TP de vision par ordinateur: Etalonnage géométrique de caméra et reconstruction 3D d'un objet polyédrique

B. Vandeportaële

1 Présentation du TP

Le but de ce TP est de réaliser une acquisition 3D d'un objet en utilisant la vision par ordinateur. Ce TP se déroule sur deux séances de quatre heures.
Dans la première partie du TP, vous travaillerez sur des images fournies et qui ont été acquises dans de bonnes conditions. Plus tard, vous acquerrez vos propres images et verrez alors des conditions critiques.
La boite à outils d'étalonnage de caméra : "Calibration Camera Toolbox for matlab" sera utilisée afin d'estimer les paramètres internes du modèle de caméra ainsi que la pose de la caméra sur deux images dans lesquelles l'objet à acquérir en 3D est visible. Le modèle de caméra utilisé dans ce TP est une caméra trou d'épingle à laquelle s'ajoute les effets de distorsions radiales et tangentielles (se reporter au cours pour la définition de ce modèle, les notations peuvent varier légèrement).
Une fois la caméra étalonnée, vous devrez réaliser la triangulation des points de l'objet et utiliser la géométrie épipolaire. Nous verrons aussi comment reconstruire les points vus sur une unique image en utilisant une connaissance sur la scène observée.
Finalement, vous utiliserez les homographies pour reprojeter les textures de l'objet sur son modèle.

2 Rappels de cours : Le modèle de caméra trou d'épingle

Le trou d'épingle (pinhole) est utilisé pour modéliser simplement les caméras. Modéliser géométriquement une caméra, c'est être capable de prévoir la projection dans le plan image d'un point dans le repère de travail (modèle direct) et inversement d'associer une droite du repère de travail à une position pixellique dans l'image (modèle inverse).

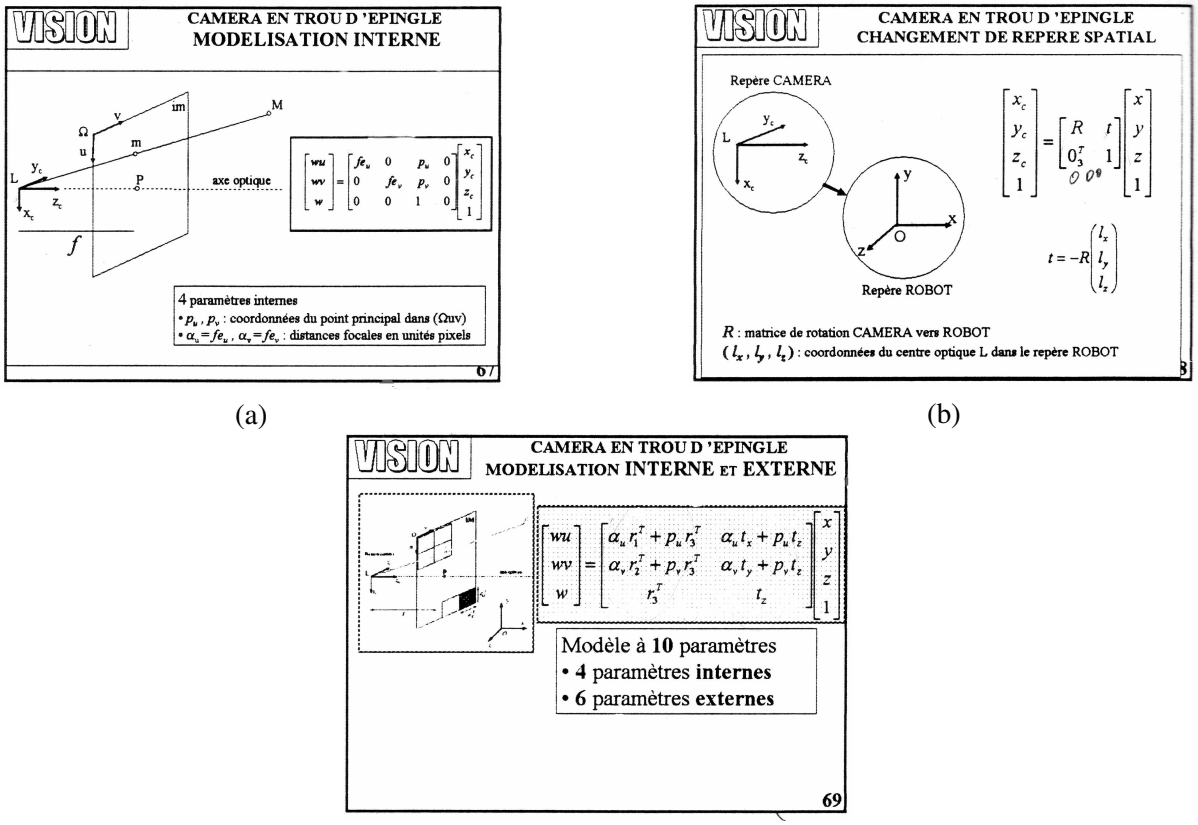
Nous développons ici les calculs pour le modèle présenté en cours (rappel Fig. 1). (Pour info, le modèle utilisé dans le TP est plus complexe puisqu'il est possible de gérer la non orthogonalité des lignes et colonnes du plan image ainsi que des distorsions optiques).

2.1 Rappels sur les coordonnées homogènes

Le point 2D $(u_i\ v_i)^T$ peut être représenté en coordonnées homogènes par $(u_i\ v_i\ 1)^T$. Il peut aussi être représenté par n'importe quel triplet $(fu_i\ fv_i\ f)^T$ pour $(f \neq 0)$.
Les coordonnées homogènes sont aussi utilisées pour représenter les directions, grâce aux **points à l'infini**. La direction $(u_i\ v_i)^T$ est représentée en coordonnées homogènes par $(u_i\ v_i\ 0)^T$. Elle peut aussi être représentée par n'importe quel triplet $(fu_i\ fv_i\ 0)^T$.
L'utilisation des coordonnées homogènes est évidemment possible en 3D mais avec 4 coordonnées cette fois-ci.

