TP Calibrage

Martin BLANCHARD, Anthony GULLIENT

12 Novembre 2019

1 Introduction

Les webcams - et même les caméras en général - sont des outils extrêmement utilisés au quotidien. Peu de gens le savent, mais ces sytèmes doivent être calibrés avant utilisation afin de déterminer la position et l'orientation des capteurs par rapport au réel. Cela permet donc de déterminer la transformation permettant de passer d'un objet défini dans un repère absolu à son image.

Dans ce TP, nous allons donc réaliser le calibrage d'une webcam Logitech grâce à une mire fournie. Le but sera de comprendre le processus de calibration et d'estimer les différents paramètres associés. In fine, nous souhaiterions vérifier la cohérence des paramètres obtenus à la fin du processus.

2 Résultats

2.1 Acquisition de la mire

On souhaite ici établir une correspondance entre les points de la mire et leur position sur l'image. Pour cela, nous allons mettre en place un protocole pour relever les coordonnées de l'objet sur l'image et pour y associer des coordonnées choisies arbitrairement.

Nous allons utiliser 2 acquisitions pour travailler car une image seule porte un biais conséquent qui donnera de mauvais résultats. Nous allons donc prendre 2 photos de la mire. La première sera prise à une distance de 30 cm de la caméra et la seconde à une distance de 15 cm.

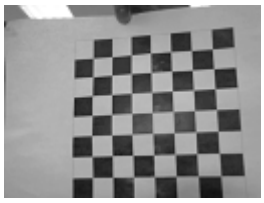

FIGURE 1 – Acquisition 1

FIGURE 2 – Acquisition 2

Nous choisissons ensuite de créer de façon arbitraire le repère visible en Figure 3, sachant que chaque case du damier mesure 20 mm.

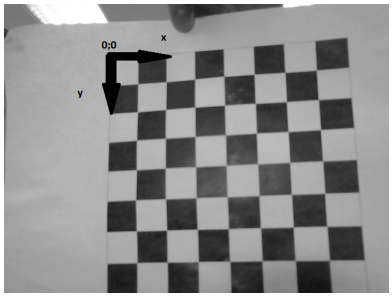

FIGURE 3 – Représentation de notre repère image

Rien de très compliqué ici, on utilise une fonction python nommée **cv2.findChessboardCorners** qui nous renvoie les coordonnées pixels des angles intérieurs du damier ainsi qu'un booléen représentant le succès de la détection.

Ces coordonnées nous permettent de déterminer dans quel sens le damier est lu par l'algorithme. Dans notre cas, il est lu du Nord au Sud, de l'Est à l'Ouest, comme spécifié dans la figure 4.

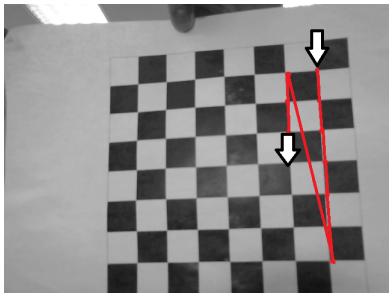

FIGURE 4 – Sens de lecture des coordonnées pixels de notre mire

Nous allons donc construire une liste de coordonnées spatiales qui suivra le même ordre, et seront définis dans le repère évoqué en Figure 3.

Ces 2 listes nous seront extrêmement utiles dans l'étape suivante, qui est le calibrage de notre caméra.

La relation entre ces coordonnées est donnée par le produit matriciel suivant, en considérant les x^c comme les coordonnées dans le repère caméra et les x^o comme les coordonnées dans le repère objet :

$$\begin{bmatrix} x_1^c \\ x_2^c \\ x_3^c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_1^o \\ x_2^o \\ x_3^o \end{bmatrix} + \begin{bmatrix} o_1^c \\ o_2^c \\ o_3^c \end{bmatrix}$$

On peut trouver une relation entre le référentiel caméra et le référentiel image, ce qui nous permet donc d'avoir d'exprimer les coordonnées image en fonction des coordonnées objet.

Le but du TP va donc être d'estimer ces paramètres de façon optimale pour nos 2 photos de mire. Nous allons donc passer à l'étape suivante qui consiste en cette estimation : le calibrage.

