

2 Séance n° 2 : Segmentation d'objet 3D

2.1 Objectif

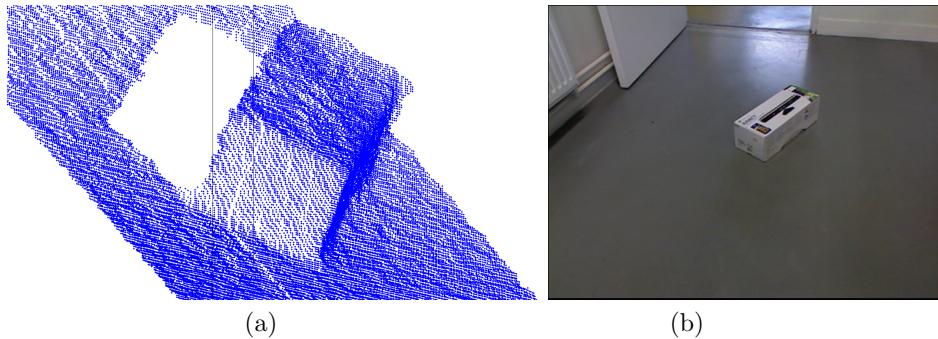


Figure 5: Données kinect : nuage de points 3D (a) et image couleur (b) de la scène.

Dans cette partie, nous allons traiter les données issues du capteur Kinect obtenues lors de la première séance. Le but va être de réaliser un contrôle qualité sur un objet *i.e.* identifier si l'objet est correct ou présente un défaut (trous, résidus,etc.). Dans notre jeu de données, nous allons essayer d'identifier si l'objet présent dans la scène présente des défauts sur sa partie supérieure. Nous proposons de décomposer cette tâche en deux sous-parties :

1. Trouver une méthode pour isoler la surface supérieure de l'objet du reste de la scène,
2. Comparer la surface extraite avec une surface de référence.

2.2 Segmentation

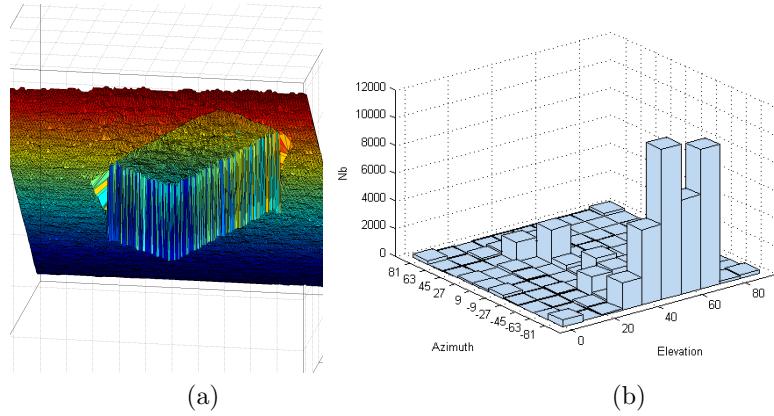


Figure 6: Triangulation de Delaunay (a) et Tableau d'accumulations (b).

1. De quoi est constitué une acquisition Kinect (rôle et composition des fichiers obtenus)? Comment peut-on récupérer les coordonnées d'un point 3D ? Combien de points contient une scène modélisée par Kinect ? Sous *MATLAB*, comment sont stockés l'ensemble de ses points ? Pour limiter les traitements, on se propose de n'étudier qu'une partie de la scène. Compléter le fichier *roi.m* afin de ne conserver et de n'avoir à traiter que l'objet et le plan-support autour de ce dernier (filtrage du bord et du fond de la scène,etc.). Conserver environ 50000 points. Ceci est à réaliser sur les données d'une acquisition d'un objet ne présentant pas de défauts.
2. Sur ces données, nous allons maintenant segmenter l'objet du reste de la scène. Pour cela, on se propose de construire un tableau d'accumulation. Rappeler brièvement son principe.

3. Pour le calcul des normales, et afin d'obtenir la liste des points adjacents à tous points, on se propose d'utiliser la méthode de la Triangulation de Delaunay. Expliquer son principe et l'implémenter dans le fichier *segmentation.m* sous MATLAB. Visualiser le résultatat de la triangulation.

N.B. : on peut regarder l'aide *MATLAB* pour voir comment celle-ci s'implémente.

