Projets de programmation
Algorithmes géométriques, théorie et pratique, SSTIM-VIM

Les sujets peuvent être pris en binôme ou individuellement. Il y aura une soutenance de 20mn dans notre bureau à l'INRIA. Merci de prendre rendez-vous avant le 13 février par courriel auprès d'un des trois enseignants (Pierre.Alliez@inria.fr, Olivier.Devillers@inria.fr, Monique.Teillaud@inria.fr), pour une soutenance dans la période du 18 au 22 février.

Vous aurez à fournir :
- votre code (archive zip ou tgz)
– un rapport bref (5 pages maxi) et
– un exposé de 20 mn intégrant une démo.

Ne répétez pas trop le cours ou le sujet, insistez sur les choix algorithmiques et d'implantation que vous faites.

1 Iteration de Lloyd en 2D

(Contact : Pierre Alliez)

L'algorithme de Lloyd consiste à alterner un déplacement des générateurs (les sommets de Delaunay) au centre de masse de leur cellule de Voronoi, avec une mise à jour de la partition de Voronoi. Après plusieurs itérations on obtient un diagramme de Voronoi centré où les générateurs coincident avec les centres de masses des cellules de Voronoi.

Vous pouvez partir du TP sur la triangulation de Delaunay 2D, qui affiche le diagramme de Voronoi et permet de générer des configurations prédéfinies (sommets sur un cercle, sur un carré). La classe CDT dérive de la triangulation de Delaunay 2D et peut être enrichie de fonctions pour implanter l'itération de Lloyd.

Dans un premier temps on ne regarde pas ce qu'il se passe au bord en considérant un domaine simple (un disque par exemple), en pla cant au préalable un grand nombre (100) de sommets sur le bord du disque qui restent fixes.

1. Implanter une fonction membre de CDT 2D qui calcule le centre de masse d'une cellule de Voronoi (en supposant une densité uniforme). Cette fonction doit prendre en paramètre un sommet (un Vertex handle) et renvoyer un point 2D. Dans le cas des cellules infinies vous pouvez renvoyer le point associé au sommet donné en paramètre.
2. Afficher les centres de masse des cellules de Voronoi finies.
3. Ajouter une fonction void lloyd() de la classe DT qui effectue une itération de Lloyd. vous pouvez dans une première boucle calculer toutes les nouvelles positions des générateurs dans une liste de points, puis mettre à jour toute la triangulation.

4. Comment pouvez-vous détecter la convergence de l'algorithme ? Ecrivez une fonction qui effectue plusieurs itérations de Lloyd jusqu'à convergence.
5. Dans le cas d'une fonction de densité non uniforme que devient la formule du centre de masse ? comment peut-on en faire un calcul approché ? quel algorithme proposez vous pour le calculer ?
6. Ajouter une fonction pour calculer l'énergie minimisée par l'iteration de Lloyd.
7. Ajouter une fonction pour mesurer l'isotropie moyenne des cellules de Voronoi.
8. Comment peut-on mesurer la vitesse de convergence de l'itération de Lloyd ?
9. Avancé : Proposer une manière (il y en a plusieurs) de déplacer les sommets du bord du domaine en les laissant sur le bord.
10. Avancé : Quelles alternatives sont possibles pour obtenir un diagramme de Voronoi centré òu les cellules de Voronoi infinies sont intersectées avec le domaine. Que peut-il advenir si le domaine n'est pas convexe ?
11. Avancé : Etant donnée une fonction de densité uniforme, quelles pistes proposez-vous pour accélérer la convergence de l'algorithme ?

2 Crust 2D

(Contact : Olivier Devillers)

Le crust est une méthode pour reconstruire une approximation du contour d'un objet 2D à partir de points connus sur le bord de cet objet. Cette méthode a ete exposée en cours.

Écrire un programme qui implante la méthode crust. On devra pouvoir saisir des points à la souris, et afficher le crust des points saisis, ou lire les points dans un fichier texte.

Dans une première version le crust sera recalculé entièrement à chaque nouveau point, dans une seconde version on ne fera que les modifications nécessaires. Commenter les différences de complexité entre les deux approches.

3 Énumération des voisins

(Contact : Olivier Devillers)

Étant donné un ensemble de points du plan coloriés S et une requête (q, c) où q est un point et c est une couleur, on veut énumérer les points de S dans l'ordre de la distance à q jusqu'à ce que l'on ait trouvé un voisin de couleur c. La complexité doit dépendre du nombre de points effectivement énumérés (il n'est pas question de trier tous les points par distance à q puis de parcourir le résultat de ce tri).

