Métrologie 3D par stéréoscopie

1 Démarche expérimentale

L'étalonnage de chacune des caméras puis du système complet s'effectuera à l'aide de la toolbox *Camera Calibration* [1] sous MATLAB tandis que la génération des points 3D s'effectue à partir d'un logiciel de stéréo-corrélation dense développé au LAAS-CNRS et compilé sous CYGWIN.

1.1 Etalonnage du système

1.1.1 Acquisition des images de la mire

Régler chacune des caméras (mise au point, ouverture). Faire l'acquisition d'une dizaine de paires d'images respectivement pour la caméra gauche et droite. A chaque fois, la mire devra être vue entièrement dans les DEUX images. Les images seront acquises à l'aide de *Paint Shop Pro.* et stockées sous le répertoire *stereo/calibration/src/images*. La convention pour nommer les images est la suivante : $< left >< num >$ *ou* $< right >< num >$ resp. pour la caméra gauche et droite. Elles sont au format standard : bmp, jpeg,...

Attention la caméra gauche (préfixe $< left >$) correspond à celle situé à droite pour un observateur extérieur. La numérotation sera 01, 02, ...

1.1.2 Etalonnage de chaque caméra

<u>A. Démarche</u>

Lancer MATLAB. Donner l'accès aux librairies *tools_ars* et *TOOLBOX_calib* qui sont spécifiques à l'étalonnage d'un système stéréoscopique (ajouter leurs chemins par *set Path*). Lancer la commande *calib* sous MATLAB. On étalonnera les deux caméras l'une après l'autre. La démarche identique pour l'une ou l'autre des caméras se résume comme suit :

1. Lire les images relative à la caméra gauche (boutons *Image names* et/ou *Read images*).
2. Extraire les indices visuels (bouton *Extract grid corners*). On laissera les paramétres par défaut. Clicker sur les 4 coins extrêmes de la grille LE PLUS PRECISEMENT POSSIBLE. Pour les images suivantes, on séléctionnera ces coins dans le même ordre. Pratiquement le premier coin clické sera associé à l'origine de la mire dans son repère propre tandis que les trois autres peuvent être choisis arbitrairement.
3. Rentrer les tailles dX et dY en X et Y de chaque carré de la grille. Cette distance sera fixée ici à 15 mm.
4. Répéter l'étape 2 pour toutes les images.
5. On choisit un modèle de distorsion $r_1 t_0$. Il est défini par la variable *est_dist* [1], soit ici : $est_dist = [1; 0; 0; 0.0]$
6. Lancer le processus d'optimisation (bouton *Calibration*) qui permet d'estimer les paramètres de calibrage. On se réferera à la doc [1] pour interpréter les résultats obtenus.
7. Sauvegarder le résultat du calibrage dans le fichier MATLAB *Calib_Results.mat* (bouton *Save*). Ce fichier contient les paramètres estimés, à savoir :
   - Les paramètres et matrice relatives aux intrinsèques I,
   - la ou les matrices extrinsèques E,
   - les paramètres de distorsion K.

Etalonner la caméra droite par la même démarche en prenant soin de renommer le fichier *Calib_Results.mat* en *Calib_Results_left.mat*. On renommera en final le fichier *Calib_Results.mat* en *Calib_Results_right.mat*

B. Analyse des résultats et commentaires

L'analyse des résidus de reprojection sont visualisables par le bouton *Analyse error* tandis que la reprojection des points 3D pour chacune des images de mire sont observables par le bouton *Reproject on images*. **Que représente ce résidu ?**

Relever les paramètres estimés du modèle de distorsion et conclure sur la distorsion : barillet ou coussinet ? Pourquoi avoir choisi un modèle r_1t_0 ?. Relever les valeurs des paramètres intrinsèques. Comparer avec les valeurs "constructeur". Comment justifier les différences obtenues sur u_0 et v_0 ? On rappelle :

$$\alpha_u = \frac{di.f}{C_x}, \alpha_v = -\frac{dj.f}{C_y}, u_0 = \frac{di}{2}, v_0 = \frac{dj}{2}$$

où C_x, C_y est la taille métrique de la matrice CCD, respectivement horizontalement et verticalement. di et dj sont respectivement le nombre de pixels ligne et colonne de l'image numérisée (ici 576 et 768). f est la focale de l'objectif utilisé. **Pourquoi α_u et α_v sont-ils de signes opposés ?**

Le bouton *Show extrinsic* permet enfin de visualiser les positions relatives caméra/mire pour chacune des prises de vues.

1.1.3 Estimation de la base entre les deux caméras

A. Démarche

Lancer la commande *stereo_gui* sous MATLAB. La démarche est alors la suivante :

1. Charger les fichiers *Calib_Results_left.mat* et *Calib_Results_right.mat*.
2. Estimer la base.
3. Sauvegarder le résultat dans le fichier *Calib_Results_Stereo.m* par le bouton *Save*.