2.2 Paramètres intrinsèques et extrinsèques

Le modèle de la caméra intègre des paramètres dits **intrinsèques** qui sont indépendants de la position et de l'orientation de la caméra dans la scène (Fig. 1 (a)). Les paramètres **extrinsèques** (Fig. 1 (b)) permettent de modéliser le placement et l'orientation du **repère caméra** par rapport à un **repère de travail** ou repère robot.



(a) (b)

(c)

FIGURE 1 – Le modèle pinhole : géométrie et matrice.

2.3 Calcul de l'image d'un point

L'image $m = (wu\ wv\ w)^T$ d'un point $M = (x\ y\ z\ 1)^T$ exprimé dans le repère de travail est obtenue à partir de la matrice caméra $\tilde{C} = \begin{pmatrix} C & \tilde{c}_4 \end{pmatrix}$ (voir Fig. 1 (c)).

$$\tilde{C} = \begin{bmatrix} \alpha_u\, r_1^T + p_u\, r_3^T & \alpha_u\, t_x + p_u\, t_z \\ \alpha_v\, r_2^T + p_v\, r_3^T & \alpha_v\, t_y + p_v\, t_z \\ r_3^T & t_z \end{bmatrix} \ ;\ m = \tilde{C}\, M$$

R est orthonormale, donc C est inversible si $\alpha_u \neq 0$ et $\alpha_v \neq 0$.

2.4 Calcul de la position du centre optique

La position du centre optique L de la caméra dans le repère de travail est obtenue à partir de la matrice caméra $\tilde{C} = \begin{pmatrix} C & \tilde{c}_4 \end{pmatrix}$ en observant que ce point se projette en un point à l'infini dans la direction $(0,0)^T$ de la façon suivante :

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = (C \quad \tilde{c}_4) \begin{pmatrix} L \\ 1 \end{pmatrix} \quad \rightarrow \quad L = -C^{-1}\, \tilde{c}_4$$

2.5 Calcul du rayon de projection

L'ensemble des points M de la demi-droite (dans le repère de travail) associés à un point m dans le plan image est donnée par : $M = L + \lambda \overrightarrow{L\,m}$, λ étant un paramètre représentant la distance du point 3D au point L.
Soit $\tilde{\Omega}$ le point à l'infini dans la direction de $\overrightarrow{L\,m}$: $\tilde{\Omega} = \begin{pmatrix} \Omega \\ 0 \end{pmatrix}$, $m = C.\Omega$ et donc $\Omega = C^{-1}\ m$
On obtient : $M = -C^{-1}\ \tilde{c}_4 + \lambda\, C^{-1}\ m$ qui donne finalement : $M = C^{-1}\ (-\tilde{c}_4 + \lambda\, m)$
En fonction du facteur multiplicatif associé à la représentation homogène de m, $\lambda > 0$ ou $\lambda < 0$.

2.6 Méthode d'étalonnage "Direct Linear Transformation" (DLT)

Cette méthode permet d'estimer (simplement et directement, d'où son nom) une matrice $\tilde{C}$ à partir de l'observation des projections sur l'image de la caméra de points 3D dont la position spatiale est connue. Cet ensemble de points 3D constitue ce que l'on appelle une **mire d'étalonnage**.
Chaque point 3D M_i observé fournit une équation de type : $\tilde{C}\ M_i = m_i$ qui se développe en :

$$\begin{pmatrix} C_1^T & c_{14} \\ C_2^T & c_{24} \\ C_3^T & c_{34} \end{pmatrix} M_i = \begin{pmatrix} w_i u_i \\ w_i v_i \\ w_i \end{pmatrix}$$

En passant des coordonnées homogènes M_i aux coordonnées réelles $\breve{M}_i$, on obtient :

$$u_i = \frac{C_1^T\ \breve{M}_i + c_{14}}{C_3^T\ \breve{M}_i + c_{34}} \text{ qui se développe en } C_1^T\ \breve{M}_i - u_i\ C_3^T\ \breve{M}_i + c_{14} - u_i\ c_{34} = 0$$
$$\text{et } v_i = \frac{C_2^T\ \breve{M}_i + c_{14}}{C_3^T\ \breve{M}_i + c_{34}} \text{ qui se développe en } C_2^T\ \breve{M}_i - v_i\ C_3^T\ \breve{M}_i + c_{24} - v_i\ c_{34} = 0$$

NB : Ce sont en fait deux équations de plans.
Avec un minimum de six points, on obtient les douze équations nécessaires à l'estimation de la matrice caméra $\tilde{C}$. Il faut pour ceci organiser les données dans une matrice A pour avoir un système linéaire de la forme : $A\ c_{[ij]} = 0$ avec les différents coefficients de la matrice $\tilde{C}$ organisés dans le vecteur $c_{[ij]} = \begin{pmatrix} c_{11} \\ ... \\ c_{34} \end{pmatrix}$
NB : $\tilde{C}$ est définie à un facteur d'échelle près et possède 11 degrés de liberté, donc en réalité, le système d'équation est sur-contraint pour six points. Par contre, cinq points n'auraient pas contraint suffisamment le système.
Pour éviter la solution triviale $\tilde{C} = 0$, on peut imposer des contraintes. Nous utiliserons la méthode SVD pour résoudre le système d'équations $A\ c_{[ij]} = 0$. La SVD est une décomposition matricielle en valeurs singulières de A : $A_{(k.l)} = U_{(k.k)}\ D_{(k.l)}\ V^T_{(l.l)}$. La solution d'un système $A\ X = 0$ au sens des moindres carrés est la colonne de V (vecteur propre) correspondant à la plus petite valeur propre dans D. La solution est telle que $\|c_{[ij]}\| = 1$.
En résolvant le système linéaire formé par les données de l'étalonnage, on obtient une matrice $\tilde{C}$ qui n'est pas forcément décomposable en la forme vue sur la Fig. 1 (c). On effectue donc une décomposition de $\tilde{C}$ pour retrouver les paramètres $(\alpha_u, \alpha_v, p_u, p_v, r_1, r_2, r_3, t_x, t_y, t_z)$ qui approximent au mieux $\tilde{C}$. (Dans notre exemple, la matrice $\tilde{C}$ calculée en résolvant le système linéaire possède onze degrés de liberté alors que la paramétrisation par les paramètres intrinsèques et extrinsèques proposée n'en possède que dix.)

