Projet de modélisation et programmation C++ MMIS

Triangulation d'un ensemble de points scannés

Cette seconde moitié du semestre est consacrée à la conception d'un code orienté objet permettant de résoudre des problèmes relatifs à votre filière. Votre équipe regroupe des compétences complémentaires (image, calcul scientifique, aide à la décision), et nous nous sommes efforcés de vous proposer un sujet où tout le monde pourra trouver son intérêt.

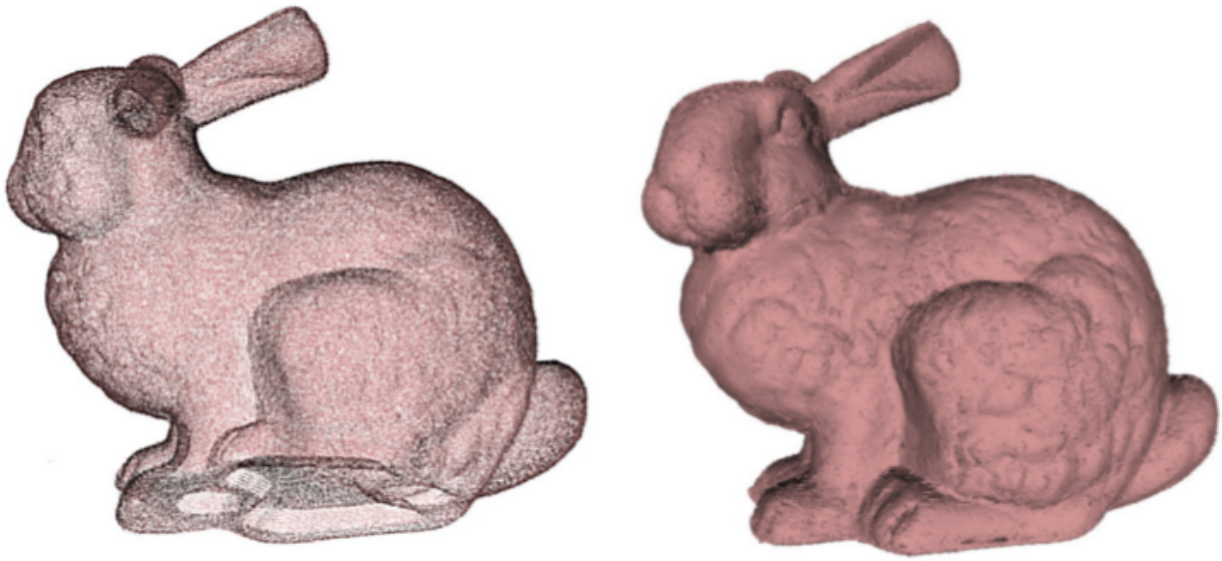
Consigne générale : **plus que les résultats, c'est la méthodologie, la clarté du code, les différents éléments de conception, qui doivent être mise en avant**. Le code doit être guidé par la description mathématique du problème, et assurer une grande souplesse au niveau des interfaces.

Vous pouvez (devez !) reprendre les structures de données de base développées durant la première moitié du semestre.

1 Objectifs généraux du projet

1.1 Objectif scientifique

Le projet MMIS 2012-2013 a pour but d'implémenter un algorithme récent qui calcule une triangulation, i.e. un maillage avec des faces triangulaires uniquement, de la surface d'un objet scanné [1]. Cet objet peut être non convexe, voir la figure 1. L'algorithme prend en entrée uniquement les coordonnées dans $\mathbb{R}^3$ des points issus du scan de l'objet.



(a) (b)

Figure 1: (a) Entrée de l'algorithme : un nuage de points issu du scan d'un modèle réel.(b) Sortie de l'algorithme : une reconstruction virtuelle 3D sous forme de maillage triangulaire.

1.2 Objectifs pédagogiques

Le projet de modélisation et programmation est un de vos premiers projets en grand groupe. Contrairement à l'APP d'APOO, ce projet est volontairement peu dirigé, afin de vous laisser une grande liberté de choix (structures de données, algorithmes, partage du travail, ...) et de développer votre autonomie et favoriser votre créativité et votre esprit d'initiative. Vous allez donc être confrontés à des difficultés nouvelles, notamment d'ordre organisationnel. Il s'agit du premier objectif pédagogique, que vous retrouverez pour certains d'entre vous en projet de spécialité et/ou en stage 2A.

