Algorithme de Toussaint résolvant le problème du rectangle minimum

Pablito Bello

Janvier 2020

SORBONNE UNIVERSITÉ

Résumé : Étant donné un ensemble fini P de points dans $\mathbb{R}^2$, le rectangle minimum correspond au plus petit rectangle contenant tous les points de P. Le cercle minimum est le plus petit cercle contenant tous les points de P. On propose ici une étude de l'algorithme de Toussaint qui permet de calculer le rectangle minimum, ainsi que l'algorithme de Ritter qui lui permet de calculer une approximation du cercle minimum.

Mots-clés : Rectangle minimum, cercle minimum, enveloppe convexe, Toussaint, Ritter.

Table des matières

Première partie

Introduction

Ce rapport de recherche a pour but d'étudier les algorithmes de Toussaint et de Ritter qui permettent de répondre respectivement aux problèmes du rectangle minimum et d'approximation du cercle minimum d'un ensemble de points.

Ces structures sont particulièrement utiles dans les algorithmes de détection de collisions entre deux ensembles de points, qui peuvent par exemple être appliqués aux domaines de la simulation physique ou aux jeux vidéos. Notre objectif sera d'implémenter ces deux algorithmes et d'étudier leurs forces et faiblesses, tout en les comparant.

Nous réaliserons des tests de vitesse et de qualité des conteneurs obtenus en utilisant la base de tests de VAROUMAS. Ensuite, nous pourrons conclure, selon les besoins en vitesse d'exécution et en précision de calculs, quel est le meilleur des deux algorithmes à utiliser.

Deuxième partie

Notions utiles

Enveloppe convexe : L'enveloppe convexe d'un ensemble de points est l'ensemble convexe le plus petit qui contient tout les points de l'ensemble. Dans le cadre de cette étude, nous utiliserons l'algorithme de Graham[1] pour calculer l'enveloppe convexe.

Aire d'un polygone : Soit un polygone convexe de coordonnées $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)$

$$Aire = \frac{1}{2} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \vdots & \vdots \\ x_n & y_n \\ x_1 & y_1 \end{vmatrix} = \frac{1}{2}[(x_1y_2+x_2y_3+x_3y_4+\cdots+x_ny_1)-(y_1x_2+y_2x_3+y_3x_4+\cdots+y_nx_1)]$$

Aire d'un rectangle : $Aire = longueur \times largeur$

Aire d'un cercle : Soit un cercle de rayon r, alors $Aire = r^2 \times \pi$

Troisième partie
Algorithme de Toussaint

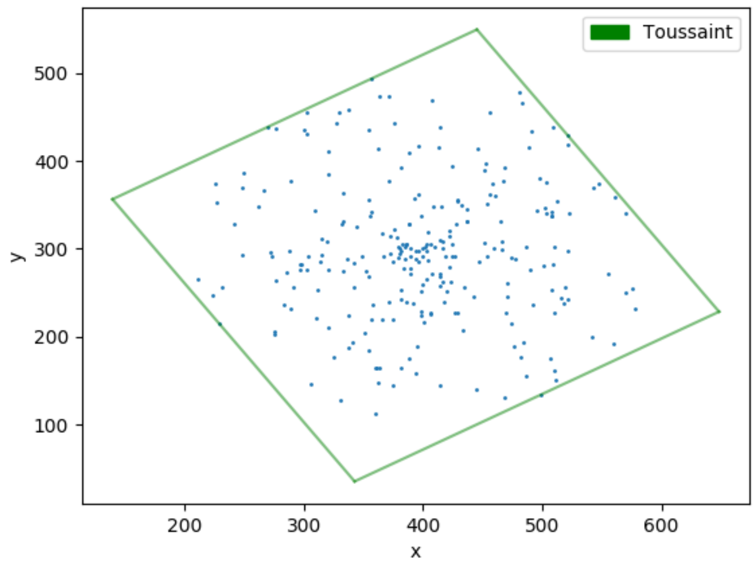

FIGURE 1 – Rectangle minimum

L'algorithme de Toussaint [2] [3] permet de résoudre le problème du rectangle minimum contenant un ensemble de points.

