TP Imagerie 3D no 3

18 octobre 2018

Ce TP a pour objectif de programmer un algorithme de segmentation 3D par recalage automatique.

- ***Le TP est noté : le compte-rendu doit être envoyé** sous **forme électronique à :** gerard.subsol@lirmm.fr avant **le mercredi 22 novembre 2017 (minuit)***
- *Le compte-rendu doit inclure **la date**, **vos noms** et être composé d'au moins 1 à 2 pages de texte **décrivant la méthode** utilisée pour répondre aux questions ainsi que les sources en C++ et **quelques captures d'écran** pour évaluer le résultat.*
- *Le tout doit être sous la forme d'un **unique fichier pdf**.*
- *Le TP peut se faire seul ou en binôme*
- *La participation active pendant le TP pourra aussi être prise en compte.*
- ***Tout plagiat sera lourdement sanctionné**.*
- *Tout compte-rendu envoyé sous une mauvaise forme ou hors-délai sera sanctionné par un 0.*

Le but est de segmenter une structure anatomique dans une image par recalage (rigide, affine ou similitude). La mise en correspondance est effectuée au voisinage du maillage 3D par comparaison des profils d'intensité.

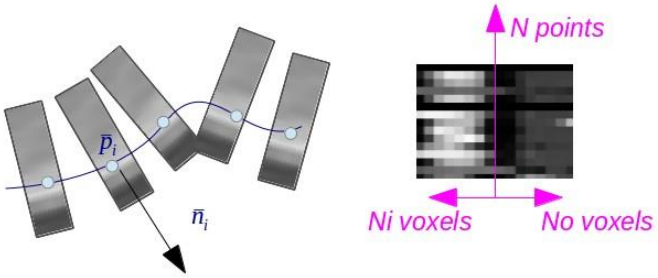
1. **Visualisation des données** :

- Télécharger les fichiers source et les données dans un sous répertoire *data.*
- Compiler le programme de base en exécutant *make* depuis un terminal.
- Par default, l'image source ./data/thigh_m.mhd et l'image cible ./data/thigh_f.mhd sont chargées. Un maillage 3D de référence, aligné sur l'image source est également chargé à l'exécution (./data/femur_m.obj).
- Lancer l'exécutable icpImg. Deux fenêtres s'ouvrent : une vue 3D et une vue « image » (visualisation MPR avec trace du maillage 3D).
- Il est possible de basculer de l'image source vers l'image cible au moyen de la touche S. Naviguez dans les images (bouton droit et molette de la souris) afin de visualiser l'écart initial entre le modèle et la cible. Il est possible de positionner l'image cible différemment en éditant le champs '*Position*' dans le fichier ./data/thigh_f.mhd (fichier texte).
- La touche espace permet d'exécuter une itération de l'algorithme de recalage dont les étapes sont les suivantes :

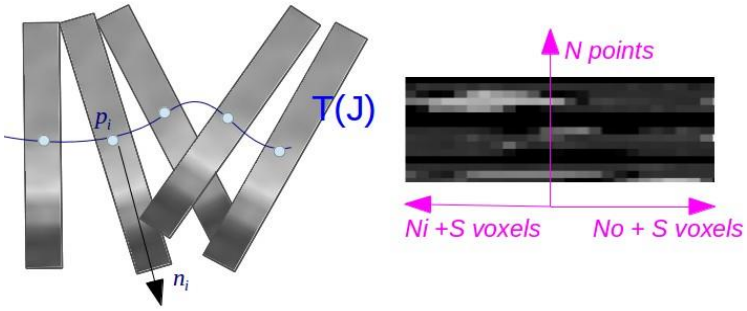
```
        // mise à jour des profils d'intensité cibles
mesh.updateNormals();
CImg<unsigned char> prof = computeProfiles(mesh, target, Ni+S, No+S, l);

        // calcul de la distance entre profils d'intensité
CImg<float> dist = computeDistance(referenceProf,prof,metric);

        // calcul des correspondences 3d
computeCorrespondences(mesh,dist,l);

        // calcul de la transformation globale

        // mise à jour des points 3D du modèle
```

2. Calcul des profils d'intensité :

Les profils sources sont calculés à l'initialisation : pour chaque point du maillage qui comprend *N points*, l'image source est échantillonnée le long de la normale. Le pas d'échantillonnage est *l* ; le nombre d'échantillons *Ni* (à l'intérieur du modèle), *No* (à l'extérieur). L'image des profils est donc de taille N(Ni+No).