Le deuxième objectif est de vous faire pratiquer la programmation en C++ dans le cas d'un problème "grandeur nature", tel que vous pourrez le retrouver dans votre vie professionnelle future. C'est le moment de mettre en application tous les bons réflexes d'algorithmique et de programmation que vous avez appris depuis votre entrée à l'Ensimag : choix réfléchi et argumenté des structures de données concrètes et du découpage du code (classes, ...), code documenté, tests unitaires et tests de performance, utilisation d'outils de débogage et de *profiling* (gdb, valgrind, gprof, ...), etc.

Le troisième et dernier objectif de ce projet est de mobiliser certaines de vos compétences spécifiques à la filière MMIS : vous retrouverez dans ce projet des notions vues en approche morphologique et statistique pour le signal et les images, en graphique 3D, en optimisation combinatoire, etc.

2 Résumé du travail demandé

Vous avez codé en TP des classes Point, Triangle et Maillage bidimensionnelles : les points étudiés étaient des objets de $\mathbb{R}^2$, les triangles et les maillages étaient plans. Nous allons reprendre ces classes, mais avec des objets parfois bidimensionnels, parfois tridimensionnels, pour les appliquer à un problème classique de modélisation géométrique.

Le récent algorithme proposé par Labatut, Pons et Keriven permet de calculer de manière robuste (i.e. peu sensible au bruit d'acquisition) une triangulation 2D de points de l'espace $\mathbb{R}^3$ issus de scans [1]. Il procède en 5 étapes successives :

1. calcul d'une tétraédrisation (i.e. triangulation 3D) de Delaunay T de l'ensemble de points $\mathcal{P}$;
2. construction du graphe G dual de T : les nœuds de G correspondent aux tétraèdres de T, et deux arcs (un orienté dans chaque sens) relient deux tétraèdres adjacents. Un arc correspond donc à un triangle *orienté* frontière entre deux tétraèdres. On ajoute également un nœud source et un nœud puits virtuels, et des arcs de la source vers tous les autres nœuds (sauf le puits), et de tous les nœuds (sauf la source) vers le puits ;
3. calcul d'une "énergie de visibilité" E_{vis} à partir de $\mathcal{P}$ (voir ci-dessous) ;
4. calcul d'une "énergie de qualité" E_{qual} à partir de $\mathcal{P}$ (voir ci-dessous) ;
5. calcul de la coupe minimale sur G, qui minimise l'énergie $E_{vis} + \lambda E_{qual}$. λ est un paramètre de l'algorithme, à ajuster dans les tests. Les nœuds du côté de la source sont considérés comme Extérieurs, et les nœuds du côté du puits comme Intérieurs.

Le résultat de l'algorithme est constitué des triangles de T correspondant aux arcs de la coupe minimale. Attention, l'union de ces triangles (et de leurs faces) forme une triangulation qui n'est sans doute pas de Delaunay.

L'objectif du projet est d'implémenter et de tester cet algorithme. On peut découper le travail en plusieurs parties, que nous détaillons ci-dessous.

3 Calcul d'une triangulation nD de Delaunay

3.1 Principe

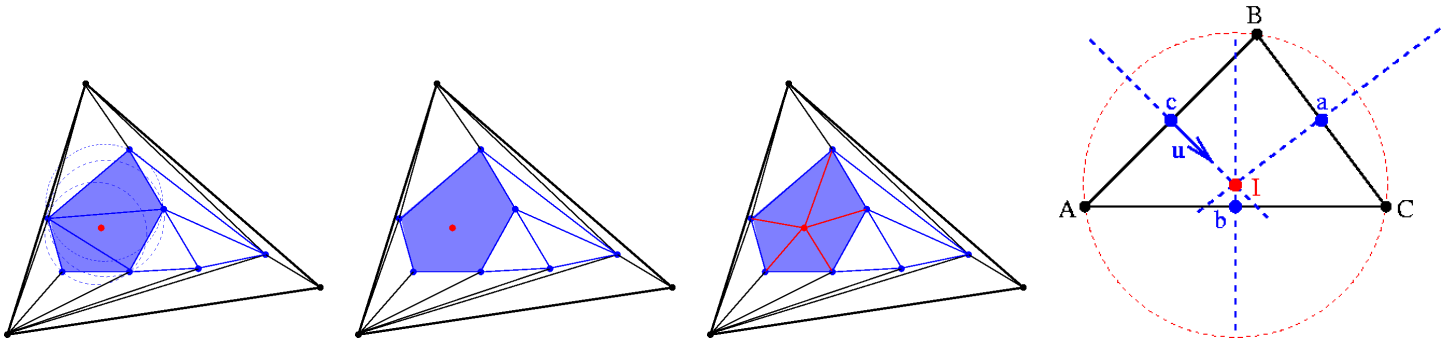
L'algorithme de Bowyer-Watson est un algorithme qui calcule la triangulation de Delaunay d'un ensemble de points $\mathcal{P}$ en dimension n quelconque. Il a été proposé simultanément mais indépendamment (!) par A. Bowyer [2] et D.F. Watson [3]. Ce n'est pas l'algorithme le plus efficace qui existe, mais il présente le double avantage d'être simple et de s'implémenter en dimension quelconque. Il s'agit d'un algorithme incrémental, dont le principe est le suivant (voir Figure 2 (a) à (c) pour un exemple dans $\mathbb{R}^2$) :