2.7 Raffinage par optimisation non linéaire

Il n'existe pas, en général, de matrice $\tilde{C}$ permettant de "faire coller parfaitement le modèle aux données" pour plus de cinq points. On recherche donc le meilleur jeu de paramètres afin de minimiser une erreur quadratique : L'erreur de reprojection (exprimée en unité pixel) exprime pour un point M_i la distance géométrique dans l'image entre le point $\hat{m}_i$ obtenu par le modèle paramétrique et le point m_i réellement observé dans l'image.
Ayant à disposition $k/2$ observations de points, la matrice $\tilde{C}$ optimale est celle qui minimise le critère $\sum_{i=0}^{k/2} d(\hat{m}_i, m_i)^2$

Pour minimiser ce critère en considérant une distance géométrique, une méthode linéaire telle que le DLT n'est pas suffisante et il faut utiliser des outils d'optimisation non linéaire. Partant de la solution fournie par le DLT, l'optimisation non linéaire permet de raffiner itérativement le jeu de paramètres afin de minimiser le critère.

2.8 Un modèle plus complet pour la caméra

En pratique, le modèle trou d'épingle est souvent insuffisant pour modéliser "correctement" (cela dépend de l'application) une caméra. Des modèles plus complexes intégrant par exemple les distorsions optiques radiales et tangentielles ont donc été proposés. Ces distorsions correspondent à une modification du plan image et peuvent être facilement ajoutées au modèle trou d'épingle déjà étudié. Il est possible de supposer les distorsions nulles dans un premier temps afin d'estimer les paramètres du modèle par DLT puis d'estimer l'intégralité des paramètres du modèle lors de l'optimisation non linéaire.
Les effets des distorsions sont souvent représentés par des images déformées en barillet ou coussinet. En pratique, comme ces distorsions s'appliquent uniquement au niveau du plan image, il est possible de **rectifier** les images afin d'obtenir les images dans le plan image d'une caméra trou d'épingle sans distorsion.

2.9 Mire 3D ou plane(s) ?

Supposons que tous les M_i soient disposés dans un unique plan. Le système linéaire à résoudre par DLT est alors dit **mal conditionné**. Cela signifie qu'il n'est pas possible d'estimer les paramètres du modèle dans cette configuration. Il faut donc utiliser un ensemble de points n'appartenant pas à un même plan. En pratique, une telle mire est assez difficile à réaliser précisément et l'on préfère utiliser **plusieurs mires planes** (ou une mire plane observée dans différentes configurations).

3 La "Calibration Camera Toolbox for matlab"

Dans ce TP, nous utilisons un outils existant pour estimer les paramètres du modèle de caméra.

3.1 Récupération du fichier

Récupérer le fichier http ://bvdp.free.fr/enseignements/TP_STEREO_2007/TOOLBOX_calib_TP2007.zip et le décompresser dans le répertoire d :/tpcalib que vous aurez préalablement créé sur le disque de votre machine.

3.2 Utilisation des fonctions d'étalonnage

Cette boîte à outils permet d'estimer les paramètres intrinsèques (constant entre chaque image) et extrinsèques (individuellement pour chaque image). Pour cette tâche, une mire en forme de damier est utilisée. Les positions 3D des points d'étalonnage répartis sur cette mire sont connues et leurs images peuvent être détectées très précisément (avec une précision dite **sub-pixellique**).
Vous allez maintenant réaliser l'étalonnage à partir d'un ensemble d'images de la mire :
– Lancer matlab et aller dans le répertoire d :/tpcalib.
– Lancer le script **calib_gui**.
– Cliquer sur mode standard
– Cliquer sur Image names
– Taper ima
– Choisir jpg en tapant j
Il faut maintenant, sur chaque image, informer la toolbox de la position grossière de la mire dans chacune des images.
– Cliquer sur 'extract grid corner'
– Taper 'entree' pour toutes les images
– Taille de la fenêtre de recherche 11*11 -> wintx=5 et winty=5
– Choisir le mode d'extraction du nombre de carré automatique

Ensuite, sur chaque image, cliquer sur les 4 extrémités du damier dans l'ordre 1,2,3,4. L'ordre est important car la mire définit le repère de travail. Voir la définition de ce repère par rapport à la mire sur la figure 2. L'ordre de selection des points doit être respecté au moins dans les deux premières images. Pour les images où les quatres points ne sont pas visibles, sélectionner quatre coins d'une mire rectangulaire plus petite. Ne pas sélectionner des points trop proches du bord de l'image (20 pixels).

