Compte-rendu TP1 VisA: Callibration de caméra

Gaëtan DEFLANDRE

September 14, 2015

1 Introduction

Dans le domaine de l'optique, nous avons vu que la projection d'un objet 3D sur le capteur d'une caméra (plan image à deux dimensions), dépend de plusieurs paramètres:

- Les coordonnées homogènes de l'objet dans la scène.
- La matrice extrinsèque, qui donne les rotations et translations de la caméra dans scène.
- La matrice intrinsèque, qui donne les paramètres internes de la caméra, pouvant changer selon les réglages de la caméra.

Dans certains cas, les paramètres intrinsèques et/ou extrinsèques de la caméra sont inconnus. Il existe plusieurs méthodes pour retrouver ces paramètres.

La méthode de Zhang est l'une d'entre elles. Pour appliquer cette méthode, il nous faut plusieurs images d'un plan dont les coordonnées sont connues dans la scène. Il faut conserver les mêmes réglages pour les différentes images.

2 Implémentation sous Scilab

Pour implémenter la méthode de Zhang, nous utilisons Scilab. La macro se découpe en trois parties:

- Initialisation des variables.
- Approximation de la matrice intrinsèque.
- Approximation des matrices extrinsèques.

M reprèsente les points de la mire dans la scène 3D. m reprèsente les points de la projection de la mire sur les plans images 2D, des différentes prise de vue. Nous disposons de quatre images, donc quatres jeux de valeurs pour m. Ces coordonnées sont connues et nous les utiliserons pour les calculer. Une lecture de fichier est nécesssaire pour initialiser M et m.

```
// Nombre d'images
ni = 4;
// Lire les coordonnees des points de la mire dans la scene
M = read('points.txt', -1, 2)';
...
// Boucler pour toutes les images
for i = 1:ni
  // Lire les points de l'image
  m(1:2,:,i) = read('points-'+string(i)+'.txt', -1, 2)';
  m(3,:,i) = ones(1, np);
  ...
end
...
```

Listing 1: Lecture de fichier pour retrouver M et m

2.1 Matrice intrinsèque

Étant donné que la mire est plane, nous n'avons pas besoin de la profondeur donnée par l'axe Z dans les coordonnées de la mire par rapport à la scène. Dans Scilab, cela se traduit par: **M(sansZ,:)**. Nous avons donc:

$$\begin{aligned} m_i &= A \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \\ m_i &= A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \end{aligned} \tag{1}$$

Où A est la matrice intrinsèque caractèrisée par:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Avec, cela, il est possible d'estimer un homographie en considérant les équations suivantes (annexe, ligne 24):

$$\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{3}$$

$$\begin{aligned} h_1^T A^{-T} A^{-1} h_2 &= 0 \\ h_1^T A^{-T} A^{-1} h_1 &= h_2^T A^{-T} A^{-1} h_2 \end{aligned} \tag{4}$$

Ensuite, nous pouvons retrouver les contraintes de homographie curante avec l'équation suivante (annexe, ligne 26).

$$v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T \tag{5}$$

Traduit en dans Scilab:

```
1  /// \brief Calcule un terme de contrainte a partir d'une
       homographie.
2  ///
3  /// \param H: matrice 3*3 d finissant l'homographie.
4  /// \param i: premiere colonne.
5  /// \param j: deuxieme colonne.
6  /// \return vecteur definissant le terme de contrainte.
7  function v = ZhangConstraintTerm(H, i, j)
8
9    // /!\ matrice scilab: M(y,x), soit M(row,col)
10
11   // La matrice non transposer
12   w = [];
13   w(1) = H(1,i) * H(1,j);
14   w(2) = (H(1,i) * H(2,j)) + (H(2,i) * H(1,j));
15   w(3) = H(2,i) * H(2,j);
16   w(4) = (H(3,i) * H(1,j)) + (H(1,i) * H(3,j));
17   w(5) = (H(3,i) * H(2,j)) + (H(2,i) * H(3,j));
18   w(6) = H(3,i) * H(3,j);
19
20   // Transpos e de w
21   v = w';
22 endfunction
```

Listing 2: Fonction ZhangConstraintTerm

Une fois itéré pour chaque image de m. Nous pouvons alors en déduire les valeurs B, une matrice symétrique, en fonction des diffèrentes estimation d'homographies.

$$B = A^{-T} = A^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{11} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{6}$$

Les valeurs de B sont calculées et rangées dans une vecteur b (ligne 12, listing 4).