1. Créer un grand n-triangle (i.e. ensemble de $n+1$ points linéairement indépendants : triangle en dimension 2, tétraèdre en dimension 3, etc.) englobant $\mathcal{P}$. Ceci forme notre triangulation de Delaunay initiale $\mathcal{T}$, composée d'un seul triangle.
2. Tant qu'il reste des points non traités dans $\mathcal{P}$:
   (a) Ajouter un point P de $\mathcal{P}$ à $\mathcal{T}$.
   (b) Trouver tous les n-triangles de $\mathcal{T}$ dont la n-sphère circonscrite contient P.

(c) Supprimer tous ces triangles.

(d) Trianguler le n-polyèdre convexe ainsi créé en joignant tous ses sommets à P et mettre à jour $\mathcal{T}$.

3. Supprimer les sommets du grand n-triangle englobant, ainsi que les n-triangles incidents en ces sommets.



(a) (b) (c) (d)

Figure 2: (a–c) Etapes 2(b) à 2(d) de l'algorithme de Bowyer-Watson. (d) Calcul du centre I du cercle circonscrit à un triangle ABC.

3.2 Travail demandé

1. Rendre template sur la dimension n de l'espace les classes Point, Triangle et Maillage codées en TP.
2. Définir une classe (elle aussi template) NuageDePoints permettant de représenter et de gérer un ensemble de points de $\mathbb{R}^n$.
3. Implémenter l'algorithme de Bowyer-Watson en tant que constructeur de la classe Maillage<n>.
   L'algorithme n'est pas très difficile à implémenter, mais il y a un peu de maths (géométrie) à faire pour l'étape 2(b), et surtout il s'agit de bien réfléchir aux structures de données à utiliser pour implémenter efficacement les étapes 2(b) à 2(d).

4 Algorithme de coupe minimale/flot maximum

4.1 Principe

L'objectif est de calculer une coupe minimale dans un graphe muni d'un nœud source et d'un nœud puits. Un théorème classique de théorie des graphes stipule que ce problème est équivalent au calcul d'un flot maximum dans le graphe.

4.2 Travail demandé

1. Créer un ensemble de classes permettant la gestion des graphes.
2. Ajouter un constructeur à votre classe Graphe, afin de construire le graphe dual d'une tétraédrisation de Delaunay quelconque.
3. Implémenter un algorithme de flot maximum/coupe minimale, à partir d'une source, d'un puits, et de poids associés aux arcs du graphe. Vous pouvez par exemple implémenter l'algorithme de Ford-Fulkerson [4,5], ou tout autre algorithme vu en cours d'optimisation combinatoire.

5 Calcul d'une énergie de visibilité

5.1 Principe

L'objectif ici est d'ajuster le poids des arcs du graphe, en fonction de ce qu'on sait ou suppose sur les tétraèdres (extérieurs ou intérieurs). L'algorithme original de Labatut, Pons et Keriven suppose qu'on connaît la direction exacte des faisceaux laser qui ont générés les points du scan $\mathcal{P}$. Ce n'est pas notre cas, donc on va adopter une autre approche.