Les profils cibles sont calculés à chaque itération à partir du maillage (dans sa position courante) et de l'image cible. La taille des profils est agrandie de *S* : la distance de recherche.



Compléter la fonction *computeProfiles* dont le but est de remplir l'image de profils d'intensité en fonction des différents paramètres.

```
CImg<unsigned char> computeProfiles(const MESH& mesh, const IMG<unsigned char,float>& img, const unsigned int Ni, const unsigned int No, const float l,const unsigned int interpolationType=1)
```

Pour récupérer la valeur d'intensité *val* (du type unsigned char) au point *p*, utiliser la fonction :

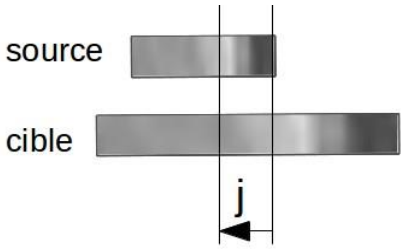
```
val = img.getValue(p,interpolationType)
```

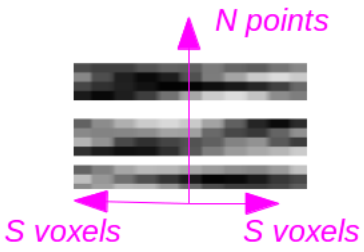
Par défaut, on effectue une interpolation linéaire dans l'image (*interpolationType=1*).

Pour vérifier cette fonction, vous pouvez visualiser l'image de profils dans une fenêtre au moyen de la fonction :

prof.display()

3. Calcul des profils de similarité :





On calcule une image de distances de taille N*2S. Pour chaque décalage j (-S≤j<S) et chaque point i (0≤i<N), on calcule une distance entre le profil source et une partie du profil cible (entre j-Ni et j+No). Cette distance est calculée au moyen d'une mesure de similarité.

Compléter la fonction *computeDistance* dont le but est de remplir l'image de distances. Implémenter les mesures SSD (somme des différences au carré) et NCC (corrélation croisée normalisée).

```
CImg<float> computeDistance(const CImg<unsigned char> &sourceProf,const CImg<unsigned char> &targetProf,const Metric metric)
```

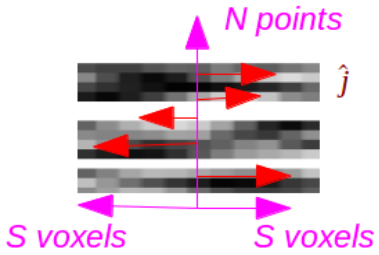
A partir de l'interface, on peut changer de mesure grâce à la touche M.

4. Calcul des correspondances

Pour chaque point, on obtient un décalage optimal l en prenant la distance minimale dans l'image de distances.

Pour le point i par exemple, le point 3D correspondant à ce décalage est : $p_c = p_i + l.n_i$. On fixe ce point grâce à la fonction :

mesh.setCorrespondence(pc,i);



Compléter la fonction *computeCorrespondences* dont le but est de calculer les correspondances 3D pour chaque point du maillage.

```
void computeCorrespondences(MESH& mesh, const CImg<float> &dist,const float l)
```

Une fois les correspondances établies, on utilise l'algorithme ICP pour calculer la transformation optimale qui approxime au mieux les correspondances.

5. Tests

Comparez qualitativement les résultats de convergence de l'algorithme de recalage en fonction de :

- la mesure de similarité (changer avec la touche M)
- le type de transformation (touche +)
- la position initiale de l'image cible (éditer thigh_f.mhd)
- Les paramètres Ni, No, l et S