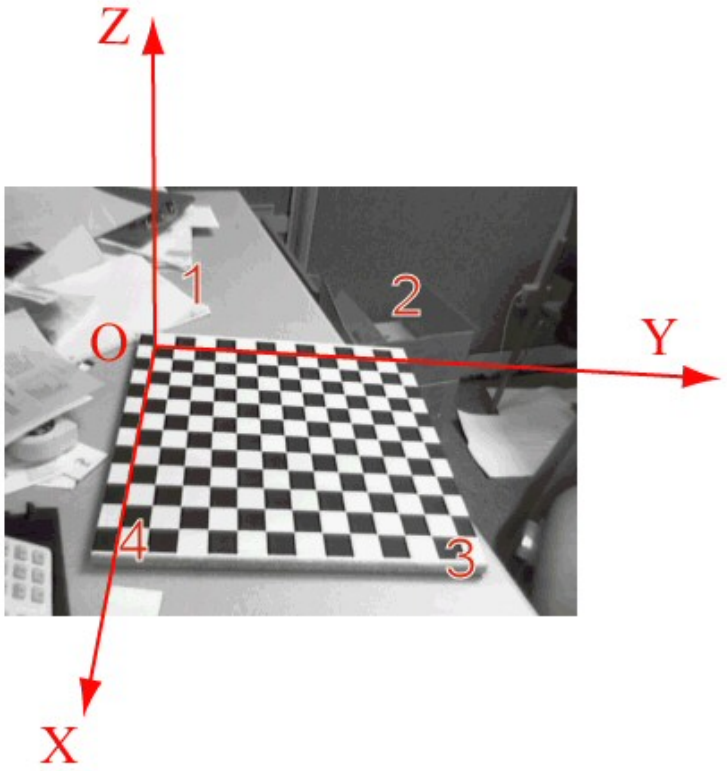


FIGURE 2 – Définition du repère par rapport à la mire

– Définir la taille en mm des carrés en x et y (ils font 30mm sur les images fournies)
– Répondre no pour "Guess for initial distortion" car notre caméra présente peu de distorsion et le programme arrivera à extraire les points d'étalonnage facilement.

La toolbox est capable, à partir des positions pixelliques des 4 coins de la mire, de trouver les positions subpixelliques de tous les points d'étalonnage de la mire. Le détecteur utilisé ici est supposé avoir une précision de l'ordre de 0.1 pixel.

Une fois toutes les images renseignées, cliquer sur calibration. La toolbox affiche les résultats après initialisation (estimation modèle linéaire) et après descente du gradient (estimation du modèle par optimisation non linéaire).

A FAIRE :

– Interpréter les résultats (les noms des paramètres sont différents ; faire le lien).
– Cliquer sur show extrinsic : Une figure s'affiche dans laquelle on peut voir soit les poses des damiers par rapport à une caméra fixe, soit les positions de la caméra par rapport à un damier fixe.
– Cliquer sur reproject on images : Ceci permet de comparer le modèle obtenu par l'étalonnage avec la caméra réelle. 4 informations sont affichées sur chaque image :
  1. le repère de la mire
  2. les points extraits de l'image représentés par des croix (les m_i)
  3. les points reprojetés grâce au modèle représentés par des ronds (les $\hat{m}_i$)
  4. des flèches qui représentent l'erreur de reprojection pour chaque point, en amplitude et direction

– Cliquer sur Analyse error : La fenêtre error représente la distribution des erreurs de reprojection. Elle permet de se rendre compte d'éventuels biais ou groupe de points posant problèmes. Elle montre aussi l'erreur max. La figure 3 montre un exemple de résultats.

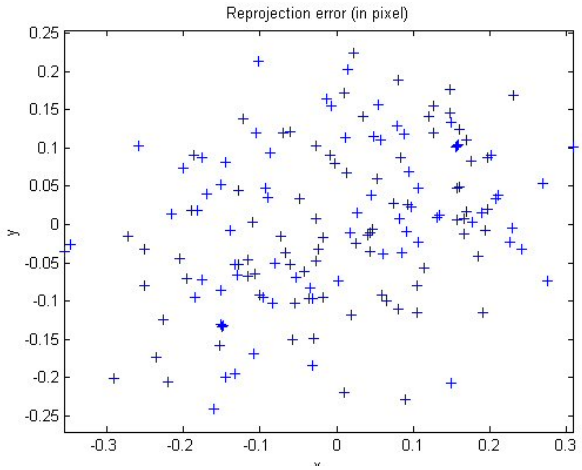


FIGURE 3 – Exemple d'affichage des erreurs de reprojection

En cliquant sur un point, les informations correspondantes s'affichent dans la console matlab. Exemple :

```
0  Selected image: 8 Selected point index: 6
1  Pattern coordinates (in units of (dX,dY)): (X,Y)=(5,12)
2  Image coordinates (in pixel): (588.87,289.50)
3  Pixel error = (0.57961,0.32108)
4  Window size: (wintx,winty) = (5,5)
```

Si l'on détecte qu'une image a beaucoup de points présentant des erreurs importantes, cela signifie probablement que cette image n'a pas été bien traitée (problème lors de l'extraction des points, flou....). Il est donc nécessaire de supprimer ces images pour qu'elles ne perturbent pas l'estimation des paramètres de la caméra.

NB : Si l'algorithme était capable d'estimer automatiquement que cette image contenait des données erronées, on parlerait d'**estimation robuste**. A défaut, on doit renseigner les données erronées à la main...

– Pour réaliser ceci cliquer sur Add/Suppress images et taper le numéro de l'image à supprimer.
– Relancer ensuite l'étalonnage en cliquant sur calibration.
– **!!!!IMPORTANT!!!!** Cliquer sur Save une fois que l'étalonnage est bon. Ceci permettra de ne pas devoir reprendre la procédure du début en cas de problème avec Matlab. Les données seront récupérables en tapant : *load Calib_Results*.

```
0  Saving calibration results under Calib\_Results.mat
1  Generating the matlab script file Calib\_Results.m containing
2  the intrinsic and extrinsic parameters...
```

– Lancer le script **check_convergence.m** qui permet de visualiser l'animation correspondant à la convergence des paramètres du modèle lors de la descente du gradient.

4 Triangulation et géométrie épipolaire

4.1 Rectification des images

Dans un premier temps il est nécessaire de rectifier les 2 images de l'objet afin de créer des images qui auraient été obtenues avec un capteur trou d'épingle et donc s'affranchir des distorsions optiques.