2.2 Calibrage de la caméra

Nous allons mettre en place un protocole pour nous permettre de calibrer notre caméra, en suivant la méthode de Tsai. Dans un premier temps, nous allons en énoncer les étapes avant de rentrer dans le détail de chacune d'entre elles :

1. Acquisition de 2 photos de la mire (similaires à celles utilisées en partie 2.1).
2. Construction des matrices A et U_1.
3. Résolution du problème $AL = U_1$ afin d'obtenir la matrice L.
4. Explicitation des paramètres du système projectif, grâce aux coefficients de L.
5. Construction des matrices B et R pour estimer les paramètres restants.
6. Calcul de o_3^c et de la taille du capteur de la webcam.

Passons maintenant aux détails de ces étapes ainsi que des matrices construites et/ou utilisées.

2.2.1 Construction des matrices A et $U1$

Pour utiliser la méthode de Tsai, il nous faut tout d'abord centrer nos coordonnées objet (celles obtenues avec la fonction **cv2.findChessboardCorners**) par rapport au centre optique. Une simple soustraction suffit et ne sera donc pas détaillée. Nous nommerons ces nouvelles coordonnées $\tilde{u}_k$. Elles nous seront utiles par la suite.

Cette étape de la méthode de Tsai va nous donner accès à un système composé de 3 matrices : A, L et U_1. Explicitons ces matrices ci-dessous :

$$A = \begin{bmatrix} \tilde{u}_2^1 x_1^{o^1} & \tilde{u}_2^1 x_2^{o^1} & \tilde{u}_2^1 x_3^{o^1} & \tilde{u}_2^1 & -\tilde{u}_1^1 x_1^{o^1} & -\tilde{u}_1^1 x_2^{o^1} & -\tilde{u}_1^1 x_3^{o^1} \\ \tilde{u}_2^2 x_1^{o^2} & \tilde{u}_2^2 x_2^{o^2} & \tilde{u}_2^2 x_3^{o^2} & \tilde{u}_2^2 & -\tilde{u}_1^2 x_1^{o^2} & -\tilde{u}_1^2 x_2^{o^2} & -\tilde{u}_1^2 x_3^{o^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \tilde{u}_2^n x_1^{o^n} & \tilde{u}_2^n x_2^{o^n} & \tilde{u}_2^n x_3^{o^n} & \tilde{u}_2^2 & -\tilde{u}_1^n x_1^{o^n} & -\tilde{u}_1^n x_2^{o^n} & -\tilde{u}_1^n x_3^{o^n} \end{bmatrix}$$

$$L^t = \begin{bmatrix} \frac{\beta r_{11}}{o_c^2} & \frac{\beta r_{12}}{o_c^2} & \frac{\beta r_{13}}{o_c^2} & \frac{\beta o_1^c}{o_c^2} & \frac{r_{11}}{o_c^2} & \frac{r_{12}}{o_c^2} & \frac{r_{13}}{o_c^2} \end{bmatrix}$$

$$U_1 = \begin{bmatrix} \tilde{u}_1^1 & \tilde{u}_1^2 & \dots & \tilde{u}_1^n \end{bmatrix}$$

Les coefficients de A et de U_1 sont connus car ils résultent de la partie 2.1. Notre inconnue est donc L dont l'explicitation des coefficients nous servira à mener à bien notre méthode de calibrage.

2.2.2 Résolution de $AL = U_1$

Nous avons face à nous un système matriciel. Instinctivement, on pourrait être tenté de multiplier par les inverses mais nous avons à faire avec des matrices non-carrées.

Heureusement, numpy nous donne accès à une fonction qui calcule ce qu'on appelle la pseudo-inverse d'une matrice.

Soit A une matrice non carrée. Sa pseudo-inverse est la matrice vérifiant la relation :

$$BAB = B$$

La méthode de Moore-Penrose nous permet donc de trouver cette matrice pseudo-inverse. On appelle donc A^{-1} cette pseudo-inverse et on obtient donc :

$$L = A^{-1}U_1$$

En utilisant la fonction **dot** de la bibliothèque numpy ainsi que la fonction **pinv** pour résoudre ce système.

On obtient ainsi la matrice L, dont les coefficients vont grandement nous aider pour la suite.

