Traitement de Nuage de Points

TP2 - Ball-Pivoting Algorithm

EPITA - Majeure IMAGE

Novembre/Décembre 2022

Introduction

L'objectif de ce TP est d'implémenter une version simplifiée de l'algorithme Ball-Pivoting pour la reconstruction de maillage triangulaire 3D à partir de nuage de points 3D.

Étape 0.1 Mise en place du projet

- télécharger et décompresser l'archive `TP2.zip`
- exécuter `mkdir build` à la racine du projet
- dans le dossier `build`, exécuter `cmake ..` pour configurer le projet
- dans le dossier `build`, exécuter `make -j` pour compiler le projet

Étape 0.2 Visualisation de nuage de points et maillage 3D

- télécharger l'AppImage depuis meshlab.net
- exécuter `chmod +x MeshLab2022.02-linux.AppImage`
- exécuter `./MeshLab2022.02-linux.AppImage data/bunny.obj`

Des paquets existent sur certaines distribution Linux (`sudo apt-get install meshlab` sur Ubuntu)

Rappel : la documentation de la librairie C++ **Eigen** est accessible à eigen.tuxfamily.org/dox.

1 Préliminaires

Étape 1.1 Implémenter la méthode `triangle_normal` dans `geometry.cpp` qui calcule le vecteur unitaire normal au triangle défini par trois points et orienté selon la règle de la main droite.

Étape 1.2 Implémenter la méthode `triangle_circumcenter` dans `geometry.cpp` qui calcule le centre du cercle circonscrit à un triangle.

Indication : le centre du cercle circonscrit d'un triangle défini par $\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2\}$ est

$$\mathbf{p}_0 + \frac{\left(\|\mathbf{p}_{10}\|^2\mathbf{p}_{20} - \|\mathbf{p}_{20}\|^2\mathbf{p}_{10}\right) \times \mathbf{v}}{2\|v\|^2}$$

avec $\mathbf{p}_{10} = \mathbf{p}_1 - \mathbf{p}_0$, $\mathbf{p}_{20} = \mathbf{p}_2 - \mathbf{p}_0$ et $\mathbf{v} = \mathbf{p}_{10} \times \mathbf{p}_{20}$.

Étape 1.3 Implémenter la méthode `compute_center` dans `geometry.cpp` qui calcule le centre de la boule supérieur de rayon r et qui passe par trois points.

Indication : le centre $\mathbf{c}$ de la boule supérieure de rayon r qui passe par trois points $\{\mathbf{p}_i\}$ se trouve sur la ligne définie par le centre du cercle circonscrit et le vecteur normal du triangle, et vérifie $\|\mathbf{p}_0 - \mathbf{c}\| = r^2$.

2 Pivot

L'algorithme repose sur une opération qui fait pivoter une boule de rayon r autour d'une arête orientée courante e_{ij} qui relie les points $\mathbf{p}_i$ et $\mathbf{p}_j$. La boule pivote est s'arrête sur le premier point rencontrée. Cela revient à selectionner dans un voisinage le point qui minimise l'angle parcouru par la boule. La figure 1 illustre cette opération de pivot. Le résultat sera l'indice l et le centre $\mathbf{s}$ qui minimise θ.

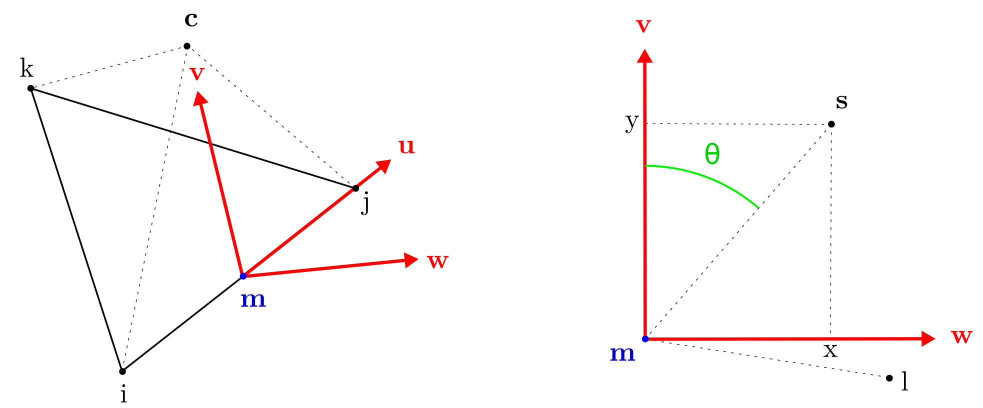


Figure 1: A gauche : état initial de la fonction pivot. On considère l'arête e_{ij} pour faire pivoter la boule de centre $\mathbf{c}$ dans le plan défini par le centre $\mathbf{m}$ de l'arête et les vecteurs unitaires $\mathbf{v}$ et $\mathbf{w}$. Les vecteurs $\mathbf{u}$, $\mathbf{v}$ et $\mathbf{w}$ forment un repère orthonormé orienté. A droite : un point indicé par l dans le voisinage de $\mathbf{m}$ est considéré comme candidat. Le point $\mathbf{s}$ correspond au centre de la boule qui passe par les points indicés par i, j et l. Dans le plan défini par $\mathbf{m}$, $\mathbf{w}$ et $\mathbf{v}$, le point $\mathbf{s}$ a pour coordonnées 2D (x, y) permettant de calculer l'angle θ.

