Segmentation & Registration 3D

I. Introduction

Dans ce TP nous allons traiter des données issues d'un capteur RGB-D RealSense. 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.).

Nous proposons de décomposer cette tâche en deux étapes :

- Trouver une méthode pour isoler la surface visible de l'objet du reste de la scène,
- Comparer la surface extraite avec une surface de référence

II. Segmentation 3D

Nous allons dans un premier temps analyser les données disponibles dans le dossier *data* et les préfiltrer :

1. De quoi est constituée une acquisition RealSense (rôle et composition des fichiers obtenus) ? Combien de points contient une scène modélisée par RealSense ?
2. Utiliser open3d afin de charger et de visualiser la scène.
3. Utiliser la fonction *draw_geometries_with_editing* d'open3d 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). Pour cela il faut appuyer sur la touche *'K'* pour bloquer l'écran et passer en mode sélection. Sélectionner la zone à conserver avec la souris puis appuyer sur *'C'* afin de la sauvegarder.

Sur ces données, nous allons maintenant segmenter l'objet du reste de la scène :

4. Charger le nuage de points à l'aide de la classe *Segmentation* et utiliser la méthode *display* afin de le visualiser. Qu'observe-t-on ? À quoi cela est-il du ?
5. Éliminer les points aberrants en implémentant la fonction d'élimination statistique des 'outliers' dans la méthode *Segmentation.removeOutliers*. Comment fonctionne la détection des points aberrants ? Visualiser le résultat en activant le 'display'. Ajuster les paramètres afin de supprimer tous les points ne faisant pas partie de la scène.
6. Nous allons maintenant calculer les normales du nuage de points. Implémenter les fonctions manquantes dans la méthode *Segmentation.computeNormals* en s'aidant de la documentation d'open3d. Comment fonctionne le calcul de ces normales ? À quoi servent les étapes d'alignement et de normalisation ? Afficher les normales en appuyant sur la touche *'N'* dans une fenêtre de visualisation.
7. Visualiser l'histogramme des vecteurs normaux à l'aide de la fonction *'normalsHistogram'*. Proposer une solution pour déduire la normale au plan du sol.
8. La fonction *Segmentation.estimateFloorNormal* permet de réaliser cela. Quelle est la valeur de ce vecteur ? Proposer une solution pour aligner le sol avec le plan horizontal.
9. Implémenter cette solution dans la méthode *Segmentation.alignFloor*, il faudra pour cela compléter et utiliser la fonction *rotationMatrixFromVectors*.
10. La dernière étape consiste en la suppression du sol. Proposer et implémenter une solution dans la fonction *Segmentation.removeFloor*. Vous pourrez vous inspirer du code de la fonction de recherche du vecteur normal au plan du sol.
11. Sauvegarder le modèle obtenu.

III. Contrôle qualité

Après segmentation de l'objet, nous allons essayer de détecter les potentiels défauts présents. La finalité est d'automatiser le processus de tri entre les pièces correctes et les pièces défectueuses.

Pour cela, on se propose d'utiliser l'algorithme Iterative Closest Point ou ICP :

12. Quels sont les entrées et les sorties de l'algorithme ICP proposé par open3d (*open3d.registration.registration_icp*)?
13. Nous avons créé un modèle de référence lors de la partie précédente. Le charger avec un objet à traiter dans *data/objects*. Visualiser les deux objets à l'aide de la fonction *Registration.display*.
14. Nous allons estimer une transformation initiale grâce à l'algorithme RANSAC. L'Implémenter dans la fonction *Registration.processGlobal*. Comment fonctionne cet algorithme ? À quoi correspondent les valeurs de fitness, inlier_rmse et correspondance_set size ? Visualiser les résultats avec la fonction *Registation.display*, commenter ces résultats.
15. Nous pouvons maintenant utiliser l'algorithme ICP. L'Implémenter dans la fonction *Registration.processICP*. Quelle est la différence entre les algorithmes 'point to point' et 'point to plane' ? Comment fonctionnent-ils ?
16. Visualiser les résultats avec la fonction *Registation.display*, commenter ces résultats.
17. 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 ?
18. Trouver une manière de distinguer les pièces correctes des pièces défectueuses en vue d'une automatisation du processus.