Soit $\mathcal{P}$ un ensemble de points de $\mathbb{R}^3$ échantillonnés sur une surface lisse S inconnue. En 2007, Katz, Tal et Basri ont proposé une technique très simple pour deviner lesquels, parmi ces points, sont sur une partie visible de S à partir d'un point de vue p donné (voir figure 3) [6] :

1. soit R un rayon supérieur à la plus grande distance entre p et les points de $\mathcal{P}$. Appliquer à tout point p_i de $\mathcal{P}$ la transformation suivante (*inversion sphérique* par rapport à la sphère de centre p et de rayon R) :

$$f(p_i) = p_i + 2(R - \|p_i - p\|)\frac{p_i - p}{\|p_i - p\|} \tag{1}$$

2. calculer l'enveloppe convexe $\mathcal{E}$ de l'ensemble $\{p\} \cup \{f(p_i), p_i \in \mathcal{P}\}$;
3. les points de $\mathcal{P}$ visibles depuis p sont les p_i tels que $f(p_i) \in \mathcal{E}$ (voir Figure 4 (a)).

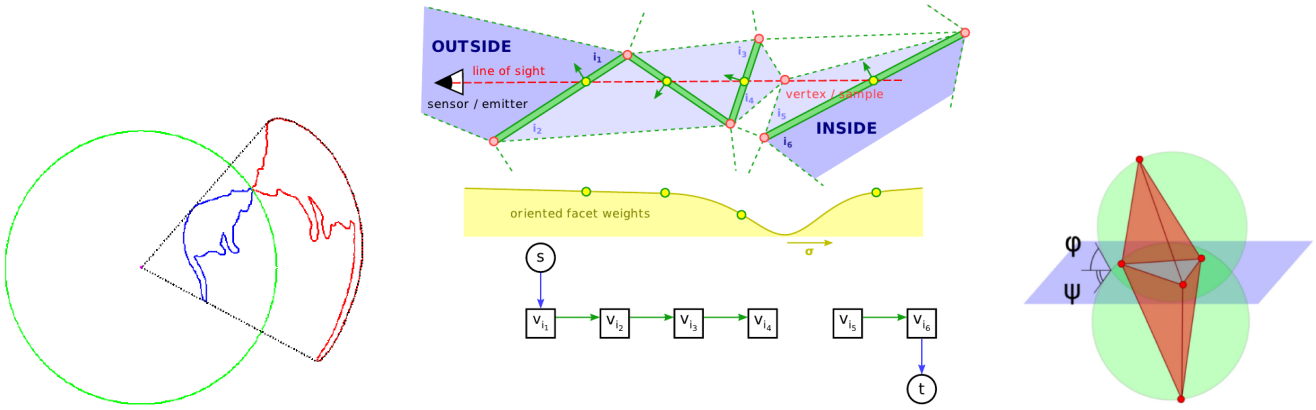
(a) (b) (c)

Figure 3: (a) Une surface S. (b) Ensemble de points $\mathcal{P}$ échantillonnés sur S. (c) Sous-ensemble des points de $\mathcal{P}$ visibles à partir du point de vue où a été prise l'image.

L'algorithme qui permet de calculer les "poids de visibilité" est le suivant (voir Figure 4 (b)).

1. Choisir quelques (3 à 6) points de vue p situés de part et d'autre du nuage de points $\mathcal{P}$. Tous les poids des arcs de G sont initialisés à zéro.
2. Pour chaque point de vue p, calculer les points de $\mathcal{P}$ visibles p_i, d'après la méthode de Katz, Tal et Basri.
3. Pour chaque demi-droite $[pp_i[$, calculer les tétraèdres et les triangles de T intersectés par cette demi-droite.
4. Dans G, l'arc reliant la source au nœud correspondant au premier tétraèdre intersecté voit son poids augmenter de α. α est un paramètre de l'algorithme, à faire varier dans vos tests.

5. Les arcs correspondants aux triangles de T intersectés par la demi-droite $[pp_i[$ avant p_i voient également leur poids augmenter, de $\alpha(1 - e^{-d^2/2\sigma^2})$. d est la distance entre p_i et l'intersection du triangle avec $[pp_i[$. σ est un autre paramètre de l'algorithme, également à faire varier dans vos tests.
6. Le tétraèdre auquel appartient le point de la demi-droite $[pp_i[$ situé au-delà de p_i à une distance 3σ de p_i est considéré comme ayant une forte probabilité d'être intérieur, et donc le poids de l'arc reliant le nœud correspondant au puits est augmenté de α.



(a) (b) (c)

Figure 4: (a) Algorithme de Katz, Tal et Basri. (b) Calcul des poids pour l'énergie de visibilité. (c) Ajustement des poids pour l'énergie de qualité.