Une fois le vecteur b obtenu, nous pouvons approximer les valeurs de la matrice intrinsèque:

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \\ \alpha &= \sqrt{\lambda/B_{11}} \\ \beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\ \gamma &= -B_{12}\alpha^2\beta/\lambda \\ u_0 &= \gamma v_0/\beta - B_{13}\alpha^2/\lambda \, . \end{aligned}$$

Voici la fonction dans Scilab:

```
1  /// \brief Calcule la matrice des parametres intrinseques.
2  ///
3  /// \param b: vecteur resultant de l'optimisation de Zhang.
4  /// \return matrice 3*3 des parametres intrinseques.
5  function A = IntrinsicMatrix(b)
6    A = zeros(3, 3);
7
8    A(3,3) = 1;
9
10   v0 = (b(2)*b(4) - b(1)*b(5)) / (b(1)*b(3) - b(2)*b(2));
11   lambda = b(6) - ( (b(4)*b(4) + v0*(b(2)*b(4) - b(1)*b(5))) / b
       (1) );
12   alpha = sqrt(lambda/b(1));
13   mybeta = sqrt((lambda*b(1)) / (b(1)*b(3) - b(2)*b(2)))
14   mygamma = ((-b(2)) * alpha * alpha * mybeta) / lambda;
15   u0 = (mygamma*v0)/mybeta -(b(4)*alpha*alpha)/lambda;
16
17   A(1,1) = alpha;
18   A(  \begin{bmatrix}
19    \alpha & \gamma & u_0 \\
20    0 & \beta & v_0 \\
21    0 & 0 & 1 \\
22   \end{bmatrix}1,2) = mygamma;
23   A(1,3) = u0;
24   A(2,2) = mybeta;
```

```
  A(2,3) = v0;
endfunction
```

Listing 3: Fonction d'approximation de la matrice intrinsèque à partir de b

Nous retrouvons la matrice intrinsèque suivante:

$$A = \begin{bmatrix} 3498.2767 & -3.1310503 & 336.76583 \\ 0. & 3503.8946 & 220.1142 \\ 0. & 0. & 1. \end{bmatrix} \tag{7}$$

2.2 Matrice extrinsèque

Maintenant que nous avons estimé les valeurs de A, nous pouvons approximation les matrices extrinsèques pour chaque image de m. Nous utilisons les formules suivantes:

$$\begin{aligned} r_1 &= \lambda A^{-1} h_1 \\ r_2 &= \lambda A^{-1} h_2 \\ r_3 &= r_1 * r_2 \\ t &= \lambda A^{-1} h_3 \end{aligned} \tag{8}$$

Ce qui nous donne la fonction suivante dans Scilab:

```
/// \brief Calcule la matrice des parametres extrinseques.
///
/// \param iA: inverse de la matrice intrinseque.
/// \param H: matrice 3*3 definissant l'homographie.
/// \return matrice 3*4 des parametres extrinseques.
function E = ExtrinsicMatrix(iA, H)

  lambda = 1 / norm(iA*H(:,1));

  r1 = lambda * iA * H(:,1);
  r2 = lambda * iA * H(:,2);
  r3 = r1 .* r2;
  t = lambda * iA * H(:,3);

  E = [r1,r2,r3,t];
  disp(E);
endfunction
```

Listing 4: Fonction d'approximation de la matrice extrinsèque

Nous retrouvons les matrices extrinsèques suivantes:

Image 1	$\begin{bmatrix} 0.9999998 & 0.0009052 & 0.0009052 & -48.811558 \\ 0.0000377 & 0.9982948 & 0.0000376 & 54.733305 \\ 0.0006696 & -0.0015763 & -0.0000011 & 9854.3606 \end{bmatrix}$
Image 2	$\begin{bmatrix} 0.7124496 & 0.0007762 & 0.0005530 & -46.123202 \\ -0.0039703 & 1.0010299 & -0.0039744 & 43.830318 \\ -0.7017120 & -0.0006379 & 0.0004476 & 7905.8899 \end{bmatrix}$
Image 3	$\begin{bmatrix} 0.9848432 & 0.1745697 & 0.1719238 & -43.812292 \\ -0.1734468 & 0.9833136 & -0.1705526 & 49.196563 \\ 0.0002832 & 0.0005524 & 0.0000002 & 8870.6728 \end{bmatrix}$
Image 4	$\begin{bmatrix} 1. & 0.0045336 & 0.0045336 & -143.8696 \\ 0.0000023 & 0.7020868 & 0.0000016 & 42.078153 \\ -0.0000609 & -0.714386 & 0.0000435 & 8872.4252 \end{bmatrix}$

2.3 Retrouver la focale

Soit A la matrice intrinsèque de la caméra qui peut s'exprimer par les formules suivantes:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} k_u f & s_{uv} f & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} k_u & s_{uv} & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Si nous disposons de la taille du capture en x et en y, ainsi que du nombres de pixels en x et en y du capteur. Alors nous retrouver le facteur d'échelle en x et en y:

$$\begin{aligned} k_u &= \frac{nombre\_de\_pixel\_en\_x}{taille\_du\_capteur\_x\_en\_mm} \\ k_v &= \frac{nombre\_de\_pixel\_en\_y}{taille\_du\_capteur\_y\_en\_mm} \end{aligned} \tag{10}$$

Donc, nous calculons la focale avec:

$$\begin{aligned} f &= \frac{\alpha}{k_u} \\ &= \frac{\beta}{k_v} \end{aligned} \tag{11}$$

Si nous disposons des coordonnées du centre de l'image, u_0 et v_0. Alors nous pouvons utiliser ces valeurs exacte plus que leurs estimation lors des calcule de la matrice intrinsèque dans la méthode Zhang.

3 Conclusion

Pour conclure, la méthode Zhang permet de retrouver la matrice intrinsèque et les matrice extrinsèques des images d'un objet dont nous connaissons les coordonnées dans la scène.

Ce type de méthode permet d'appréhender d'autres technique tel que les tenseurs trifocale.

4 Annexe

```
// Definition des fonctions utilitaires
exec ('glue.sci', -1);
exec ('gaetan-deflandre.sci', -1);
// Nombre d'images
ni = 4;

// Lire les coordonnees des points de la mire dans la scene
M = read('points.txt', -1, 2)';
np = size(M, 2);
M = [M; zeros(1, np); ones(1, np)];
sansZ = [1, 2, 4];
// Initialiser la matrice des contraintes
V = [];
// Matrices des homographies
H = zeros(3, 3, ni);
// Coordonnees de tous les points image
m = zeros(3, np, ni);
// Boucler pour toutes les images
for i = 1:ni
  // Lire les points de l'image
  m(1:2,:,i) = read('points-'+string(i)+'.txt', -1, 2)';
  m(3,:,i) = ones(1, np);
  // Estimer l'homographie entre la mire et l' image
  H(:,:,i) = ZhangHomography(M(sansZ,:), m(:,:,i));
  // Ajouter deux lignes de contraintes dans V
  V = [V; ZhangConstraints(H(:,:,i))];
end
// Calculer le vecteur b
b = SmallestRightSingular(V);
// Estimation de la matrice intrinseque
A = IntrinsicMatrix(b);
iA = inv(A);
// Estimations des matrices extrinseques
E = zeros(3, 4, ni);
for i = 1:ni
  E(:,:,i) = ExtrinsicMatrix(iA, H(:,:,i));
end
```

Listing 5: Macro générale qui va appeler les fonctions