La triangulation de Delaunay permet de faire ça (on a fait en cours des exercices similaires) :

- on cherche le plus proche voisin de q (le 1er voisin de q)
- pour $i \geq 2$, le i-ème voisin de q est le plus proche de q parmi les voisins des $i-1$ premiers voisins de q dans la triangulation de Delaunay.

Écrire un programme qui implante cela. On devra pouvoir saisir des points de différentes couleurs à la souris, et saisir des requêtes (une interface très simple sera très bien). On mettra alors en évidence à l'affichage, un à un, les points dans l'ordre de la distance à q jusqu'à ce que l'on ait trouvé le bon.

4 Approximation robuste d'une fonction distance a une courbe en 2D

(Contact : Pierre Alliez)

A partir d'un ensemble de points 2D mesures sur une courbe, et entaches d'incertitude (bruit, points aberrants) on peut calculer une approximation d'une fonction distance a la courbe inferree en evaluant une fonction scalaire qui renvoie la somme des carres des distances aux K plus proches voisins (ici K est un parametre, voir `ftp://ftp-sop.inria.fr/geometrica/alliez/signing.pdf`, section 2.1). Si K est assez grand (typiquement 30) la fonction est robuste au bruit et aux points aberrants.

A partir des composants CGAL Constrained Delaunay triangulation 2D et K neighbor search `http://www.cgal.org/Manual/latest/doc_html/cgal_manual/Spatial_searching/Chapter_main.html#Subsection_59.3.1` (ou nearest neighbor Delaunay 2, `http://www.cgal.org/Manual/latest/doc_html/cgal_manual/Point_set_2_ref/Function_nearest_neighbors.html#Cross_link_anchor_1607`) implanter un algorithme qui affiche cette fonction interpolée linéairement sur les triangles d'une triangulation de Delaunay contrainte obtenue par insertion de tous les points fournis en entrée, puis raffinée par raffinement de Delaunay. Notez qu'il faut insérer 4 arêtes de contrainte sur une boîte englobant les points d'entrée avant raffinement pour que le raffinement démarre.

Affichez ensuite les lignes de niveaux de cette fonction en itérant sur les triangles de ladite triangulation. On doit pouvoir visualiser N lignes de niveaux.

Experimentez avec des données plus ou moins denses, plus ou moins bruitées, avec ou sans points aberrants, en faisant varier K, puis commentez.

Dans le programme on devra pouvoir saisir des points soit un par un, soit en déplacant la souris le long d'une courbe imaginaire. On doit pouvoir aussi ajouter des points aberrants choisis aléatoirement.

5 Diagramme de Voronoi borné 2D

(Contact : Pierre Alliez)

Le diagramme de Voronoi borné est le pseudo-dual de la triangulation de Delaunay contrainte.

On peut le construire à partir de la triangulation contrainte en marquant tous les triangles ("aveugles" ou non) dont le sommet de Voronoi dual n'est pas visible depuis ce triangle, la visibilité étant bloquée par une ou plusieurs contraintes. Pour chaque sommet de Delaunay on peut ensuite construire sa cellule de Voronoi duale en circulant sur ses triangles incidents pour assembler des arêtes de Voronoi soit en reliant deux sommets de Voronoi dual de deux triangles successifs t_1 et t_2 (dans le cas général), soit en intersectant le segment de Voronoi dual de l'arête entre t_1 et t_2 avec l'arête de contrainte la plus proche (ou les deux arêtes de contraintes les plus proches) bloquant la visibilité (d'autres cas se produisent au bord de l'enveloppe convexe lorsque les arêtes de Voronoi sont des rayons ou des droites).

La méthode précédente est naïve car procède à des tests exhaustifs sur les contraintes. En remarquant que l'ensemble des triangles aveugles est un ensemble de composantes connexes de triangles incidentes aux arêtes de contraintes, on peut pre-calculer le marquage des triangles aveugles en itérant sur les contraintes, et en propageant ledit marquage de triangle en triangle en partant d'un cote puis de l'autre de chaque contrainte (voir `ftp://ftp-sop.inria.fr/geometrica/alliez/imr16-meshing.pdf`). On peut en profiter pour stocker dans chaque triangle aveugle la contrainte qui bloque sa visibilité.

Écrire un programme qui calcule et affiche le diagramme de Voronoi borné. On devra pouvoir saisir des segments de contraintes et/ou les charger a partir d'un simple fichier texte.