B. Analyse des résultats et commentaires

La commande *to calife* permet de dispatcher et stocker les résultats de calibrage (contenus dans *Calib_Results*.mat*) dans différents fichiers ASCII. Ces fichiers ont pour extension le nom de la matrice sauvée : *.i, .e1, .k ou .ie1,...* avec pour préfixe $< .g >$ *ou* $< .d >$. Ces fichiers sont générés sous le répertoire *stereo/calibration/src/images/vecteurs*. **Identifier le contenu de chacun de ces fichiers. Comment envisagez-vous le processus d'optimisation pour l'estimation de la base ?**

1.2 Reconstruction 3D

1.2.1 Acquisition d'images d'objets 3D

Faire l'acquisition d'images pour les quelques objets 3D texturés qui vous sont proposés. Les images seront nécessairement au format *"gif"* et stockées sous *stereo/stereoApp/src/examples/SNAPPER/images*. La convention pour nommer les images est ici la suivante : $< obj >< num_ >< d\ ou\ g > .gif$ d ou g respectivement pour caméra gauche ou droite.

1.2.2 Génération des points 3D

Nous supposons ici le système stéréoscopique préalablement étalonné. Sont connus en particulier :

- les paramètres intrinsèques de chaque caméra, stockés dans les fichiers *intrinseque.g(ou .d)* sous le répertoire *stereo/stereoApp/src/examples/SNAPPER/vecteurs*. Idem pour le modèle de distorsion et les fichiers *distorsion.g(ou .d)*.
- les paramètres de rectification (matrice 3×3), stockés dans les fichiers *rectification.g(ou .d)* sous le même répertoire.
- les fichiers *intrinseque.rectifie.g(ou .d)* combinaison des paramètres précédents toujours sous le même répertoire.

A. Démarche

Copier les fichiers résultats de calibrage initialement sous *stereo/calibration/src/images/vecteurs* dans le répertoire *stereo/stereoApp/src/examples/SNAPPER/vecteurs*. Lancer l'éxécutable *stereo/stereoApp/src/examples/stereo.exe* (plus exactement son raccourci). La démarche est alors la suivante :

1. Sélectionner un facteur de réduction de 2 et un score de corrélation ZNCC.
2. Capturer les images par les boutons *init, capture* puis *display images*.
3. Rectifier les images par les boutons *rectify* puis *display rectification*. La rectification et la correction en distorsion des images seront étudiées en § 2.1.2.
4. Lancer la corrélation et afficher l'image de disparité par les boutons *correlate* puis *display disparity*. La corrélation ZNCC sera étudiée en § 2.2.1.
5. Générer les points 3D par *reconstruct* puis *export*. Les points 3D reconstruits au format ASCII sont stockés dans les fichiers *"test.off"* et *"test.3d"* sous le répertoire *stereo/stereoApp/src/examples/images3d*. La triangulation sera étudiée en § 2.2.2

B. Analyse des résultats et commentaires

Comment interpréter l'image obtenue ? Noter le nombre de points 3D reconstruits. Juger qualitativement de la reconstruction obtenue en **lancant le fichier batch** *pmask.bat* sous le répertoire *stereo/stereoApp/src/examples/* Recopier au préalable les fichiers test.* sous ce même répertoire. Cet outil permet également de visualiser la reconstruction avec la texture associée. Une évaluation quantitative des erreurs de reconstruction 3D sera menée en § 2.2.3.

2 Etude de quelques fonctions élémentaires

Nous nous focalisons ici sur quelques fonctions élémentaires de l'algorithme de stéréovision dense. Le but est ici de compléter ces fichiers MATLAB (fichiers à trous). Ces fichiers sont stockés sous *stereo/Matlab* tandis que les listings de tous ces fichiers sauf *rectif.m* sont donnés en annexe. **Il est conseillé de regarder ces fichiers avant la séance.**

2.1 Etalonnage du système

2.1.1 Extraction des indices visuels

Pour une mire constituée de croix, chaque croix est recherchée dans une fenêtre englobante, en estimant les paramètres X d'une fonction $f(U,V,X)$, qui donne la luminance en tout pixel (U,V) dans un voisinage de la croix [2]. Ce modèle est estimé par minimisation de l'erreur entre les luminances calculées par le modèle et celles de l'image réelle. La fonction à minimiser est la suivante :

$$err = \sum_{j=1}^{n} [f_j(U,V,X) - I_j(U,V)]^2$$

avec $f_j(U,V,X) = a + b.U_j + c.V_j + d.exp(-f^2.U_j^2) + e.exp(-g^2.V_j^2)$, n le nombre de pixels inclus dans la fenêtre et $X = (a,\ b,\ c,\ d,\ e,\ f,\ g,\ u_c,\ v_c,\ \varphi,\ \psi)^t$ tel que :

$$\begin{pmatrix} U_j \\ V_j \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & -u_c \\ 0 & 1 & -v_c \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} u_j \\ v_j \\ 1 \end{pmatrix}$$

1. **Quel rôle joue chacun des coefficients ? On lancera le programme** *croix.m* **qui trace la fonction croix** f **en fonction des paramètres** X**;** les coefficients seront tous pris positifs sauf d et e.
2. La minimisation du critère *err* est décrite dans le fichier *extract_croix.m*. Elle repose sur une optimisation non linéaire. **Mettre en équation le système à résoudre puis compléter ce fichier et notamment la fonction** *Find_1Croix* **qui calcule les jacobiennes du critère par rapport au vecteur** X**.** On pourra évaluer qualitativement le résultat en visualisant (comparant) les images *modIm.jpg* et *windIm.jpg* correspondant respectivement au modèle et l'image de la croix sélectionnée.

