Etalonnage de robots par vision

Nicolas Andreff

Novembre 2012

Fondation unit
Université Numérique Ingénierie et Technologie

GDR ROBOTIQUE

Institut Français de Mécanique Avancée

U.V. Étalonnage et Identification des Systèmes

TP Étalonnage de caméra

Nicolas Andreff

Année 2001–2002

1 Partie "presse-bouton"

Préliminaires:

1. Créer le répertoire : `D:\UV EIS\Etalonnage Camera\Matlab\TOOLBOX_Calib`
2. Copier `D:\andreff\UV EIS\Etalonnage Camera\TOOLBOX_Calib.zip` dans `D:\UV EIS\Etalonnage Camera\Matlab\TOOLBOX_Calib.zip`
   Décompresser l'archive
3. Copier `D:\andreff\UV EIS\Etalonnage Camera\Images.zip` dans `D:\UV EIS\Etalonnage Camera\Images.zip`
   Décompresser l'archive
4. Lancer Matlab
5. Modifier le chemin d'accès :

   ```
   >> path(path,'D:\UV EIS\Etalonnage Camera\Matlab\TOOLBOX_Calib');
   ```
6. Changer le répertoire de travail :

   ```
   >> chdir('D:\UV EIS\Etalonnage Camera\Images');
   ```

1.1 Votre premier étalonnage

1. Lancer `calib_gui`

   ```
   >> calib_gui
   ```

   Choisir l'option `Standard`
2. Charger les images en mémoire

   Cliquer sur `Image names`
3. Extraire les points de l'image

   Cliquer sur `Extract grid corners`
   Choisir les images 1 à 3
   (Répondre `1:3` à la question `Number(s) of image(s) to process ([] = all images) =`)
   Suivre les indications en choisissant les valeurs par défaut de tous les autres paramètres.

   ATTENTION : Garder la même origine dans toutes les images !
4. Lancer l'étalonnage

   Cliquer sur `Calibration`.
   Interpréter les résultats.

5. Estimer la qualité de l'étalonnage

   Cliquer sur `Show Extrinsic`
   Cliquer sur `Reproject on images`
   Interpréter les résultats.
   On peut cliquer sur `Analyse error` pour accéder à un peu plus d'information.

6. Sauvegarder les résultats

   Cliquer sur `Save`

1.2 Observation de la distorsion

Les paramètres obtenus par l'étalonnage précédent sont optimisés par rapport au jeu de données fourni. La question à laquelle il nous faut maintenant répondre est : ces paramètres sont-ils valables pour d'autres images ? Autrement dit, a-t-on suffisamment excité le système ?

Pour cela, nous allons calculer la pose de la mire par rapport à la caméra lorsque la mire est observée sous un nouveau point de vue.

Cliquer sur `Comp. Extrinsic`. Choisir l'image 15.
Interpréter le résultat.

1.3 Etalonnage complet

1. Extraire les coins de toutes les images

   Pour l'image 15, on utilisera l'option d'estimation de la distorsion. On prendra la valeur `kc=-0.3`.

2. Étalonner

3. Analyser l'erreur

   Observer que les erreurs de reprojection sont globalement importantes.
   Au passage, à votre avis, à partir de quand peut-on dire qu'une erreur de reprojection est importante ?

4. Optimiser l'étalonnage

   Cliquer sur `Recomp. corners` et choisir toutes les images. Étalonner à nouveau.
   Observer que les erreurs ont été largement réduites.
   Comment expliquer cela ? Que s'est-il passé ?
   Certaines images ont malgré tout des erreurs beaucoup plus importantes que les autres. On peut alors cliquer à nouveau sur `Recomp. corners` en modifiant les paramètres `wintx` et `winty` pour ces images.

1.4 Identifiabilité

Quitter `calib_gui`.

Effacer les variables utilisées jusqu'à présent :

```
>> clear all
```

Relancer `calib_gui`. Extraire les coins d'une seule image. Étalonner. Que se passe-t-il ?

1.5 Optionnel : à faire à la fin du TP s'il reste du temps

Jouer avec `Add/Suppress images`.

Changer les valeurs de `est_dist`, `est_alpha` et/ou `est_kc`..

2 Étalonnage d'une caméra réelle

2.1 Acquisition d'images

Utiliser *Intellicam* pour prendre des images de la mire qui vous est fournie.

1. D'après ce qui précède, choisir une stratégie de prise de vue. Sauvegarder les images sous la forme `ImageCTT*.tif` dans le répertoire `D:\UV EIS\Etalonnage Camera\Images`.