4. Grâce à la liste des points adjacents, calculer la normale de chaque facette et construire le tableau d'accumulations selon *l'azimuth* et *l'élévation* de chaque normale. Comment est représenté le plan du sol ? Quelles sont les limites de cette approche dans le cas de la détection d'un objet ?
5. Nous allons maintenant isoler la partie supérieure de l'objet. Pour cela, on propose de faire une rotation de la scène puis de ne conserver que la partie supérieure par seuillage. Autour de quel axe la scène doit-elle pivoter ? A l'aide du tableau d'accumulation, trouver automatiquement l'angle de rotation et construire la matrice R . Quel est l'angle et la matrice obtenus ? Effectuer la transformation sur chaque point, puis ne conserver que la partie supérieure de l'objet (environ 6000 à 7000 points).

2.3 Contrôle qualité

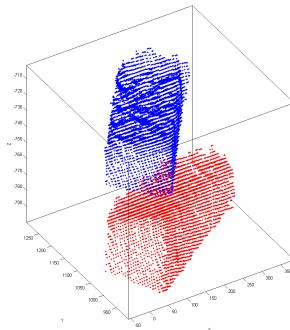


Figure 7: Données en vue de l'utilisation de l'algorithme *Iterative Closest Point* ou *ICP*.

Une fois la surface de l'objet isolée du reste de la scène, nous allons essayer d'y détecter les potentiels défauts présents. Le but étant de trouver une méthode en vue d'automatiser le processus de tri entre pièces correctes et pièces défectueuses. Pour cela, on se propose d'utiliser l'algorithme *Iterative Closest Point* ou *ICP*.

1. Télécharger l'algorithme *ICP* sur le site de *MATLAB* et décrivez comment cet algorithme fonctionne : ses entrées, ses sorties, comment est faite l'optimisation et sur quel critère, ...
2. Les données d'un objet de référence (ou objet modèle) ont été extraites lors de la partie précédente. Nous allons maintenant extraire les données d'un objet à traiter. Compléter le fichier *donnes.m* en s'inspirant de la séquence d'opérations de la partie précédente. Faire en sorte de ne conserver qu'environ 7000 points sur la surface de l'objet à traiter.
3. L'appel à l'*ICP* se fait dans le fichier *metrologie.m*. Avant de réaliser l'appel de l'algorithme, mettre en forme les données. Grâce aux résultats de l'*ICP*, réaliser la transformation sur le jeu de données à traiter, et visualiser dans une même figure les points du modèle et les points des données transformées. Relever les valeurs obtenues et commenter les résultats.
4. Que se passe-t-il si on change le nombre de points des données à traiter ? Tester avec d'autres acquisitions. Commenter les différents résultats. Quels sont les points forts et les faiblesses de cet algorithme ?
5. Trouver une manière de distinguer les pièces correctes des pièces défectueuses en vue d'une automatisation du processus.

3 Séance n° 3 : Localisation 3D d'objet polyédrique

3.1 Objectif



Figure 8: Système de vision Logitech (a), dimensions de l'objet perçu (b).

L'objectif est de localiser en 3D un objet polyédrique par un système mono- ou multi-caméras constitué de caméras Logitech (figure 8-(a)) supposé étalonné au préalable. Le modèle CAO de l'objet perçu est connu *i.e.* on dispose d'un ensemble d'arêtes reliées par des sommets modélisant les dimension 3D de l'objet (figure 8-(b)). La démarche sera alors d'estimer les paramètres de pose 3D de l'objet à partir de mises en correspondance entre primitives modèle (3D) et primitives image (2D). L'étude, formalisation puis implémentation resp. en sections 3.2 et 3.3, sera menée pour une caméra puis généralisée à un système à deux caméras en fin de TP.

3.2 Formalisation sous-jacente



Figure 9: Image gauche notée I_1 (a) et droite notée I_2 (c) du système de vision Logitech.

La figure 9 illustre deux exemples d'images acquises par le banc. On se focalise tout d'abord sur l'image I_1 . Nous supposons connu *a priori* : (i) le modèle CAO de l'objet à saisir, (ii) une estimée initiale grossière, notée par les paramètres extrinsèques $(\alpha, \beta, \gamma^3, T_u, T_v, T_w)$ et la transformation homogène associée $[\mathcal{M}_{oc_1}]_{k=0}$, entre le repère objet \mathcal{R}_o et le repère caméra #1 noté \mathcal{R}_{c_1} .

