Traitements géométriques de points 3D

Travaux pratiques : Recalage par ICP

1. Vous devez faire évoluer ce code au fur et à mesure du TP, pour répondre aux questions. Vous pouvez réutiliser des parties réalisées pendant les précédents TPs.
2. **Déposer sur le moodle un zip contenant votre fichier tp.cpp (qui doit contenir tout votre code), ainsi que votre rapport contenant une description très brève des questions que vous avez traitées et des captures d'écran présentant vos résultats.**

1 Base de code

Pour compiler le projet, ouvrez une console, placez vous dans le répertoire principal, et tapez :

```
make
```

Pour l"executer, tapez :

```
./tp
```

1.1 Principaux éléments fournis

1. un viewer simple sous `glut`
2. une classe de kd-tree utilisant la librairie open source `ann`
3. une classe point 3D (`Vec3`, dans `Vec3.h`).
4. une classe de matrice 3x3 (`Mat3`, dans `Vec3.h`). Elle contient notamment une méthode `setRotation()` qui vous sera bien utile pour l'ICP...

2 Modèles

Le répertoire "./pointsets" contient des fichiers binaires encodant des pointsets, sous la forme $x, y, z, nx, ny, nz, \cdots$.
Les modèles peuvent apparaitre sous plusieurs formes :

1. model.pn
2. model_subsampled.np : est une version décimée de model.pn
3. model_subsampled_extreme.np : est une version **extrèmement** décimée de model.pn (contient entre 10 et 15% des points seulement)
4. model_partial_x.pn : est une version partielle du modèle model.pn

Ces modèles serviront à tester vos algorithmes.

3 Exercice : ICP

I) Implémentez une fonction

```
void ICP(std::vector<Vec3> const & ps , std::vector<Vec3> const & nps ,
         std::vector<Vec3> const & qs , std::vector<Vec3> const & nqs ,
         BasicANNkdTree const & qsKdTree , Mat3 & rotation , Vec3 & translation , unsigned int nIterations ) {
 ...
}
```

qui doit mettre à jour la matrice de rotation `rotation` ainsi que le vecteur de translation `translation`, afin que `positions2` soit recalé au mieux sur `positions` :

$$R, t = argmin\{\sum_i w_i \|R.positions2[i] + t - proj(positions2[i], positions)\|^2\} \quad (1)$$

`BasicANNkdTree const & kdtree` est le kd-tree que vous devez construire en preprocess à partir de `positions`. w_i est le poids du point numéro i dans le calcul (pour l'instant, mettez le à 1).

— Exécutez votre code sur les modèles dino.pn et dino2.pn. Est-ce que vous obtenez un recalage parfait ? (captures d'écran)
— Exécutez votre code sur les modèles dinosubsampledextreme.pn et dino2subsampledextreme.pn. Est-ce que vous obtenez un recalage parfait ? Discutez ce point. (captures d'écran)
— Adaptez votre code pour que l'étape de projection n'utilise pas simplement le kd-tree pour trouver le point le plus proche, mais qu'elle permette de projeter le point sur une surface HPSS interpolante. Est-ce que vous obtenez un meilleur recalage ? (Pour cela, vous pourrez utiliser les normales associées à positions, afin de définir la surface HPSS). Discutez ce point. (captures d'écran)
— Proposez, selon votre intuition, une manière d'adapter w_i afin de ne pas tenir compte de certains points, dans les cas où les deux pointsets ne se recouvrent que partiellement (c'est le cas avec la plupart des données scannées réelles !).

3.1 Recalage de pointsets identiques

Charger les pointsets de telle sorte que le pointset 2 est EXACTEMENT le pointset 1, placé légèrement différemment dans la scène 3D.

```
        loadPN("pointsets/african_statue2_subsampled_extreme.pn", positions, normals); // ou bien "pointsets/face.pn
        positions2 = positions;
        normals2 = normals;

5       srand(time(NULL));
        ICProtation = Mat3::RandRotation();
        ICPtranslation = Vec3(-1.0 + 2.0 * ((double)(rand()) / (double)(RAND_MAX)),-1.0 + 2.0 * ((double)(rand()) / (

        for( unsigned int pIt = 0; pIt < positions2.size(); ++pIt ) {
10          positions2[pIt] = ICProtation * positions2[pIt] + ICPtranslation;
            normals2[pIt] = ICProtation * normals2[pIt];
        }
```

Dans ce contexte, on devrait pouvoir retrouver un alignement parfait à chaque fois. ICP est une méthode *itérative*, *locale*, et en conséquence ne marchera pas à tous les coups : il est possible de tomber dans des minima locaux de l'énergie. Proposez une méthode simple pour améliorer les résultats (discussion ouverte pendant le TP).

3.2 Recalage de pointsets incomplets

Charger les pointsets de telle sorte que le pointset 2 soit une version partielle du pointset 1, placé légèrement différemment dans la scène 3D.

```
        loadPN("pointsets/african_statue.pn", positions, normals);
        loadPN("pointsets/african_statue_partial_1.pn", positions2, normals2); // ou "pointsets/african_statue_part
3
        srand(time(NULL));
        ICProtation = Mat3::RandRotation();
        ICPtranslation = Vec3(-1.0 + 2.0 * ((double)(rand()) / (double)(RAND_MAX)),-1.0 + 2.0 * ((double)(rand()) / (

8       for( unsigned int pIt = 0; pIt < positions2.size(); ++pIt ) {
            positions2[pIt] = ICProtation * positions2[pIt] + ICPtranslation;
            normals2[pIt] = ICProtation * normals2[pIt];
        }
```

Dans ce contexte, l'ICP peut être encore moins stable. Donnez votre intuition de la raison de ce problème.

3.3 Recalage de pointsets inconsistents

Charger les pointsets de telle sorte que le pointset 1 soit une version partielle du pointset 2, placé légèrement différemment dans la scène 3D.

```
        loadPN("pointsets/african_statue_partial_1.pn", positions, normals); // ou "pointsets/african_statue_partia
        loadPN("pointsets/african_statue.pn", positions2, normals2);

4       srand(time(NULL));
        ICProtation = Mat3::RandRotation();
        ICPtranslation = Vec3(-1.0 + 2.0 * ((double)(rand()) / (double)(RAND_MAX)),-1.0 + 2.0 * ((double)(rand()) / (

        for( unsigned int pIt = 0; pIt < positions2.size(); ++pIt ) {
9           positions2[pIt] = ICProtation * positions2[pIt] + ICPtranslation;
            normals2[pIt] = ICProtation * normals2[pIt];
        }
```

Dans ce contexte, l'ICP va chercher à trouver dans `pointset` des parties qui n'existent pas (par exemple, on essaie d'aligner une statue qui a des bras sur une statue qui n'a pas de bras...), et l'algorithme est encore moins stable. Ce problème est plus compliqué que le précédent. Donnez votre intuition de la raison du problème, et si vous y arrivez, proposez une possible solution (autre que celle donnée dans le cours, qui est d'utiliser la méthode **Sparse ICP**).