) La matrice intrinsèque $K$ est connue.

Ces connaissances seront les entrées de la fonction `solvePnP`. De plus, on peut faire les remarques suivantes :

- la fonction `solvePnP` requiert les distorsions :on utilisera `dist=np.zeros(4)` afin de supposer qu'aucune distorsion n'est présente.
- la fonction `solvePnP` retourne les angles de rotation (selon les axes X, Y et Z) que l'on pourra afficher (`print`) pour comparer avec l'estimation manuelle. Attention à la conversion degré en radians.
- la fonction `solvePnP` retourne les angles de rotation que l'on pourra exprimer sous forme matricielle (matrice 3x3) à l'aide de la fonction `opencv Rodrigues : R,b=cv2.Rodrigues (rvec)`. En ajoutant le vecteur de translation (retourné par `solvePnP`) à cette matrice 3x3, on obtiendra la matrice extrinsèque finale de taille 3x4 (la colonne de translation : `M_ext=np.hstack((R,tvec))`

## 2 Exercices

**Exercice 1**

1. Intégrez, dans le fichier `helper.py`, une fonction nommée `compute_extrinsic_matrix( pts_3d_ref,pts2d,K)`, capable d'estimer la matrice extrinsèque.

*Remarque :* Une fois cette fonction ajoutée, votre code principal aura la structure suivante :

```
#Chargement de l'image de référence
image_reference=cv2.imread('reference_image.png')
retval,corners=cv2.findChessboardCorners(image_reference, ... )
#Coordonnees 3D des coins du damier dans l'image de reference
corners_3d=helper.pixels2camera(corners, ... )
...
#Image courante
image_current = cv2.imread('pose_2.png')
retval,corners2d_current=cv2.findChessboardCorners(image_current,...  )
#Matrice extrinseque
M_ext=helper.compute_extrinsic_matrix(pts_3d_ref, pts_2d_cur, K, True)
#Matrice de projection
P=np.dot(K,M_ext)
...
```

2. Appliquez cette technique de recalage à l'ensemble des photos de la vidéo `video_damier.m4v`.

**Exercice 2**

L'objectif de cet exercice est de calculer la distance qui sépare le damier du plan $(C, X, Y)$. L'idée est la suivante :

- Le barycentre $P$ du damier est connu dans l'image de référence.
- Pour une pose donnée, on peut, grâce à la matrice extrinsèque, connaître les coordonnées de ce point $P$ dans le repère caméra.
- Finalement, sa coordonnée $Z$ donnera la distance recherchée.

1. Développez un programme Python pour cette fonctionnalité, en exploitant les programmes précédemment écrits.
   *Indication :* prenez garde à exprimer le barycentre du damier en coordonnées homogènes, pour pouvoir lui appliquer la matrice extrinsèque.
   Le résultat attendu est illustré par la figure 1.
2. Appliquez enfin ceci à chaque image de la vidéo `video_damier.m4v`.

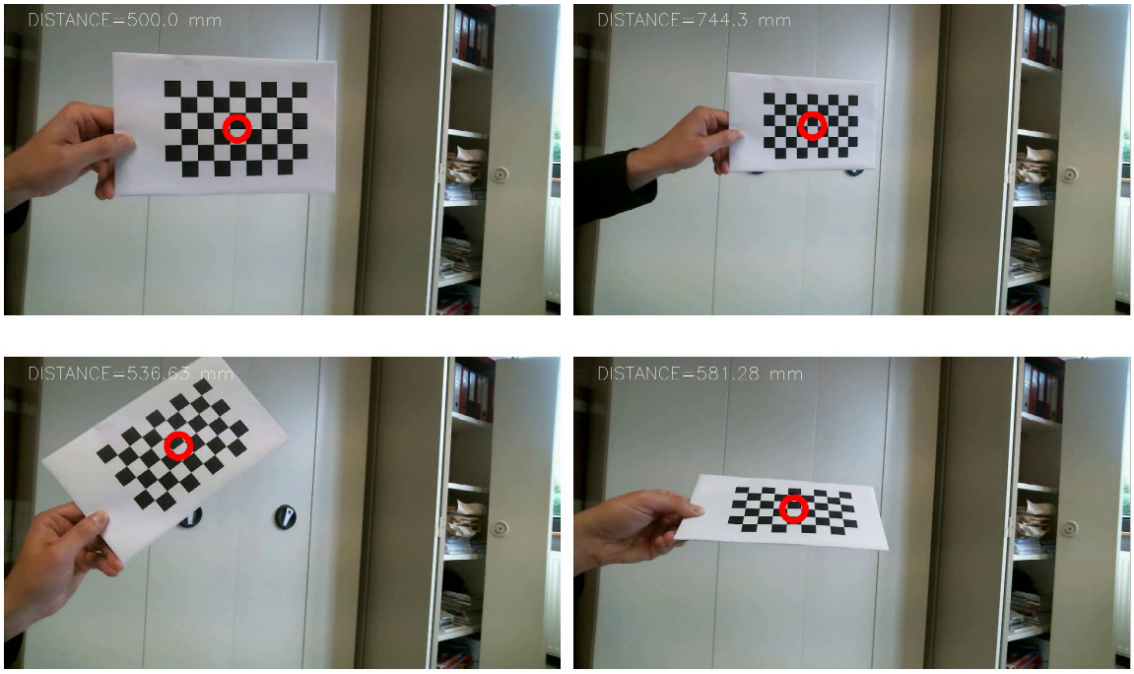


FIGURE 1 – Calcul automatique de distances.