2.2.3 Explicitation des paramètres du système projectif

La suite va découler d'un raisonnement *en cascade*.

Les coefficients de L vont nous permettre de déterminer $|o_2^c|$, qui nous permettra de déterminer le coefficient β, qui lui-même nous permettra de déterminer o_1^c. L'ensemble de ces paramètres nous permet de déterminer les deux premières lignes de la matrice de rotation vue en partie 2.1 (les r_{1x} et r_{2x}).

A cette étape, nous pouvons déjà vérifier notre raisonnement et nos calculs en analysant notre espace de travail.

En effet, d'après les photos de notre mire, le centre du capteur de la caméra est proche du centre optique de la mire. En conséquence, les valeurs de o_1^c et o_2^c devraient être peu élevées. Or, dans notre cas, ces valeurs sont voisines de 2.5 cm, ce qui est tout à fait correcte.

De plus, nous pouvons également vérifier le calcul des coefficients de la rotation. Assurément, il n'y a qu'une faible rotation entre les 2 mires donc l'angle est très petit. Ainsi, on étudie avec parcimonie les coefficients r trouvées en fonction des composantes de la matrice. Cela peut permettre de trouver une erreur dans notre ordre de lecture des coordonnées. En effet, si on décide du mauvais ordre de lecture, on aurait des valeurs d'angle à 180 degrés ou 90 degrés. En cas d'erreur, il est donc conseillé de vérifier l'ordre de lecture de l'algorithme et celui donné par votre matrice (voir partie 2.1).

Il nous reste néanmoins des inconnues comme les composantes r_{3x} de la matrice de rotation. Dans la partie suivante, nous allons estimer ces paramètres en utilisant une propriété "classique" de la matrice de rotation.

2.2.4 Construction de B et R

Il nous reste maintenant à estimer les derniers paramètres nécessaires au calibrage de notre caméra, à savoir o_3^c, la distance estimée de notre caméra à notre mire, ainsi que la taille de notre capteur. D'après les formules du cours, on a l'équation matricielle suivante :

$$R = B \begin{pmatrix} o_3^c \\ f_2 \end{pmatrix}$$

avec les matrices R et B construites de la façon suivante :

$$B = \begin{bmatrix} \tilde{u}_2^1 - (r_{21}x_1^{o^1} + r_{22}x_2^{o^1} + r_{23}x_3^{o^1} + o_2^c) \\ \tilde{u}_2^2 - (r_{21}x_1^{o^2} + r_{22}x_2^{o^2} + r_{23}x_3^{o^2} + o_2^c) \\ \vdots \\ \tilde{u}_2^n - (r_{21}x_1^{o^n} + r_{22}x_2^{o^n} + r_{23}x_3^{o^n} + o_2^c) \end{bmatrix}$$

$$R = \begin{bmatrix} -\tilde{u}_2^1(r_{31}x_1^{o^1} + r_{32}x_2^{o^1} + r_{33}x_3^{o^1}) \\ -\tilde{u}_2^2(r_{31}x_1^{o^2} + r_{32}x_2^{o^2} + r_{33}x_3^{o^2}) \\ \vdots \\ -\tilde{u}_2^n(r_{31}x_1^{o^n} + r_{32}x_2^{o^n} + r_{33}x_3^{o^n}) \end{bmatrix}$$

Encore une fois, B n'est pas une matrice carrée donc n'est pas inversible. Ainsi, comme évoqué en partie 2.2.2, nous allons utiliser sa pseudo-inverse B^{-1} pour résoudre le système mis à notre disposition. On obtient donc l'équation :

$$\begin{pmatrix} o_3^c \\ f_2 \end{pmatrix} = B^{-1}R$$

Néanmoins, un problème majeur se pose : on ne connaît pas les r_{3x} et il semble donc impossible de construire la matrice R. Or, les coefficients de rotation sont en fait les composantes d'une matrice de rotation (évoquée dans la partie 2.1). Et de plus, la propriété d'une matrice de rotation est d'être orthonormale. Ses lignes sont donc génératrices d'une base orthonormale et on peut obtenir une ligne manquante en faisant le produit vectoriel des deux autres.

