SY32 – TD Vision 04 : Calibrage/étalonnage des distorsions géométriques

Exercice 1 : Procédure de calibrage

La procédure la plus classique de calibrage caméra consiste à prendre une série de photos d'un objet de dimensions et caractéristiques connues : une mire. La mire la plus commune présente un motif de damier, comme en Figure 1.

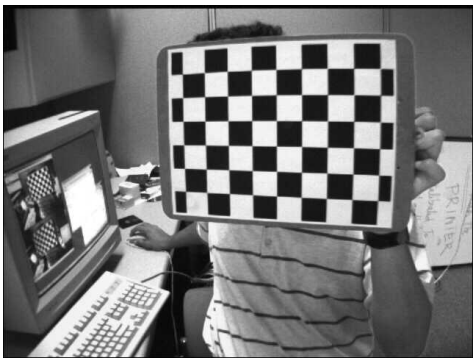

FIGURE 1 – Exemple d'image de mire.

Il s'agit de détecter dans les images les coins de la mire (donc dans le repère image), pour lesquels les positions réelles sont connues par rapport au repère de la mire. La relation entre les points réels et les points dans l'image permet de procéder aux calculs des paramètres de projection. Pour calibrer correctement les distorsions, il faut exploiter une série d'images de sorte à couvrir toutes les zones de l'image avec la mire. Par exemple, le centre de l'image présente généralement moins de distorsions que les bords, et se limiter à optimiser les paramètres sur des points au centre ne permet pas de corriger correctement les distorsions.
Un code fonctionnel et accompagné d'images de mire est fourni pour pratiquer la procédure classique de calibrage (inspiré des guides OpenCV : `https://docs.opencv.org/4.3.0/dc/dbb/tutorial_py_calibration.html`). Le but de l'exercice est de le compléter pour illustrer et analyser chaque étape. Les parties à insérer devraient l'être là où un commentaire « `# A FAIRE` » ou « `# A COMPLETER` » apparaît dans le code. Il faut s'inspirer des différentes fonctions utilisées. Note : ce TD utilise largement les fonctions d'OpenCV, une librairie de traitement d'images très populaire et bien optimisée, écrite au départ en C++.

1. Exécuter le code pour s'assurer de ne pas avoir d'erreur Python (librairies absentes, dossier des images non trouvé, etc). Le calibrage devrait fonctionner, enregistrer sur le disque une image corrigée « calibresult.png », et donner une matrice K proche de :

$$K = \begin{pmatrix} 534,11 & 0 & 341,44 \\ 0 & 534,35 & 232,08 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Afficher les coins détectés dessinés sur les images. Ils devraient correspondre aux intersections entre les cases noires et blanches.

3. Dessiner une image sur fond noir du cumul des coins détectés dans toutes les images. Commenter. Une illustration du résultat attendu est donnée en Figure 2.

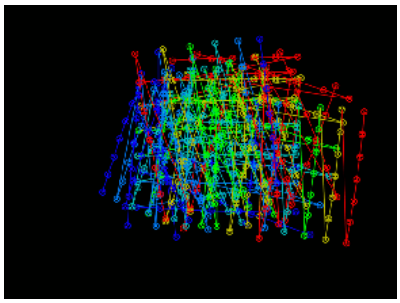

FIGURE 2 – Image de cumul des positions des points détectés.

4. Afficher une vue 3D des points des mires à chaque position par rapport à la caméra. Pour cela utiliser les estimations `rvecs` et `tvecs`, qui sont des listes contenant le vecteur de rotation (selon la notation de Rodrigues) et le vecteur de translation de chaque mire pour les images où la détection des points a été correcte. `rvecs` et `tvecs` permettent de passer du repère de la mire au repère de la caméra. Pour plus de détails, chercher dans la documentation d'OpenCV la fonction `calibrateCamera`. Pour cette question, il faudra employer la fonction OpenCV `cv.Rodrigues`, qui permet de convertir un vecteur de rotation de Rodrigues en matrice de rotation.
Une illustration du résultat attendu est donnée en Figure 3.

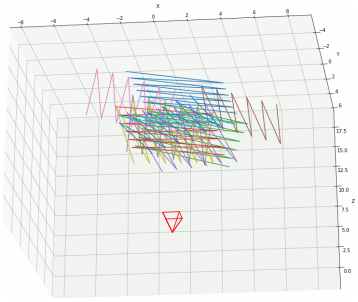

FIGURE 3 – Image des mires en 3D par rapport au repère caméra.