– Cliquer sur undistord images
– Taper 1 pour rectifier une image différente de celles utilisées pour l'étalonnage
– Entrer le nom imaobj1 puis j pour jpg
– La toolbox crée l'image imaobj1_rect.jpg
– Suivre la même procédure pour imaobj2.jpg

– Comparer les images avant et après rectification.

Tous les traitements suivants seront réalisés sur les images rectifiées !

Les scripts utilisés dans la suite du TP ne proviennent pas de la toolbox mais sont placées dans le même répertoire pour des raisons pratiques.

4.2 Calcul des matrices caméra

– lancer le script **calculer_cg_et_cd.m**. Ce script va créer les matrices $\tilde{C}_1$ et $\tilde{C}_2$ telles que définies dans le cours à partir des résultats de l'étalonnage. Regardez comment est fait ce script.

4.3 Interaction pour informer le programme des points et faces de l'objet

– Lancer le script **interaction.m**. L'image 1 rectifiée de l'objet s'affiche.
– Cliquer sur les pixels correspondant aux coins en mémorisant l'ordre des points.
– Taper Entrée une fois tous les points saisis. Une deuxième fenêtre s'ouvre, montrant l'image 2.
– Cliquer sur les pixels homologues aux pixels que vous venez de sélectionner, dans le même ordre.
– Une fois tous les points saisis, il faut décrire les facettes de l'objet en reliant ses points. Utiliser la syntaxe suivante : [1 2 3 4] pour créer une face reliant les points 1, 2, 3 et 4. Taper Entree pour valider. Pour terminer la liste de facettes, entrer []. **Créer uniquement des faces définies par 4 points, car on aura besoin de ces 4 points dans la suite du TP pour la reprojection des textures !**
  Le tableau points_2d est créé et rempli par le script, ainsi qu'une liste de faces dans le tableau faces.
  Les 2 premières colonnes du tableau points_2d contiennent les coordonnées x et y du point dans l'image 1 et les colonnes 3 et 4 celles dans l'image 2.
– **SAUVER les tableaux points_2d et faces ! ! !**
  `save points_2d.mat points_2d` et `save faces.mat faces`

4.4 Triangulation des points par stéréoscopie

– Compléter le script **calcul_points_3d.m**. Ce script doit trianguler les points en fonction de leurs projections sur les 2 images et des paramètres de la caméra. La solution trouvée pour chaque point est la position qui minimise la somme des écarts au carré aux 4 plans correspondant aux 2 rayons de projections associés aux 2 points m_i des images. Bien qu'il n'y ait pas réellement intersection des rayons de projection, il est ainsi possible de calculer une estimée de la position 3D du point.

A partir des deux matrices caméra, on peut remonter aux coordonnées 3D d'un point vu sur les deux images :

– les équation cartésiennes du rayon de projection peuvent s'écrire en développant les équations vues dans le chapitre sur le modèle trou d'épingle :

$$\begin{cases} (c_{11} - u_i\, c_{31})x + (c_{12} - u_i\, c_{32})y + (c_{13} - u_i\, c_{33})z + c_{14} - u_i\, c_{34} = 0 \\ (c_{21} - v_i\, c_{31})x + (c_{22} - v_i\, c_{32})y + (c_{23} - v_i\, c_{33})z + c_{24} - v_i\, c_{34} = 0 \end{cases}$$

– le calcul du point M, antécédent du couple de points homologues (m_1, m_2), s'obtient en résolvant au sens des moindres carrés linéaires le système linéaire de 4 équations à 3 inconnues (x, y, z) que l'on obtient en écrivant les équations cartésiennes simultanées des rayons de projection m_1L_1 et m_2L_2.

– Mettre les équations sous la forme d'un système $A\,X = B$ que l'on résoudra ensuite en calculant la pseudo inverse ($X = \left(A^T\,A\right)^{-1}\,A^T\,B$).

Le tableau points_3d est créé et rempli par le script, analyser les résultats.

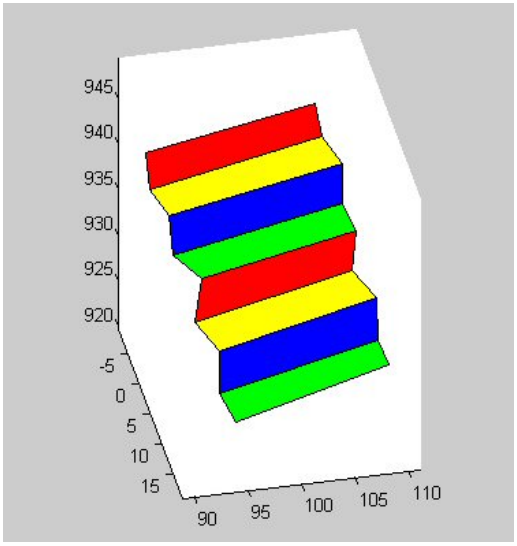
-**SAUVER le tableau points_3d ! ! !** `save points_3d.mat points_3d`

4.5 Triangulation des points par monoscopie

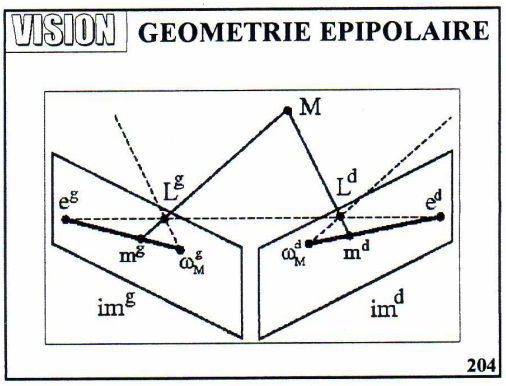
Proposer une méthode pour acquérir le modèle 3D de l'objet d'exemple à partir d'une seule image. Et oui, c'est possible pour cet objet... Indice : Les points de l'objets sont situés dans deux plans.