Posons donc les vecteurs r1, r2 et r3 tels que :

$$r1 = \begin{pmatrix} r_{11} & r_{12} & r_{13} \end{pmatrix}$$
$$r2 = \begin{pmatrix} r_{21} & r_{22} & r_{23} \end{pmatrix}$$
$$r3 = \begin{pmatrix} r_{31} & r_{32} & r_{33} \end{pmatrix}$$

On a donc d'après la propriété précédente :

$$r3 = r1 \wedge r2$$

Ainsi, on déduit les composantes r_{3x} grâce à la formule précédente et on peut donc construire R. Nous pouvons donc passer à la dernière étape de notre calibrage qui est d'estimer grâce à nos résultats, les paramètres évoqués en début de partie.

2.2.5 Calcul de o_3^c et de la taille de notre capteur

Grâce au calcul précédent, on trouve que la distance estimée à notre caméra vaut environ 19 cm, ce qui est plutôt cohérent (nous avions utilisé un double décimètre pour calculer la distance théorique). Nous avons aussi accès à f_2 qui nous permet de calculer s_2, la largeur nécessaire de capteur pour capter 1 pixel. En effet, en posant f la distance focale de notre caméra, on a :

$$s_2 = \frac{f}{f_2} \quad s_1 = \frac{f}{f_1}$$

L'unité de ces deux valeurs est le mm/pixel, car nous avons bien pris soin de garder nos longueurs en mm pendant toute la durée de notre protocole. Or, on ne connaît pas f_1 mais on peut le déduire de la formule de $\tilde{u}_1$ donnée dans le cours :

$$f_1 = \tilde{u}_1 \frac{r_{31}x_1^0 + r_{32}x_2^0 + r_{33}x_3^0 + o_c^3}{r_{11}x_1^0 + r_{12}x_2^0 + r_{13}x_3^0 + o_c^1}$$

Ces deux valeurs ainsi que la résolution de la caméra nous permettent de calculer la taille du capteur. Nous obtenons ainsi un capteur d'une taille d'environ $3mm \times 3mm$, ce qui semble cohérent avec les valeurs d'un capteur de webcam standard.

2.2.6 Rotation de la mire

On s'intéresse maintenant à une rotation de la mire. On construit donc une nouvelle image d'après une acquisition de la mire, n'ayant pas eu accès à la caméra :

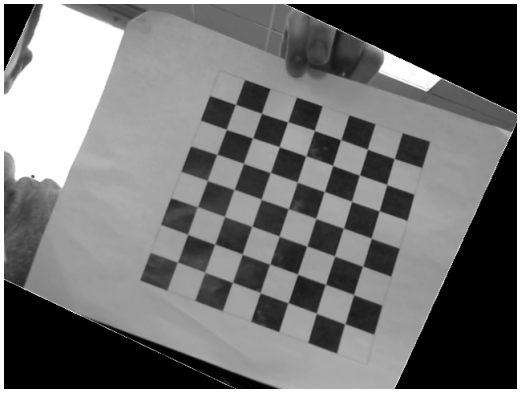

FIGURE 5 – Image rotationnée de 25 degrés

On fait attention à ce que l'image soit bien dimensionnée comme notre première acquisition (640 x 480 pixels). Nous allons vite faire face à un problème majeur : le sens de lecture des angles du damier n'est pas le même entre les deux acquisitions, à cause de la rotation. Il nous faudrait donc trier les coordonnées de notre acquisition "en rotation" pour qu'elles matchent l'ordre dans lequel sont parcourus les angles internes du damier. Avec cette nouvelle matrice, nous pourrions appliquer notre algorithme comme tout à l'heure, et trouver les composantes de rotation comme décrit en partie 2.2.4.

Nous n'avons malheureusement pas eu le temps de mettre en place cette implémentation.

3 Conclusion

Ce TP nous aura permis d'apprendre à calibrer une caméra mais aussi de discuter des différentes valeurs trouvées au fil des étapes et des possibilités de vérification de notre méthode après certains calculs. De plus, nous avons également réalisé un protocole expérimental qui pourrait parfaitement être réutilisé en cas de besoin de calibrage.

Nous pourrions également mettre au point une solution plus "automatique" de calibrage, qui ferait elle-même le tri dans les valeurs pour faire correspondre les coordonnées objet et les coordonnées image.