2. A partir de la dernière image, faire 2 ou 3 mouvements particuliers (aussi précisément que possible), par exemple: rotation de 90, translation pure de 10cm selon l'axe X de la mire, translation pure de 20cm selon l'axe Y, etc. Prendre une image après chaque mouvement (la sauvegarder sous la forme `Mvt*.tif`) et noter les mouvements effectués.

3. Prendre de plus quelques images (`Petite*.tif`) où la mire ne couvre pas plus d'un quart de l'image.

2.2 Étalonnage

1. Utiliser `calib_gui` avec `ImageCTT*.tif`. Sauvegarder les résultats.

   La focale est-elle la mme que celle qui est inscrite sur l'objectif ? Pourquoi ?

2. Calculer la pose de la mire dans les images `Petite*.tif`.

   Conclusion ?

3. Quitter `calib_gui`, effacer les variables et recommencer les étapes 1. et 2. en intervertissant les rles de `ImageCTT` et `Petite`.

   Comparer et conclure.

4. Reprendre les résultats du 1. Calculer les poses des images `Mvt*.tif`. Retrouvez-vous les mouvements effectués ?

3 Faites-le vous-même

Dans cette partie, nous allons reproduire une partie de `calib_gui`. Le texte qui suit vous donne les grandes lignes de l'algorithme ainsi que quelques pointeurs vers les fonctions Matlab à mettre en oeuvre. A vous d'écrire les fonctions qui manquent.

Ne pas hésiter à consulter l'aide en ligne.

3.1 Traitement d'image

1. Lire une des images (par exemple `Image1.tif`) et la visualiser.

```
>> I=imread('Image1.tif','tif');
>> imshow(I)
>> hold on
```

2. Cliquer sur les 4 coins extrêmes

```
>> [x,y]=ginput3(4);
```

3. Extraction des coins au voisinage des points cliqués.

   Par souci de simplicité, nous utilisons ici le détecteur de coins de `calib_gui`.

```
>> wintx =5;
>> winty =5;
>> [xc,good,bad,type] = cornerfinder([x'; y'],I,wintx, winty);
```

   Sauvegarder `xc` et afficher les points extraits

```
>> xc5=xc;
>> plot(xc(1,:), xc(2,:),'r+');
```

   Jouer avec les valeurs de `wintx` et `winty` et observer ce qui se passe. Pour afficher autrement remplacer `'r+'` par `'bo'` ou `'md'` ou `'gx'` etc.

4. Reprendre les résultats pour `wintx=winty=5`

```
>> xc=xc5;
```

3.2 Calcul de l'homographie initiale

Construire le modele 3D de la grille à partir de la taille réelle des cases. Ce modèle doit être sous la forme d'une matrice 4xN ou N est le nombre de points et chaque point est de la forme `[X;Y;0;1]` Appelons `pts3D` ce modèle et `XC` les 4 coins extrêmes de ce modèle.

A partir de `xc` et de `XC`, déterminer une méthode numérique pour estimer la transformation plan-plan (homographie) qui transforme le plan de la grille physique en le plan de la grille projetée dans l'image :

$$s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{h_1} & \mathbf{h_2} & \mathbf{h_3} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

où s est un facteur d'échelle inconnu. Cela implique bien évidemment que $\mathbf{H}$ est définie à un facteur d'échelle près.

Implanter cette méthode dans une fonction Matlab : `function H = homographie(pts2d,pts3d)`

En déduire une estimation de la projection des points de la grille dans l'image.

Extraire tous les coins de l'image en utilisant l'estimation précédente. Appelons `pts2d` ces coins.

3.3 Étalonnage

Dans cette partie, nous allons utiliser un algorithme différent de celui de `calib_gui`. En particulier, nous n'utiliserons pas de modèle de distorsion d'image. Cela simplifiera le programme à réaliser.

3.3.1 Calcul de l'homographie

De la même manière que précédemment, calculer l'homographie $\mathbf{H}$ qui relie les `pts3D` aux `pts2D`.

3.3.2 Estimation des paramètres

Analyse du problème La forme générale de la projection d'un point 3D dans l'image s'écrit :

$$s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{KR} & \mathbf{Kt} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{Kr_1} & \mathbf{Kr_2} & \mathbf{Kr_3} & \mathbf{Kt} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad \text{avec} \quad \mathbf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

où $\mathbf{K}$ est la matrice des paramètres intrinsèques et $(\mathbf{R}, \mathbf{t})$ est la pose de la mire par rapport à la caméra.

Dans le cas de l'étalonnage, les inconnues de cette équation sont $s, \mathbf{K}, \mathbf{R}$ et $\mathbf{t}$ et les connues sont les points 2D et 3D.