5.2 Travail demandé

1. Implémenter l'algorithme décrit ci-dessus. Le nombre de points de vue sera un paramètre de l'algorithme. Ils devront être régulièrement espacés.

6 Calcul d'une énergie de qualité

6.1 Principe

L'objectif ici est de modifier les poids des arcs du graphe, de manière à obtenir une triangulation plus robuste au bruit d'acquisition. En effet, le nuage de points 3D généré par un scan laser contient généralement un nombre non négligeable de points fantômes, ne correspondant pas à la surface échantillonnée, et de points avec des coordonnées totalement erronées (*outliers*).

L'idée de Labatut, Pons et Keriven est de favoriser dans la triangulation résultat les triangles qui sont presque de Delaunay. Pour cela, on étudie tout triangle Δ de T, et ses deux tétraèdres adjacents (voir Figure 4 (c)). Soient S_1 et S_2 les sphères circonscrites à ces deux tétraèdres. Ces sphères intersectent le plan portant Δ avec un angle de φ et de ψ, respectivement. On ajoute aux deux arcs orientés de G correspondant à Δ le poids $1 - min(cos\varphi, cos\psi)$.

6.2 Travail demandé

1. Implémenter l'algorithme décrit ci-dessus.

7 Interface de visualisation et tests

7.1 Travail demandé

1. Ecrire (par exemple comme constructeur de la classe NuageDePoints) un petit parseur qui construit un nuage de points à partir d'un fichier au format .pts. Ce format est très simple : chaque ligne correspond exactement à un point, et contient les coordonnées de ce point.
2. Ecrire une méthode qui calcule la triangulation d'un nuage de points, en faisant appel aux classes et méthodes développées dans les parties précédentes.
3. En parallèle, développer une petite interface de visualisation de nuages de points et de triangulations 2D et 3D. Vous êtes libres de reprendre du code développé pour un autre projet, ou d'utiliser des bibliothèques existantes (Qt, libQGLViewer, ...). Néanmoins, votre code doit être documenté et s'intégrer harmonieusement dans le reste du projet.
4. Tester votre code sur des données soit trouvées par vos soins, soit fournies par les encadrants, en faisant varier les paramètres. L'Ensimag dispose d'un scanner laser, identique à celui-ci : `http://www.nextengine.com/`. Vous pouvez l'utiliser pour scanner vos propres objets. Pour cela, demandez à vos encadrants.

Remarque sur le choix des paramètres : Labatut, Pons et Keriven précisent qu'ils utilisent toujours $\lambda = 5, \alpha = 32$ et $\sigma = 0.5*$ la diagonale médiane de la plus petite boîte englobante de $\mathcal{P}$. D'après vos tests, ces valeurs sont-elles effectivement optimales ?

Remarque concernant les performances : d'après Labatut, Pons et Keriven, leur implémentation est *en pratique* presque linéaire en temps et en place mémoire par rapport au nombre de points de $\mathcal{P}$. Cependant, ils utilisent des algorithmes très performants pour le calcul de la tétraédrisation de Delaunay et de la coupe minimale. A quelle performance pouvez-vous parvenir ?

8 Livrable attendu

Le livrable sera rendu sur Teide sous forme d'archive *non plate* au format .tgz, et devra contenir les éléments suivants :

- un fichier README décrivant l'architecture de votre archive ;
- le code **sans** fichiers binaires ;
- un diagramme de classes ;
- une documentation Doxygen.

Bibliographie

[1] P. Labatut, J.-P. Pons, R. Keriven. *Robust and efficient surface reconstruction from range data*. Computer Graphics Forum, 2009.

[2] A. Bowyer. *Computing Dirichlet tessellations.* The Computer Journal, 24, pp. 162–166, 1981.

[3] D.F. Watson. *Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes.* The Computer Journal, 24, pp. 167–172, 1981.

[4] L.R. Ford, D.R. Fulkerson, *A simple algorithm for finding maximal network flows and an application to the Hitchock problem*, Rand Report, Rand Corporation, 1955. *Détaillé dans tout bon ouvrage d'optimisation combinatoire.*

[5] Cours d'optimisation combinatoire, 2A Ensimag MMIS, Zoltan Szigeti, 2013.

[6] S. Katz, A. Tal, R. Basri. *Direct visibility of point sets*. ACM Transactions on Graphics (Proceedings of SIGGRAPH), 26(3), 2007.