PIDR - Recherches sur le traitement d'images

Aurélie Demure

Mars 2024

1 Présentation des recherches sur OpenCV et son utilisation

Différentes commandes nécessaires :

- Lire une image existante : **image = cv2.imread('name.png')**
- Sauvegarder une image : **cv2.imwrite('name.png',gray)**
- Mettre la fenêtre en attente (pour calibrer la vitesse d'enchaînement des images) : **cv2.waitKey(0)**
- Fermer et libérer la mémoire : **cv2.destroyAllWindows()**

Possibilités :

- histogramme
- conversion de couleurs
- seuillage
- binarisation
- remplissage des trous pour une meilleure binarisation
- dessin des contours
- aire
- perimètre
- circularité
- ratio
- étendue
- solidité
- texture

2 Recherche théorique sur la calibration d'une image et le traitement de la distorsion

2.1 Modélisation d'une caméra

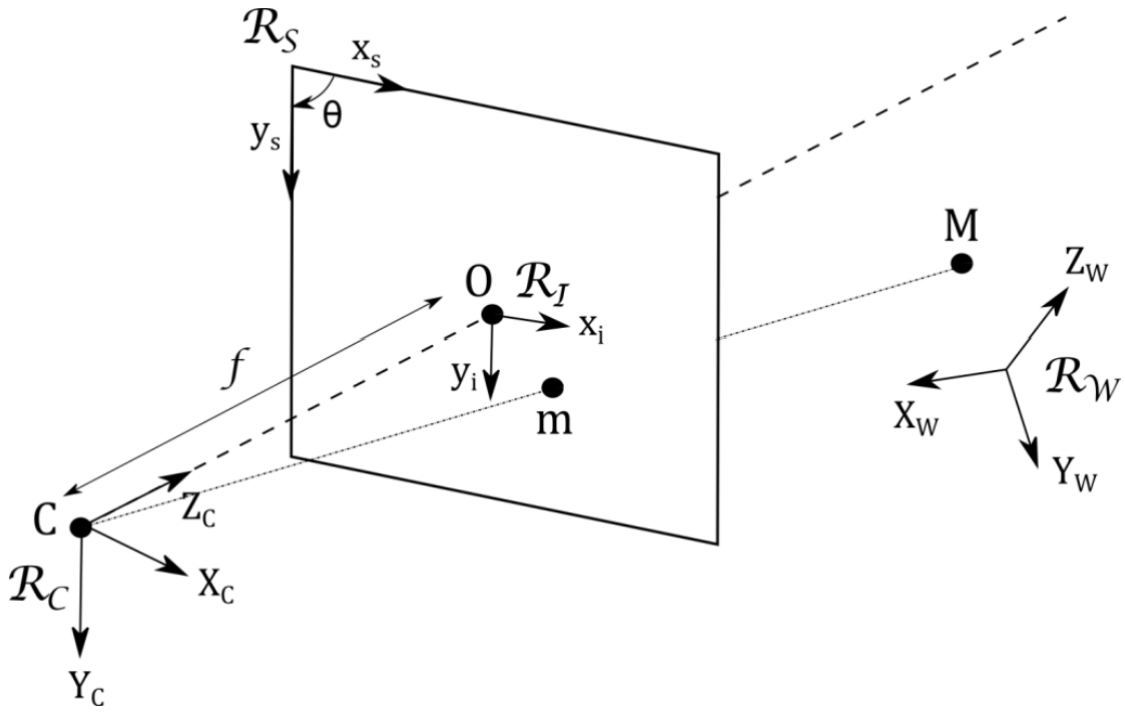

Figure 1: Modélisation d'une caméra

Légende :

- repère Rc : repère de la caméra avec (Xc, Yc, Zc)
- repère Rw : repère monde "world" qui représente la 3D avec (Xw, Yw, Zw)
- repère Ri : repère de l'image qui représente la 2D avec (xi, yi)
- repère Rs : repère du capteur "skew" avec (xs, ys, theta)

2.2 Explication de l'origine de la matrice de projection de la caméra

La matrice de projection de la caméra est obtenue grâce à trois transformations.

2.2.1 Première transformation

Si nous reprenons la figure, le point M est défini en 3D par le repère "monde" Rw avec ses coordonnées (Xw, Yw, Zw). Cette même position est exprimée dans le repère local Rc de la caméra C par les coordonnées locales (Xc, Yc, Zc). La première transformation est donc ce changement de repère qui donne une matrice T que nous pouvons décomposer en une matrice de rotation R et un vecteur de translation t.

L'obtention de la matrice se fait par le calcul suivant :

$$\begin{Bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{Bmatrix} = [T] \begin{Bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{Bmatrix} = \begin{bmatrix} [R] & \{t\} \\ \{0_{1\times 3}\} & 1 \end{bmatrix} \begin{Bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{Bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{Bmatrix}$$

Figure 2: Calcul pour la matrice de la première transformation : W ->d C

Il y a donc six paramètres : 3 angles et 3 translations qui sont les **paramètres extrinsèques** permettant de définir le positionnement de la caméra dans l'espace 3D.