5. Afficher toutes les images rectifiées (en incluant les bordures noires ; c'est-à-dire sans découper la région centrale).
6. Commenter les paramètres de la caméra calibrée, l'affichage est déjà écrit dans le code.
Avec OpenCV, la convention pour stocker les paramètres de distorsions géométriques est d'enregistrer une liste de valeurs dans cet ordre $(k_1, k_2, p_1, p_2, k_3)$, permettant d'une part de représenter les distorsions radiales par un modèle polynomial :

$$\begin{cases} x_{distrad} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{distrad} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases}$$

où (x, y) sont les positions du point rectifié par rapport au centre optique, r la distance de ce point au centre optique, et $(x_{distrad}, y_{distrad})$ sont les positions du point avec les distorsions radiales.
D'autre part, ce modèle permet de représenter les distorsions tangentielles, par l'équation :

$$\begin{cases} x_{disttan} = x_{distrad} + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{disttan} = y_{distrad} + [2p_2 xy + p_1(r^2 + 2y^2)] \end{cases}$$

avec $(x_{disttan}, y_{disttan})$ les positions du point avec les distorsions tangentielles.
La projection pespective de la caméra (modèle sténopé) est donnée par :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

(Source : `https://docs.opencv.org/4.5.0/d4/d94/tutorial_camera_calibration.html` et `https://docs.opencv.org/4.5.0/d9/d0c/group__calib3d.html#ga7dfb72c9cf9780a347fbe3d1c47e5d5a`)

7. A FINIR ! Le décalage des points causé par les distorsions radiales doit être monotone. Afficher la courbe du modèle de distorsion estimé en fonction du rayon, en abscisse les positions sans distorsion, en ordonnée les positions avec distorsion, en fonction du rayon par rapport au centre de projection. atention focale normalisée
8. Dessiner le champ de vecteurs des distorsions radiales, c'est-à-dire les vecteurs représentant le décalage de la position idéale sans distorsions à la position avec distorsions radiales (sans rectification). La fonction `cv.initUndistortRectifyMap` permet de calculer les coordonnées des points des images après application des distorsions. On affichera un point sur 16 selon X et selon Y. Y inclure aussi la projection du centre optique. Commenter. La Figure 4 illustre le résultat attendu.

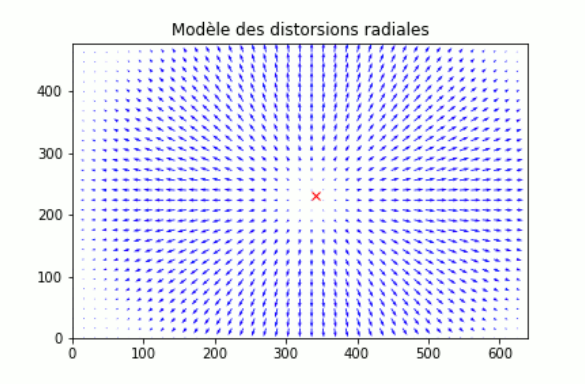

FIGURE 4 – Champ des vecteurs des distorsions radiales de la caméra calibrée.

9. Reporter l'erreur de reprojection 2D (en pixels).
10. Pour chaque image, afficher les points détectés et la reprojection des points 3D de la mire dans l'image. Pour cela on peut utiliser la fonction `cv.circle` avec des paramètres différents. L'erreur de reprojection est en fait la mesure moyenne de leurs écarts de position.
11. Comparer le résultat du calibrage en utilisant uniquement les images 01, 04 et 07, avec le calibrage obtenu en utilisant toutes les images. Que dire de l'erreur de reprojection et de l'aspect visuel des images rectifiées ?

bonus Expérimenter et calibrer une webcam.

Liste de fonctions utiles :

- `cv2.drawChessboardCorners`
- `matplotlib.pyplot.subplots`
- `range`
- `len`
- `enumerate`
- `numpy.transpose` (ou `*matriceNumPy*.T`)
- `numpy.matmul` (ou opérateur `@`)
- `numpy.meshgrid`
- `numpy.flatten`
- `numpy.shape`
- `matplotlib.pyplot.quiver`