3. Après estimation du vecteur X, une série de tests permet de vérifier qu'il s'agit bien d'une croix. **Comment les interpréter ?** Ces tests sont listés ci-après :
   - $err < Seuil_1$
   - $2 \mid \frac{d-e}{d+e} \mid < Seuil_2$
   - $\mid d \mid < Seuil_3$ et $\mid e \mid < Seuil_3$

2.1.2 Correction de distorsion et rectification des images

On se propose d'écrire un programme permettant de corriger en distorsion une paire d'images stéréo puis de les rectifier. La paire d'images traitées ici sera *obj1_g.jpg* et *obj1_d.jpg*.

1. **Compléter le programme** *distor_g.m* **et** *distor_d.m* **respectivement pour la caméra gauche et droite.** Le calcul du niveau de gris dans l'image corrigée sera calculée par interpolation bilinéaire. L'image corrigée est appelée *obj1_g2.jpg* ou *obj1_d2.jpg* respectivement pour la caméra gauche et droite. **Pour ce modèle de distorsion, quelle est la plus grande distorsion observable dans l'image ?**
2. A partir du programme *rectif.m* **appliquer une rectification épipolaire à ces deux nouvelles images.** On constatera que les lignes épipolaires sont bien alignées. Les images rectifiées sont notées respectivement *obj1_g3.jpg* et *obj1_d3.jpg*. Il existe divers algorithmes de rectification ([6], [5]); la méthode de rectification utilisée ici est détaillée dans [3]. Les nouveaux paramètres intrinsèques des deux caméras (virtuelles) en rectification épipolaire sont pris identiques.

2.1.3 Estimation des paramètres et de la base entre les deux caméras

1. **Mettre en équation le système à résoudre.**
2. On désire vérifier la pertinence du positionnement relatif estimé entre les deux caméras. **Compléter le fichier** *baseline.m* **qui s'appuie sur les matrices de projection décrites dans les fichiers** *calib*.lmv.ie1* **correspondant aux images** *mire0_g.gif* **et** *mire0_d.gif***.** On cherchera par ce programme à calculer la distance entre les centres optiques des deux caméras puis à comparer avec la réalité.

2.2 Mises en correspondances et reconstruction 3D

2.2.1 Mises en correspondance sur images non rectifiées

On suppose ici les images non corrigées en distorsion et non rectifiées. Pour apparier des pixels entre les deux images, on désire programmer le score de corrélation "ZNCC" noté $score_{zncc}$. **Compléter le programme** *match1.m*. La démarche est la suivante : à partir de trois points sélectionnés dans l'image *obj1_g.jpg* et leurs correspondants approximatifs dans l'image *obj1_d.jpg*, on cherchera la position précise de ces derniers en maximisant le score $score_{zncc}$ sur une fenêtre de 11×11 pixels.

2.2.2 Mises en correspondance sur images rectifiées et reconstruction 3D

On suppose ici les images corrigées en distorsion et rectifiées, soient *obj1_g3.jpg* et *obj1_d3.jpg*.

1. **Compléter le programme** *match2.m* **afin de trouver les correspondants dans l'image de droite de deux points préalablement sélectionnés dans l'image de gauche.**
2. Le temps de calcul étant assez prohibitif, **proposer sans le programmer un nouvel algorithme pour accelérer le processus.**
3. **Compléter le programme afin de déduire les coordonnées 3D des deux couples de points appariés. En déduire la distance euclidienne entre ces deux points 3D et vérifier.**

2.2.3 Précision des mesures 3D

Pour quantifier les erreurs de reconstruction, on se propose de reconstruire des objets plans texturés, de calculer alors le plan moyen passant au mieux par les points 3D obtenus puis d'en déduire les erreurs de reprojection sur ce plan. Cette démarche permet d'évaluer la reconstruction et donc indirectement le calibrage.

A l'aide du fichier de points 3D *points.3d* correspondant à la reconstruction d'un objet plan, **compléter le programme** *precision3D.m* **qui permet de calculer les coefficients d'un plan par les moindres carrés. Quel est l'ordre de grandeur obtenu ?**

References

[1] Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/htmls.

[2] V. Garric. *Vision pour la Robotique de manipulation : calibration, localisation et saisie d'objets*. PhD thesis, LAAS-CNRS, Mai 1996.

[3] P. Lasserre. *Vision en Robotique en environnement naturel*. PhD thesis, LAAS-CNRS, Septembre 1996.

[4] M. Minoux. *Programmation mathématique*. 1990.

[5] N.Ayache. *Vision stéréoscopique et perception multi-sensorielle*. Inter Editions, 1989.

[6] R.Horaud and O.Monga. *Vision par ordinateur : outils fondamentaux*. Editions Hermès, 1993.