Ce problème peut être résolu par un algorithme dit "naif" en calculant les quatre points extrêmes de l'ensemble de points (point d'abscisse minimum, d'abscisse maximum, d'ordonnée minimum et d'ordonnée maximum) puis en traçant les droites parallèles deux à deux (passant par ces points), qui forment un rectangle contenant l'ensemble des points. Néanmoins, ce rectangle n'est pas le rectangle minimum, et la complexité de cet algorithme est en $O(n^2)$.

Un autre algorithme qui permet de résoudre ce problème est l'algorithme de Freeman et Shapira [4]. Cependant la compléxité totale de celui-ci est aussi en $O(n^2)$.

Étapes de l'algorithme de Toussaint : On peut diviser l'algorithme de Toussaint en 8 étapes :

1. On calcul l'enveloppe convexe avec l'algorithme de Graham.
2. On trouve les points i, j, k, l lesquels forment les paires antipodales de l'enveloppe convexe.
3. On trace les droites L_i, L_j, L_k, L_l passant par les points i, j, k, l. On calcul le rectangle formé par ces droites et le sauvegarde en tant que rectangle minimum.
4. On trouve l'angle minimum formé par une des droites et un coté de l'enveloppe convexe.
5. On effectue une rotation des droites par l'angle minimum.
6. On créer le nouveau rectangle à partir de la nouvelle position des droites et calcule son aire. Si l'aire du nouveau rectangle est inférieur à l'aire de l'ancien rectangle, il devient le nouveau rectangle minimum.
7. On répète les étapes 4 à 6 jusqu'à avoir fait un tour complet.
8. On retourne le rectangle minimum.

Compléxité : L'algorithme de Toussaint est un algorithme de compléxité $O(n)$, ce qui est meilleur que les deux algorithmes précédents.

Quatrième partie
Algorithme de Ritter

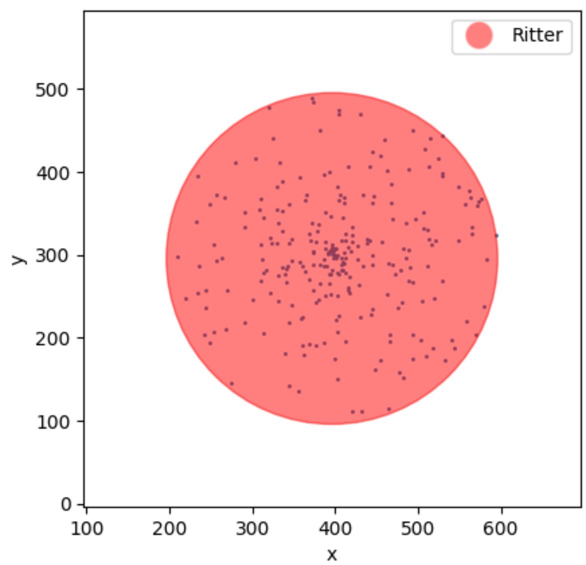

FIGURE 2 – Cercle minimum

L'algorithme de Ritter [5] [6] est un algorithme permettant de donner une approximation au problème du cercle minimum contenant un ensemble de points.

Ce problème peut être résolu par un algorithme dit "naif" en considérant les cercles dont les diamètres sont les segments formés par les points pris deux à deux en parcourant l'ensemble de points, ainsi que les cercles circonscrits aux triangles formés par les points pris trois par trois. On cherche le plus petit de ces cercles contenant l'ensemble de points. Cet algorithme a une complexité en $O(n^4)$, ce qui n'est pas satisfaisant.

Il existe beaucoup d'autres algorithmes traitant de ce problème [7].

