TP Reconstruction
Master PPMD
ENSG

Bruno VALLET - IGN/MATIS

April 6, 2017

1 Introduction

L'objectif de ce TP est de vous faire appréhender certains enjeux de l'extraction de primitives par l'intermédiaire du problème de l'extraction de plans dans un nuage de points. Il s'agira donc d'implémenter des algorithmes permettant de retrouver un puis plusieurs plans dans un nuage de points donné:

$$\mathcal{X} = \{\mathbf{x}_i = (x_i, y_i, z_i)^t \in \mathbb{R}^3\}_{i=1..n}$$

2 Estimation d'un plan aux moindres carrés

Si les points de notre nuage $\mathcal{X}$ sont approximativement répartis sur un plan $\mathcal{P}$, la façon la plus simple de retrouver $\mathcal{P}$ est de minimiser l'erreur (aussi appelée norme L_2):

$$\mathcal{E}_{L_2}(\mathcal{P}) = \sum_{i=1}^{n} dist(\mathbf{x}_i, \mathcal{P})^2$$

2.1 La fonction `EstimationPlanL2()`

2.1.1 Création de la matrice d'énergie

Le plan estimé $\mathcal{P}$ sera cherché sous la forme $ax + by + cz + d = 0$, c'est a dire que nos inconnues sont les coefficients $\mathbf{a} = (a, b, c, d)^t$. L'énergie L_2 est alors une fonction de ces coordonnées:

$$\mathcal{E}_{L_2}(\mathbf{a}) = \sum_{i=1}^{n} (ax + by + cz + d)^2 = \mathbf{a}^t \sum_{i=1}^{n} (\mathbf{X}_i^t \mathbf{X}_i)\mathbf{a} = \mathbf{a} E_{L_2} \mathbf{a}^t$$

où $\mathbf{X}_i = (x_i, y_i, z_i, 1)^t$ sont les *coordonnées homogènes* du point $\mathbf{x}_i$, et E_{L_2} est appelée la matrice d'énergie.

- Construisez la matrice d'énergie E_{L_2} dans la fonction `EstimationPlanL2()`.

2.1.2 Minimisation de l'énergie

On peut démontrer que cette énergie est minimale pour le plus petit vecteur propre de X^h (cf. cours).

- Finissez d'implémenter la fonction `EstimationPlanL2()` en utilisant une fonction permettant de calculer les vecteurs/valeurs propres (eigen vector/values) d'une matrice.

2.2 La fonction `TestEstimationPlan()`

Nous allons maintenant tester la fonction `EstimationPlanL2()` en générant un nuage de points test. Pour cela, nous allons nous servir de la fonction `rand`: `rand` renvoie un nombre réel aléatoire dans $[0, 1]$ et `rand(i,j)` renvoie une matrice (i, j) de tels nombres aléatoires.

2.2.1 Générer des points sur un plan

- Utilisez `rand` pour définir a, b, c, d aléatoirement dans [`-sigmaCoef`,`+sigmaCoef`].
- Utilisez `rand` pour définir les vecteurs lignes de coordonnées X et Y des points de données dans $[0, 1]$.
- Calculez le vecteur ligne de coordonnées Z des points de données pour qu'ils soient dans le plan.
- Assemblez ces lignes dans une matrice XYZ de taille $(3, npts)$.

2.2.2 Ajouter du bruit

- Ajoutez un bruit uniforme dans [`-sigma`,`sigma`] aux coordonnées des points de donnée.

2.2.3 Ajouter des outliers

- Ajoutez un bruit uniforme dans [0,`sigmaOut`] aux coordonnées de `noutliers` points de données.

2.3 Lancez le test

Lancez `TestEstimation()`, jouez avec les paramètres. Quelle est l'influence des outliers ?

3 Estimation d'un plan par RANSAC

Nous allons maintenant implémenter un algorithme plus robuste aux erreurs de mesures et très employé dans la communauté de la vision par ordinateur: le RANdom SAmple Consensus (RANSAC).

RANSAC va tirer de nombreux échantillons de 3 points au hasard dans la donnée $\mathcal{X}$ et chercher si le plan défini par ces échantillons explique bien $\mathcal{X}$, c'est a dire si il est proche de beaucoup de points de $\mathcal{X}$.

3.1 Selection d'un échantillon

- Dans la boucle principale de RANSAC, créez le vecteur `ind` contenant 3 indices de points distincts dans [1,`npts`].
- Assemblez les coordonnées de ces 3 points dans une matrice 3x3 `P`.
- Optionel: rejetez les cas où les 3 points de l'échantillon sont alignés.

3.2 Distance d'un point a un plan: La fonction `DistPlan(P, XYZ)`

RANSAC nécessite de calculer les distances entre les points de données et les plans résultant des échantillons. On rappelle que la distance d'un point A à un plan $\mathcal{P}$ est:

$$dist(A, \mathcal{P}) = |\overrightarrow{PA}.\overrightarrow{n}|$$

où P est un point de $\mathcal{P}$ et $\overrightarrow{n}$ est la normale (unitaire) de $\mathcal{P}$

- Ecrire une fonction `DistPlan(P, XYZ)` qui renvoie le vecteur des distances des points de `XYZ` au plan `P` défini par 3 de ses points.

3.3 Separation inliers/outliers

Le score du plan est le nombre d'inliers, c'est a dire le nombre de points dont la distance au plan est inférieure au seuil.

- Trouver les vecteurs `inliers` et `outliers` des indices des inliers et outliers. On pourra utiliser une fonction qui renvoie les indices des éléments d'un `vecteur` qui vérifient une `condition`.
- Calculez le score du plan.

3.4 Sélection du meilleur plan

- Retenir les inliers, outliers et l'échantillon de 3 points du meilleur plan, c'est a dire de celui qui avait le plus d'inliers.

3.5 Tester RANSAC

Testez votre algorithm, jouez avec les paramètres. Quel(s) problème(s) résout RANSAC ?

4 Estimation de plusieurs plans par RANSAC

4.1 Cas de 2 plans

4.1.1 Générer des points sur 2 plans

- Modifiez `TestEstimationPlan()` pour que les points de données soient maintenant sur 2 plans.

4.1.2 Estimation de 2 plans par RANSAC

- Utilisez votre fonction `EstimationPlanRansac(...)` pour implementer la fonction `EstimationMultiPlanRansac(...)` qui permet de détecter ces 2 plans.

4.2 Cas de n plans, n connu

4.2.1 Générer des points sur n plans

- Modifiez `TestEstimationPlan()` pour que les points de données soient maintenant sur `nPlan` plans, où `nPlan` est un paramètre que vous ajouterez.

4.2.2 Itérer l'estimation RANSAC

- Modifiez `EstimationMultiPlanRansac()` pour détecter ces n plans.

4.3 Cas de n plans, n inconnu

- Proposez une modification de `EstimationMultiPlanRansac()` pour que la détection se fasse sans connaître le nombre de plans. On pourra proposer un paramètre pertinent pour l'arrêt de la recherche.