4.6 Affichage 3D

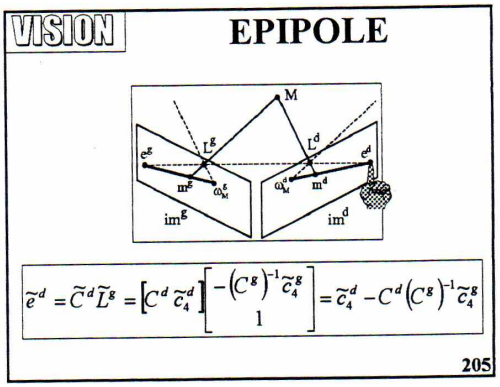
– Lancer le script **recons_3d.m** pour afficher une représentation 3D de l'objet. Vous pouvez le faire tourner à l'aide de la souris (Voir Fig. 4 (a).)



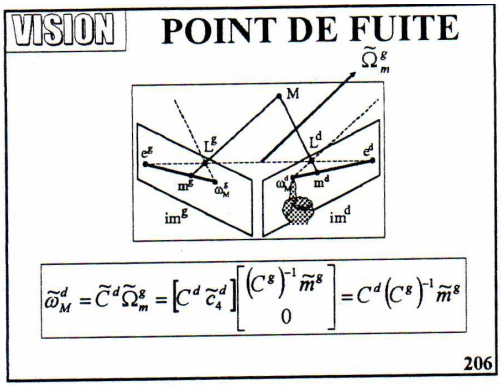
(a)



(b)



(c)



(d)

FIGURE 4 – Exemple d'affichage 3D et rappels de géométrie épipolaire

4.7 L'écart intraoculaire

Compléter la fonction **calcBaseline.m** pour qu'elle retourne la distance entre les centres optiques des caméras pour les deux images (les matrices caméras sont fournies $\tilde{C}_1$ et $\tilde{C}_2$).

4.8 Géométrie épipolaire

La position de l'homologue d'un point de l'image 1 dans l'image 2 est contrainte par ce qu'on appelle la géométrie épipolaire. Cette contrainte implique que l'homologue d'un point doit être situé sur un **segment épipolaire**. Dans cette partie du TP, vous allez coder une fonction permettant d'afficher ces segments dans l'image 2 pour les points renseignés dans l'image 1.

Vous allez maintenant éditer la fonction **calc_epipole.m** pour lui faire afficher un segment [e2 w2] dans l'image 2 en fonction d'un point dans l'image 1 m_1 et des matrices $\tilde{C}_1$ et $\tilde{C}_2$.

Soient $\tilde{C}_1$ et $\tilde{C}_2$ les matrices de projection des caméras 1 et 2, le point 3D M est situé sur une droite reliant le centre optique de la caméra 1 L_1 à la projection du point 3D sur le capteur 1 $m_1 = \tilde{C}_1 \, M$. Soit Ω_1, un point à l'infini dans la direction m_1 vers L_1. On peut donc conclure que M appartient à [L_1 Ω_1).

Vue par la caméra 2, cette propriété va nous permettre de contraindre la projection de M. La projection per-

spective d'une droite est une droite (ou éventuellement un point dans le cas particulier où la droite contient le centre optique), $[L_1\ \Omega_1)$ va donc se projeter en un segment de droite au niveau de l'image de la caméra 2. Cherchons 2 points de cette droite :

L'un d'eux, appelé l'épipole, est la projection de L_1 par $\tilde{C}_2$, nous le noterons $e_2 = \tilde{C}_2\ L_1$.

L'autre est la projection de Ω_1 par $\tilde{C}_2$, nous le noterons $w_2 = \tilde{C}_2\ \Omega_1$. Nous obtenons donc un segment épipolaire $[e_2\ w_2]$ sur lequel doit se trouver le point homologue à m_1.
NB :
- Les coordonnées réelles d'un point 2D représenté par $(u, v, w)^T$ sont $(u/w, v/w)^T$.
- Tous les segments épipolaires dans l'image 2 passent par e_2, quelque soit le point m_1, on a donc un faisceau de segments épipolaires.
- L'affichage du segment [a,b] est réalisé par matlab dans la figure en cours en appelant :
  line([ax ;bx],[ay ;by],'Color','r','LineWidth',1)
  De plus, dans cet exemple, nous indiquons la couleur et l'épaisseur du segment.

Vous pouvez tester votre fonction en entrant les coordonnées d'un point connu dans l'image 1 et en vérifiant que le segment épipolaire passe bien par son homologue dans l'image 2. Vous pouvez aussi utiliser le script **testcal_epipole.m** qui trace les segments épipolaires dans l'image en cours pour tous les points_2d de l'image 1.

4.9 Validation de la Géométrie épipolaire

- Relancer le script interaction. Lorsque vous serez invités à rentrer les points sur l'image 2, vous devriez voir apparaître pour chaque point un segment proche de l'homologue.