Étapes de l'algorithme de Ritter On peut diviser l'algorithme de Ritter en 5 étapes :

1. Prendre un point p_1 dans l'ensemble de points P
2. Chercher un point p_2 dans P, de distance maximale avec p_1.
3. Chercher un point p_3 dans P, de distance maximale avec p_2.
4. Créer un cercle C, de centre (p_2, p_3) et de rayon $\frac{1}{2}distance(p_2, p_3)$.
5. Si tous les points sont dans C, le retourner. Sinon construire un nouveau cercle couvrant l'ancien cercle et le point en dehors du cercle jusqu'à ce qu'il n'y ait plus de point en dehors du cercle.

Compléxité : L'algorithme de Ritter est un algorithme de compléxité $O(n \times r)$ avec n le nombre de points et r la taille de l'espace dans lequel on se situe. Il ne calcule pas le cercle minimum (le cercle trouvé est entre 5% et 20% plus grand que le minimum), mais est très utilisé grâce au fait qu'il est simple à implémenter et qu'il donne tout de même de bons résultats.

Cinquième partie

Tests de performance

Les algorithmes sont implémentés en Python 3 et testés via la base de tests VAROUMAS fournie. On comparera notamment le temps d'exécution des deux algorithmes, ainsi que la qualité des conteneurs par rapport à l'enveloppe convexe.

1 Temps d'exécution

On va calculer le temps d'exécution des algorithmes de Toussaint et de Ritter en secondes. On appliquera chaque algorithme à des ensembles de points de taille 256 à 425 728. Néanmoins, le langage Python n'étant pas des plus rapides et la machine utilisée pas des plus performantes, chaque taille ne sera pas évalué. On appliquera un gap de 1000, les tailles évalués seront donc $256, 1256, 2256, \ldots, 425728$.

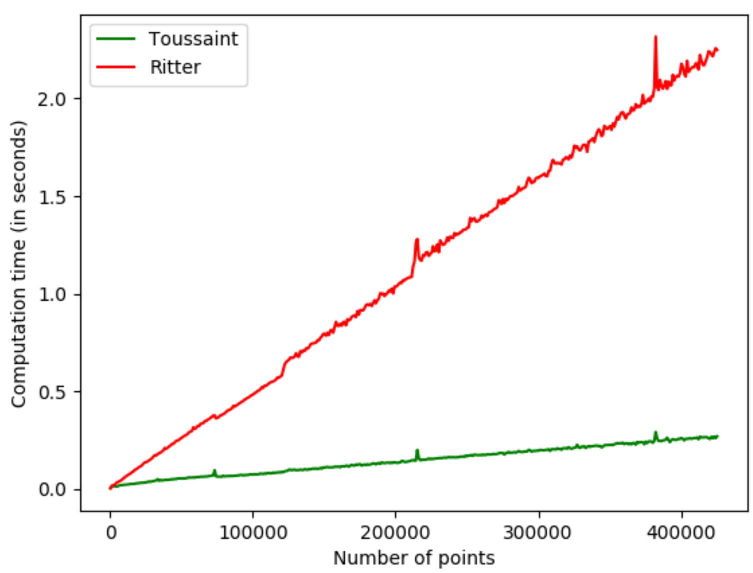

FIGURE 3 – Temps d'exécution par rapport au nombre de points.

On peut remarquer que l'algorithme de Toussaint est plus rapide que celui de Ritter, en accord avec leur complexité respectives.

2 Qualité des conteneurs

On va maintenant s'intéresser à la qualité des conteneurs, définis par la formule suivante : $qualite = \left(\frac{AireDuConteneur}{AireDeL'enveloppeConvexe}\right) - 100\%$.

On appliquera chaque algorithme à chaque test de la base en calculant la qualité de celui-ci. On calculera également la moyenne et l'écart type des résultats obtenus pour chaque algorithme.