Étape 2. Implémenter la fonction `pivot` de `pivot.cpp`.

Indications

- $\theta \in (0, 2\pi)$ (attention à la fonction `std::acos`, par exemple si $x < 0$)
- l doit être différent de i, j et k
- les voisins indicés par l doivent se trouver à une distance d'au plus $2r$ de $\mathbf{m}$
- à causes d'erreurs numériques potentielles, utiliser la fonction `safe_acos` définie dans le fichier `pivot.cpp` au lieu de `std::acos`

3 Algorithme principal

L'algorithme Ball-Pivoting suit le principe d'un front qui avance progressivement sur le nuage de points créant au fur-et-à-mesure des triangles jusqu'à ce que la forme 3D soit recouverte. Le front est représenté par une pile de liste doublement chaînée dont l'élément principal est une arête de type `Edge` défini dans le fichier `main.cpp`. De plus, un status `FREE`, `FRONT` ou `INSIDE` est assigné à chacun des points et un vecteur `outter_edges` permet de stocker les arêtes qui partent de chaque point. L'algorithme 1 résume les principales étapes.

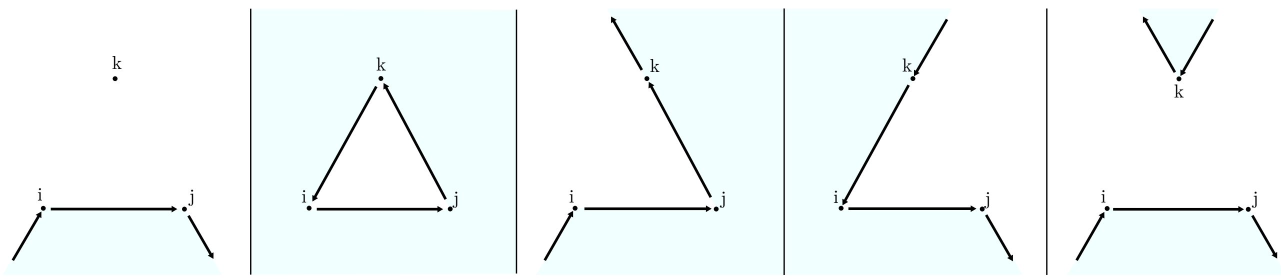
La dernière étape de mise-à-jour du front de propagation dépend de 5 cas possibles illustrés par la figure 2. La mise-à-jour suit la procédure générale suivante

- création de nouvelles arêtes
- mise-à-jour du chaînage `prev`/`next`
- mise-à-jour des `outter_edges`
- mise-à-jour des status
- marquage de certaines arêtes à supprimer
- empilement sur la pile

Algorithm 1 Algorithme principal du Ball-Pivoting

initialize all point status to `FREE`
initialize empty outter edges for each points
initialize stack with a first triangle whose points are set to `FRONT`
while stack is not empty **do**
  pop e_{ij} from the stack
  if e_{ij} must be removed **then**
    delete the edge and continue
  if e_{ij} is not on the front **then**
    continue
  if cannot pivot **then**
    continue
  update front depending on the case

Suivant le cas, certaines étapes ne sont pas à effectuer. Si l'algorithme fait face à aucun des 5 cas alors la boucle principale peut continuer.



cas 1 cas 2 cas 3 cas 4 cas 5

Figure 2: Mise-à-jour du front après pivot autour de l'arête e_{ij}. Un nouveau triangle i, k, j est ajouté.

Étape 3. Implémenter l'algorithme principal dans le fichier `main.cpp` en suivant l'algorithme 1 et la figure 2. Exécuter `./ball_pivoting ../data/bunny.obj` et visualiser le maillage `mesh.obj` produit. Faites varier `ball_radius` pour analyser l'impacte sur la triangulation.

4 Initialisation

La fonction `initial_triangle` retourne pour le moment un triangle codé en dur pour le nuage de point `bunny.obj` seulement. Pour pouvoir mailler n'importe quel nuage de point, cette fonction doit renvoyer un triangle initial duquel partir. Un triangle contenant le point le plus haut du nuage selon l'axe Z est une solution possible. L'algorithme d'initialisation consiste alors à effectuer

- la recherche du point k avec la coordonnée z maximale
- la recherche (et le stockage) de tous ses voisins dans un rayons $2r$
- la recherche dans ses voisins d'une paire de points (i, j) qui produit avec k une boule vide

Étape 4. Implémenter la fonction `initial_triangle`.