1. Pour une image acquise \mathcal{I}_1 , la localisation repose sur des appariements 2D/3D entre segments de droite l_i extraits de \mathcal{I}_1 (donc 2D) et segments 3D du modèle CAO de l'objet projetés dans \mathcal{I}_1 . **Expliciter la relation mathématique** entre un segment 3D noté L_i via ses deux extrémités $P_i^o = (X_i, Y_i, Z_i)', i = 1, 2$ et leurs coordonnées image $p_i = (u_i, v_i)', i = 1, 2$ en fonction de $[\mathcal{M}_{oc_1}]_k$ et les paramètres intrinsèques de la caméra.
2. La localisation 3D de l'objet repose sur une résolution numérique et itérative à partir des appariements $\{l_i, L_i\}_{i>4}$. Cette résolution repose sur le critère $F_i = N_i \cdot P_i^{c1}$ avec N_i la normale au plan d'interprétation associé à l_i et $P_i^{c1} = [\mathcal{M}_{oc_1}]_k \cdot P_i^o$ qui requiert à chaque itération k le calcul d'une jacobienne (matrice notée A sur le support de cours) et donc les dérivées de F_i par rapport aux paramètres $(\alpha, \beta, \gamma, T_u, T_v, T_w)$. On se focalise ici sur les dérivées $\frac{\partial F_i}{\partial \alpha}, \frac{\partial F_i}{\partial \beta}, \frac{\partial F_i}{\partial \gamma}, \frac{\partial F_i}{\partial T_u}, \frac{\partial F_i}{\partial T_v}, \frac{\partial F_i}{\partial T_w}$. L'estimation de ces paramètres à l'itération k vise à calculer un incrément par rapport aux paramètres estimés à $k-1$ (on parle alors de localisation par recalages successifs/itératifs) de sorte que on peut approximer $\alpha \sim 0, \beta \sim 0, \gamma \sim 0$. **Montrer avec ces approximations que :**

³ α, β, γ sont les angles d'Euler selon les axes x, y, z du repère.

$$\frac{\partial F_i}{\partial \alpha} = [N_i] \cdot \begin{pmatrix} 0 \\ -Z_i \\ Y_i \end{pmatrix}, \quad \frac{\partial F_i}{\partial \beta} = [N_i] \cdot \begin{pmatrix} Z_i \\ 0 \\ -X_i \end{pmatrix}, \quad \frac{\partial F_i}{\partial \gamma} = [N_i] \cdot \begin{pmatrix} -Y_i \\ X_i \\ 0 \end{pmatrix}, \quad \frac{\partial F_i}{\partial T_u} = N_{i,x}, \quad \frac{\partial F_i}{\partial T_v} = N_{i,y}, \quad \frac{\partial F_i}{\partial T_w} = N_{i,z}$$

On s'appuiera sur les rappels et conventions suivants :

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}, \quad R_\beta = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix}, \quad R_\gamma = \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.3 Implémentation MATLAB

Le but est de compléter le fichier MATLAB *loca.m* (fichier à trous) en répondant aux questions suivantes. Seules les questions (7) et (8) ne donnent pas lieu à implémentation. On note $soluL = (\alpha, \beta, \gamma, T_u, T_v, T_w)_k$ le 6-uplet de pose et $matPos = [\mathcal{M}_{oc_1}]_k$ la matrice homogène associée⁴ à l'itération k . L'animation du modèle 3D et sa projection image sont gérées respectivement par les fonctions *DrawModel(.)* et *Projection(.)*.

1. Les appariements $\{l_i, L_i\}_{i=1,\dots,5}$ sont donnés manuellement donc par simple *click souris*. Prendre en main l'interface afin de sélectionner cinq appariements 2D/3D. Calculer dans le programme principal les normales $[N_i]_{i=1,\dots,5}$ aux plans d'interprétation et remplir les structures de données 3D et 2D resp. *App3d.ini* et *App2d*.
2. Compléter ensuite les fonctions pré-citées *DrawModel(.)* et *Projection(.)*.
3. Compléter alors la fonction *EvalCrit(.)*. Quelle est la valeur du résidu initial *i.e.* pour $soluL$ à $k = 0$?
4. Implémenter le système matriciel à résoudre et notamment la fonction *GlobalDerivative(.)*.
5. Le système estime le vecteur $dsoluL$, mettre à jour la variable $soluL$ en exploitant la fonction *ParamFromTrans(.)*. Relever la valeur du résidu après convergence.
6. On considère maintenant les images issues de la caméra droite, *e.g.* I_2 , ici en configuration stéréo (figure 8-(a)). Cette seconde caméra est ici exploitée pour vérifier. Compléter le programme pour projeter la pose estimée sur l'image I_2 . La matrice de passage $\mathcal{M}_{c_1 c_2} = \{m_{ij}\}_{1 \leq i,j \leq 4}$ entre caméras #1 et #2 est notée *matPos_c1c2* dans le programme.
7. Quel serait l'apport attendu en rajoutant les images issues de cette caméra #2 dans le processus de minimisation ? Reformuler le critère F_i pour considérer dans un et unique processus de localisation propre à \mathcal{R}_{c1} les appariements issus de \mathcal{I}_2 donc de \mathcal{R}_{c2} .
8. Comment automatiser le processus d'appariement ?

⁴Générée par la fonction *TransFromParam(.)*.