5 Estimation d'homographie

Depuis le début du TP, nous avons travaillé sur l'étalonnage et l'utilisation de la caméra trou d'épingle, dont le modèle projectif permet de passer de l'espace au plan. Il existe un modèle permettant de représenter les relations projectives existantes entre deux plans : l'homographie. Elle est représentée sous la forme d'une matrice de dimension 3.3 :

$$\begin{pmatrix} w\,X1 \\ w\,Y1 \\ w \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \begin{pmatrix} X2 \\ Y2 \\ 1 \end{pmatrix}$$

Cette matrice est définie à un facteur multiplicatif près (comme la matrice $\tilde{C}$) et possède donc 8 degrés de liberté. En développant cette équation comme pour le cas de la matrice caméra (avec le DLT), montrez que 4 correspondances de points suffisent pour estimer l'homographie.
Vous avez déjà utilisé les homographies pour calculer des **mosaïques d'images** dans le TP de Traitement et Synthèse d'Image (reconstruction de mosaïque). Vous aviez alors estimé l'homographie existant entre des points de deux images acquises depuis la même position. L'une des images était alors reprojetée sur le plan correspondant à l'autre image. Nous proposons ici une autre application : rectifier un DataMatrix, qui est un code barre bidimensionnel. Cette opération pourra vous être notamment utile dans le cadre du projet de traitement d'image de Michel Cattoen et nous montrerons plus loin comment elle permet de calculer et de reprojeter la texture d'un objet à facettes planes.
Nous souhaitons ici générer une image dans laquelle la lecture des bits du code barre peut être effectuée directement à partir d'une image que nous appellerons 'rectifiée'. Cette image sera calculée à partir d'une image acquise avec la caméra dans une position et orientation quelconque par rapport au code barre. La figure 5 montre une image acquise par la caméra et sa version rectifiée.
A FAIRE :
-Compléter le script **Homography.m** pour calculer les coefficients de la matrice H à partir des coordonnées des 4 coins du DataMatrix dans l'image test1.bmp ($X11, Y11$ à $X14, Y14$) calculé d'après les positions cliquées à l'aide de la souris) et des 4 coins du DataMatrix dans l'image rectifiée ($X21, Y21$ à $X24, Y24$)

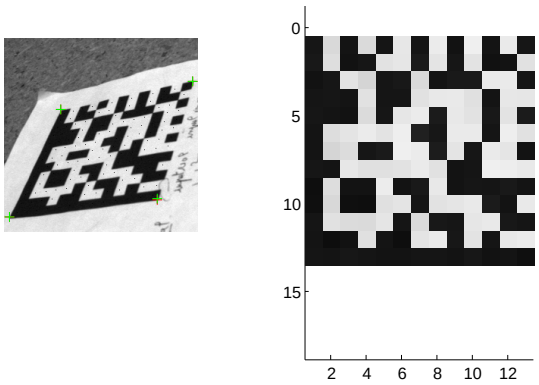


FIGURE 5 – Exemple d'homographie permettant de décoder un DataMatrix

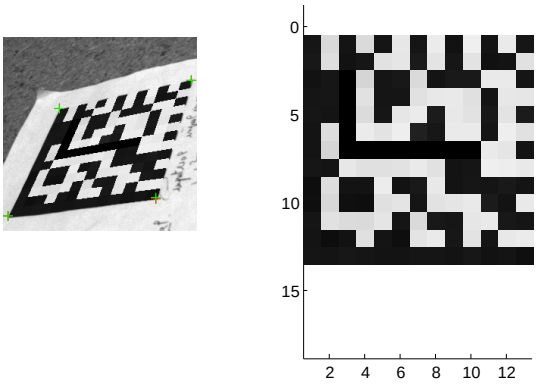


FIGURE 6 – Modification du DataMatrix et augmentation de l'image observée par la caméra

calculée d'après la taille du DataMatrix et le nombre de pixels par point du DataMatrix souhaité (commencer avec 1 pixel par point du DataMatrix).
-Regarder le script **CalculeImageHomographie.m** et interpréter son fonctionnement.

5.1 Modification du DataMatrix et augmentation de l'image d'origine

Une fois l'image du DataMatrix rectifié obtenue, il est très facile de la modifier et donc de modifier le code correspondant. Nous souhaitons maintenant actualiser l'image observée par la caméra avec le nouveau code. Il s'agit de **réalité augmentée** puisqu'une image d'une scène réelle est modifiée par des données synthétiques (Voir Fig. 6). Il faut donc maintenant calculer les positions dans l'image du Datamatrix rectifiée pour chaque pixel de l'image observée par la caméra. Elles sont obtenues grâce à une autre homographie :

$$\begin{pmatrix} w\,X2 \\ w\,Y2 \\ w \end{pmatrix} = \begin{pmatrix} H'_{11} & H'_{12} & H'_{13} \\ H'_{21} & H'_{22} & H'_{23} \\ H'_{31} & H'_{32} & H'_{33} \end{pmatrix} \begin{pmatrix} X1 \\ Y1 \\ 1 \end{pmatrix}$$

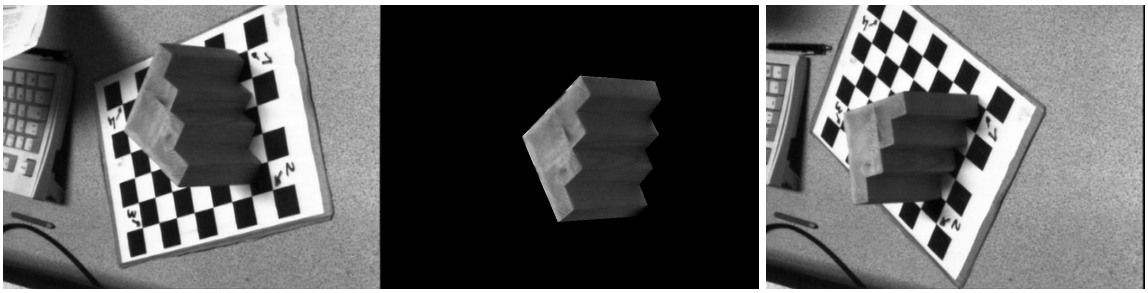
A FAIRE :
-Compléter le script **Homography.m** pour modifier le DataMatrix (modifier l'image imout).
-Déterminer l'homographie inverse (réfléchir un peu... la solution est dans la question...).
-Mettre à jour l'image de la caméra et l'afficher.

6 Reprojection des textures sur l'objet

Nous allons maintenant utiliser tout ce qui a été étudié précédemment pour générer des images de la scène 3D observée. Ces images, contrairement à celles générées en 4.6, rendent compte de la texture de l'objet grâce à des homographies.