2.2.2 Deuxième transformation

La deuxième transformation consiste à projeter le point 3D M sur le plan image associé au repère Ri. Nous obtenons par cette projection un point 2D m de coordonnées (xi, yi).

Calcul de changement de repère :

$$s \cdot \begin{Bmatrix} x_i \\ y_i \\ 1 \end{Bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{Bmatrix}$$

Figure 3: Calcul de la projection de W -> I

Le seul paramètre intervenant ici est la **focale f** de la caméra.

2.2.3 Troisième transformation

Enfin, il faut exprimer les coordonnées projetées, donc celles de m, dans le repère du capteur Rs. Pour cela il est impératif de connaître :

- **l'angle de décadrage** entre les axes horizontaux et verticaux du capteur (on le supposera souvent égal à 90°)
- la position du **centre optique**
- la **taille physique du pixel** dans les deux directions

Les deux transformations précédentes nous avaient donné une matrice dde projection que l'on nommera K :

$$[K] = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 4: Matrice K obtenue par composition des deux premières transformations

Cette matrice possède quatre **paramètres intrinsèques** : les longueurs focales horizontale et verticale (fx, fy) en pixels et la position du centre optique dans l'image (cx,cy) en pixels également.

La matrice de projection complète M est composée des paramètres extrinsèques obtenus par T et des paramètres intrinsèques obtenus par K :

$$s \cdot \{x_s\} = [K][T]\{X_W\} = [M]\{X_W\}$$

Figure 5: Calcul de la matrice finale

Après calculs, la matrice de projection M de la caméra est donc la suivante :

$$[M] = \begin{bmatrix} r_{11}f_x + r_{31}c_x & r_{12}f_x + r_{32}c_x & r_{13}f_x + r_{33}c_x & t_x f_x + t_z c_x \\ r_{21}f_y + r_{31}c_y & r_{22}f_y + r_{32}c_y & r_{23}f_y + r_{33}c_y & t_y f_y + t_z c_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

Figure 6: Matrice M explicite

C'est donc cette matrice que nous chercherons à obtenir avec différentes librairies et fonctions afin d'étalonner notre caméra.

2.3 Etude des trois principales méthodes d'étalonnage

2.3.1 Etalonnage par mire

L'étalonnage par mire vise à se servir d'un objet de géométrie connue appelé **mire d'étalonnage**. Un damier est très souvent utilisé car il présente des points 3D spécifiques à chaque intersection de lignes verticales et horizontales. Ensuite, un algorithme permet de déterminer les paramètres de la caméra pour lesquels l'écart entre la position des points 3D projetés et la position réelle des points est minimal. Dans les faits, on minimise la somme quadratique des erreurs de reprojection. Le système se complique dans le cas où l'on fonctionne avec deux caméras mais ce n'est pas notre cas.

2.3.2 Auto-étalonnage

L'auto-étalonnage vise à utiliser la pièce à tester comme objet d'étalonnage. Il faut pour cela une description dite **dense** de l'objet telle qu'un maillage. On cherche ainsi à trouver simunltanément les paramètres implicites de la caméra. Cette fois, on cherche à minimiser l'écart en niveau de gris entre la projection et l'image de référence.

2.3.3 Etalonnage hybride

Si l'objet d'étalonnage n'est pas tridimensionnel mais plan ou cylindrique, il faut recourir à l'étalonnage hybride pour fixer de façon explicite une partie des paramètres de la caméra (ceux de la matrice K) et identifier sous forme implicite ceux de la matrice T par auto-étalonnage.

3 Etude pratique des logiciels permettant de faire cet étalonnage

3.1 OpenCV

La calibration de la caméra peut se faire de plusieurs manières.

Une des plus connues utilisée sur openCV consiste à prendre un damier de taille connu, le capturer avec différents points de vue grâce à la caméra et trouver les coordonnées de pixel des points 3D des images grâce à la fonction findChessboardCorners().

Enfin, la méthode calibrateCamera() permet de trouver les paramètres de notre caméra.

Elle nous donnera la matrice de la camera, le coefficient de distorsion et les vecteurs de rotation et de translation.

3.2 Limite et ajout d'ArUco

Cependant, cette méthode peut poser problème en fonction du choix d'orientation de la caméra selon le point de vue. L'ajout d'ArUco permet de détecter en premier lieu les marqueurs permettant d'obtenir les coins du damier. cette détection partielle empêche toute erreur d'interprétation par la suite.

ArUco permet de détecter les coins et de les numéroter, il n'y a donc plus de problème d'orientation et de vérification pour s'assurer que l'ensemble du damier est bien pris en compte.

4 Sources

1. https://eikosim.com/articles-techniques/etalonnage-camera-principes-et-procedures/
2. https://www.kongakura.fr/article/OpenCV_Python_Tutoriel