Dans notre cas, $Z = 0$. Par conséquent, l'équation précédente devient :

$$s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{Kr_1} & \mathbf{Kr_2} & \mathbf{Kt} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

On en déduit donc que :

$$\begin{pmatrix} \mathbf{h_1} & \mathbf{h_2} & \mathbf{h_3} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{Kr_1} & \mathbf{Kr_2} & \mathbf{Kt} \end{pmatrix}$$

où λ est un scalaire quelconque.

Sachant que $\mathbf{r_1}$ et $\mathbf{r_2}$ sont orthonormaux, on en déduit 2 contraintes :

$$\begin{aligned} \mathbf{h_1}^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h_2} &= 0 \\ \mathbf{h_1}^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h_1} &= \mathbf{h_2}^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h_2} \end{aligned}$$

Notons à présent $\mathtt{B} = \mathbf{K}^{-T}\mathbf{K}^{-1}$ et $\mathtt{b} = (\mathbf{B}(1,1), \mathbf{B}(1,2), \mathbf{B}(2,2), \mathbf{B}(1,3), \mathbf{B}(2,3), \mathbf{B}(3,3))$.

Montrer que

$$\mathtt{B} = \begin{pmatrix} \frac{1}{\alpha_u^2} & 0 & -\frac{u_0}{\alpha_u^2} \\ 0 & \frac{1}{\alpha_v^2} & -\frac{v_0}{\alpha_v^2} \\ -\frac{u_0}{\alpha_u^2} & -\frac{v_0}{\alpha_v^2} & \frac{u_0^2}{\alpha_u^2} + \frac{v_0^2}{\alpha_v^2} + 1 \end{pmatrix}$$

Montrer aussi qu'on peut écrire :

$$\mathbf{h_i}^T \mathtt{B} \mathbf{h_j} = \mathbf{w_{ij}}^T \mathtt{b}$$

où $\mathbf{w_{ij}} = (h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3})$. Nos deux contraintes s'écrivent donc :

$$\begin{pmatrix} \mathbf{w_{12}} \\ \mathbf{w_{11}} - \mathbf{w_{22}} \end{pmatrix} \mathtt{b} = 0$$

Cette équation est valable pour une image donnée. En effet, pour chaque image, on peut calculer $\mathbf{h_1}$, $\mathbf{h_2}$ et $\mathbf{h_3}$ et en déduire les $\mathbf{w_{ij}}$.

En accumulant n images, nous disposons donc de n équations de ce type que nous pouvons superposer pour obtenir :

$$\mathtt{W}\mathtt{b} = 0$$

qui est un système linéaire dont le noyau est de dimension 1. La solution `b` de ce système est donc définie à un facteur d'échelle près et la solution de norme 1 est le vecteur singulier associé à la plus petite valeur singulière de `W`.

Première étape de résolution : calcul de `B` On trouve donc `B` par quelque chose qui ressemble à :

```
>> W=...;
>> [u,sigma,v]=svd(W);
>> b=v(:,6);
>> B=reshape(b,3,3)';
```

où `W` est construite à partir des homographies calculées pour chaque image et `B` est définie à un facteur d'échelle μ inconnu.

Ecrire la fonction Matlab qui implante complètement le calcul de `B` : `function B = CalculB(H);`

Deuxième étape de résolution : Extraction des paramètres intrinsèques Montrer que l'on retrouve alors les paramètres intrinsèques par les formules suivantes :

$$\begin{aligned} u_0 &= -\mathbf{B}(1,3)/\mathbf{B}(1,1) \\ v_0 &= -\mathbf{B}(2,3)/\mathbf{B}(2,2) \\ \mu &= \mathbf{B}(3,3) + u_0 * \mathbf{B}(1,3) + v_0 \mathbf{B}(2,3) \\ \alpha_u &= \sqrt{\mu/\mathbf{B}(1,1)} \\ \alpha_v &= \sqrt{\mu/\mathbf{B}(2,2)} \end{aligned}$$

Ecrire la fonction Matlab qui fait ce calcul : `function K=intrinseques(B)`

Troisième étape de résolution : Calcul de la pose Pour chaque image, on peut déterminer, à partir de $\mathbf{H}$ et $\mathbf{K}$, la pose $(\mathbf{R}, \mathbf{t})$ de la mire par rapport à la caméra.

Ecrire la fonction qui implante ce calcul : `function [R, t]=pose(K,H)`

3.3.3 Recollez les morceaux

... et comparer vos résultats à ceux de `calib_gui`.

Si vous êtes arrivés jusqu'ici, bravo !