6.1 Une vue intermédiaire

L'objectif est de générer l'image que verrait une caméra virtuelle dans une pose intermédiaire entre les deux images acquises. Le script matlab **GenereVueIntermediaire.m** fourni permet de calculer une pose de caméra. Cette pose est paramétrée par la variable *temps*. Cette variable, dont la valeur doit être comprise entre 0 et 1, permet de choisir la pose générée (*temps*=0 correspond à la première image, *temps*=1 à la seconde, *temps*=0.5 génère une pose intermédiaire équidistante (voir Fig. 7)).
A FAIRE :
-Visionner la vidéo de Démonstration :
http ://bvdp.free.fr/enseignements/TP_STEREO_2007/video_tp_calib_cinepack.avi
-Compléter le script **GenereVueIntermediaire.m** pour calculer la texture de la première face de l'objet (générer une image carré de dimension TailleTexture*TailleTexture).



(a) (b) (c)

FIGURE 7 – (a) et (c) Les deux images acquises par la caméra. (b) Une image intermédiaire générée à partir du modèle 3D de l'objet pour *temps*=0.5.

-Afficher la texture
-Boucler pour traiter toutes les faces de l'objet
-Calculer la projection de chacune des faces dans l'image intermédiaire (Les 4 points de chaque facette doivent être projetés par la matrice caméra $\tilde{C}_3$ correspondant à la vue intermédiaire)
-Calculer les homographies entre les images de texture et l'image intermédiaire
-Afficher l'image intermédiaire (Elle devrait ressembler à Fig. 7 (b)).
-Jouer sur les paramètres de caméra : Changer les paramètres intrinsèques et extrinsèques et générer les images correspondantes.

7 A vos images...

Bravo, vous êtes arrivés à la dernière étape du TP, et il faut maintenant tout recommencer, avec vos propres images. Vous pouvez le faire chez vous si vous possédez matlab et une webcam...

7.1 Acquisition d'images pour la stéréovision

Nous rappelons ici le nécessaire pour faire de la stéréovision avec la méthode proposée :

1. Au moins 2 vues de l'objet à acquérir. NB : seule la partie de l'objet visible à la fois sur les 2 images pourra être reconstruite par stéréoscopie.
2. Un ensemble d'images correspondant à la projection de mires sur le plan image de la caméra, afin d'estimer les paramètres du modèle de caméra.
3. L'information de placement de la caméra par rapport à l'objet sur les 2 prises de vues. Cette information pourra être extraite grâce à l'analyse de deux images de la mire.

A FAIRE : Réaliser l'acquisition d'une séquence d'images pour reconstruire un objet. La caméra restant fixe, vous devrez donc déplacer la mire et l'objet. l'écart entre les positions L_1 et L_2 (positions du centre optique de la caméra par rapport au repère de travail dans les images 1 et 2) influe sur la précision de la triangulation. Néanmoins, il ne faut pas le choisir trop grand car il faut que les images correspondantes soient suffisamment similaires pour pouvoir être appariées.

7.2 Acquisition d'images

Sur les machines de la salle de TP où le logiciel WinTV2000 fonctionne, l'acquisition des images se fait de la façon suivante :
– Lancer Traitement d'images>WinTV2000
– Cliquer sur pref>capture> et régler 800*600
– Cliquer sur le bouton clr et mettre saturation à 0 pour avoir des images en noir et blanc
– Presser CTRL-T pour passer en mode plein écran et en revenir
– Régler la caméra (netteté, ouverture, gain) et NE PAS MODIFIER LES REGLAGES DE LA CAMERA

AU COURS DE L'ACQUISITION car ces réglages influent sur la géométrie.
- Ensuite pour chaque image, cliquer sur capture
- Cliquer droit sur la petite image à gauche
- Enregistrer sous fichier et choisir
  - Type> jpg
  - Profondeur de couleur>256 Level Grayscale
  - Type de compression>none
  - Qualité de l'image JPEG 100
  - Nommer l'image sans oublier le .jpg
- Cliquer sur enregistrer

Sur les autres machines, l'acquisition des images se fait de la façon suivante en utilisant le logiciel de Michel CATTOEN :
- Cliquer sur le bouton Image->Acquisition Hauppauge
- Cliquer Règle-> mettre saturation à 0
- régler 24 bits/pixel
- Sélectionner le format JPEG en qualité 100
- donner un nom au fichier image

7.3 Acquisition de la séquence

Réaliser la séquence suivante en respectant bien les noms des images. Soit Pc_i la position de la mire pour une position de caméra L_i donnée.
- Positionner et orienter la caméra. Positionner l'objet sur la mire en Pc_1 et vérifier que l'objet et les 4 coins de la mire sont visibles.
- Retirer l'objet de la mire et capturer une image de la mire dans la position Pc_1 (ima1.jpg)
- Laisser la mire en place en position Pc_1 et placer l'objet à acquérir sur la mire
- Prendre une image de l'objet (imaobj1.jpg)
- Déplacer l'objet et la mire conjointement en Pc_2, de manière à voir la totalité de l'objet et les 4 coins de la mire.
- Prendre une nouvelle image de l'objet (imaobj2.jpg)
- Retirer l'objet et prendre une image de la mire dans la position Pc_2 (ima2.jpg)
- Prendre 8 images de la mire à différentes positions (ima3.jpg à ima10.jpg)

8 Références

- L'excellente Calibration Camera Toolbox for Matlab créée par Jean Yves BOUGUET est disponible sur internet :
  http ://www.vision.caltech.edu/bouguetj/calib_doc/index.html
- Il existe une implémentation C/C++ de cette toolbox dans OpenCV :
  http ://www.intel.com/research/mrl/research/opencv/
- Une séquence d'exemple d'étalonnage :
  http ://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html