TP - Reconstruction de formes 3D

CPE - Julie Digne

16 Octobre 2020

Dans ce TP nous verrons comment passer de plusieurs scans 3D à un objet 3D complet, tout en filtrant les données. En plus de matlab, nous utiliserons pour cela deux logiciels libres.

— *Meshlab*
— *CloudCompare*

En annexe se trouvent quelques commandes utiles de matlab.

1 Prise en main des outils de visualisation

— Téléchargez les données disponibles sur le site. Les afficher dans Meshlab et dans CloudCompare
— Activez ou desactivez la prise en compte des normales.
— Sous CloudCompare, jouez avec l'éclairage (touches F6, F7 ou Display>lights>Toggle Sun Light, Toggle Custom Light). En activant la lumière personnalisée, modifiez la position de l'éclairage (Ctrl+déplacement de la souris).

Qu'observez-vous ?

2 Recalage grossier de cartes de profondeur

Sur les deux scans $set0$ et $set1$, trouvez les coordonnées de points qui s'apparient grossièrement.
Grâce à ces coordonnées, calculez la transformation rigide qui aligne $set1$ *sur* $set0$.
La méthode de calcul de R et t est rappelée en annexe.

3 Recalage de cartes de profondeur : ICP

On considère deux scans en position proches de l'optimum. Nous allons effectuer le recalage par la méthode Iterative Closest Point vue dans le cours. Pour cela nous utiliserons matlab.

L'algorithme Iterative Closest Point itère les étapes suivantes :
— Pour chaque point du nuage de point $\mathcal{P}$, chercher son plus proche voisin dans $\mathcal{Q}$
— Si la distance entre ces deux points est inférieure à δ, ajouter ce couple à l'ensemble des matches
— Calculer la rotation R et la translation T permettant d'aligner les deux ensembles de points
— Effectuer l'alignement $\mathcal{P} \leftarrow R\mathcal{P} + T$ et itérer.

Implémentez l'icp. Il y a 3 parties importantes

— La recherche du plus proche voisin
— Le calcul de la rotation et translation optimale en utilisant la méthode vue en cours et rappelée en annexe du sujet.
— L'application de la transformation à $\mathcal{P}$.

Testez ensuite votre algorithme avec différents paramètres : δ*, le nombre d'itération.*

Calculez l'erreur commise à chaque étape de l'ICP.

4 Reconstruction finale

Dans Meshlab, testez la reconstruction Poisson Reconstruction et la reconstruction Ball Pivoting. Qu'observez-vous ?

5 Débruitage (facultatif)

Implémentez un débruitage par filtre moyenneur

Pour cela pour chaque point p : on cherche son voisinage, on en calcule le barycentre éventuellement pondéré par un poids dépendant de la distance à p (par exemple : $w(q) = \exp -\frac{\|p-q\|^2}{2\sigma^2}$). Le point débruité a pour position la position du barycentre.

Recherche de plus proches voisins et matlab

Pour rechercher des voisins dans un nuage de points, utiliser une approche naive (tester pour chaque point requête tous les points revient à une complexité quadratique. Au lieu de ça on utilise généralement des structures accélératrices comme des kdtree ou des octree. Matlab propose de réaliser cela en effectuant une triangulation de Delaunay préalable à la recherche de voisins. Si A est une matrice $n \times 3$ et B une matrice $m \times 3$, et qu'on cherche les plus proches voisins dans A des points de B, le code matlab est :

```
dt = DelaunayTri(A);
[pid d] = nearestNeighbor(dt,B);
```

pid renvoie les indices dans A des plus proches voisins et d la distance euclidienne.

Annexe : Calcul de R et T optimale, étant donné un ensemble de matches

Supposons que nous avons deux nuages de points $\mathcal{P} = (p_i)_{i=1\cdots n}$ et $\mathcal{Q} = (q_i)_{i=1\cdots n}$ tels que (p_i, q_i) sont appariés. Nous cherchons R, T minimisant :

$$F(R,T) = \sum_{i=1}^{n} \|Rp_i + T - q_i\|^2.$$

Pour cela, l'algorithme expliqué en cours est résumé ci-dessous :

1. Centrer les points $\tilde{p}_i = p_i - \frac{1}{n}\sum_{i=1}^{n} p_i$ et $\tilde{q}_i = q_i - \frac{1}{n}\sum_{i=1}^{n} q_i$.
2. Calculer les matrices P et Q de $\mathbb{R}^{3\times n}$ dont les colonnes sont respectivement $\tilde{p}_i$ et $\tilde{q}_i$
3. Calculer $M = P \cdot Q^T$ et sa décomposition en valeur singulière (fonction svd de matlab) $M = USV^T$
4. En déduire :

$$R = V \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & det(VU^T) \end{pmatrix} U^T$$

5. En déduire

$$T = \frac{1}{n}\sum_{i=1}^{n} q_i - R(\frac{1}{n}\sum_{i=1}^{n} p_i)$$