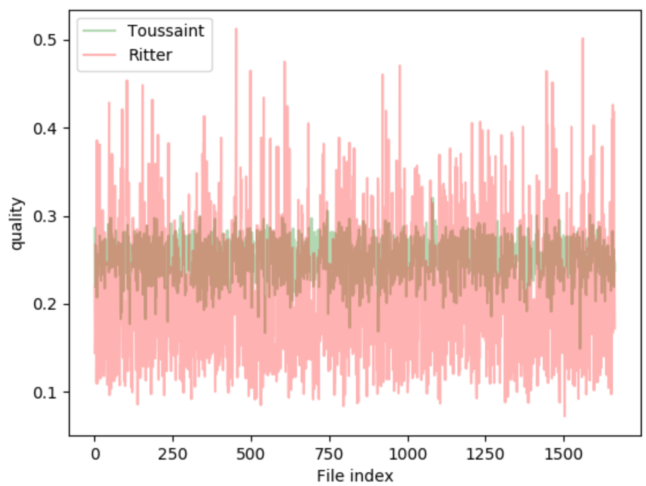

FIGURE 4 – Qualité des algorithme pour chaque test.

La moyenne de qualité pour l'algorithme de Toussaint est de 0.25. Pour l'algorithme de Ritter elle est de 0.20.

L'écart type est de 0.02 pour l'algorithme de Toussaint, et de 0.07 pour l'algorithme de Ritter.

On peut voir qu'en moyenne, l'algorithme de Ritter trouve un plus petit conteneur (seulement 20% plus gros que l'enveloppe convexe, contre 25% pour Toussaint). On peut également constater, que la taille des conteneurs trouvés par l'algorithme de Toussaint est plus stable que pour l'algorithme de Ritter, cela est s'explique notamment par la forme de l'enveloppe convexe. Au plus, la forme de l'enveloppe convexe s'éloigne de celle d'un polygone régulier, au plus la qualité du conteneur trouvé par l'algorithme de Ritter sera mauvaise, celui-ci étant un cercle.

Sixième partie

Conclusion

À partir des travaux réalisés, on peut voir que l'algorithme de Toussaint est plus rapide que celui de Ritter, mais que celui de Ritter donne en moyenne un conteneur plus petit. Néanmoins, au vu du compromis à faire entre temps d'exécution et qualité du conteneur, si le temps d'exécution importe d'avantage, il serait plus approprié d'utiliser l'algorithme de Toussaint. Il est également à noter que le conteneur trouvé par l'algorithme de Toussaint est en général plus proche de la forme de l'enveloppe convexe.

L'enveloppe convexe est quant à elle plus précise. Cependant, contrairement au rectangle et au cercle pour lesquels on peut évaluer les collisions en temps constant, ce n'est pas le cas avec des polygones. Il faut donc choisir la structure appropriée selon le cas d'utilisation.

Dans le domaine des jeux vidéo, on peut se permettre une précision approximative, mais on a besoin en général d'un temps de calcul très rapide. Il serait donc plus approprié d'utiliser l'algorithme de Toussaint.

Si on a besoin de précision dans les calculs de collisions entre ensembles de points, l'enveloppe convexe est sûrement plus appropriée, malgré un temps de calcul de collision plus mauvais entre polygones.

Références

[1] Wikipedia contributors. Graham scan — Wikipedia, the free encyclopedia, 2020. [Online ; accessed 3-January-2020].

[2] Wikipedia contributors. Rotating calipers — Wikipedia, the free encyclopedia, 2019. [Online ; accessed 3-January-2020].

[3] Godfried T. Toussaint. Solving geometric problems with the rotating calipers. *Proc.MELECON*, 1983.

[4] H. Freeman and R. Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Commun. ACM*, 18(7) :409–413, July 1975.

[5] Wikipedia contributors. Bounding sphere — Wikipedia, the free encyclopedia, 2019. [Online ; accessed 3-January-2020].

[6] Jack Ritter. An efficient bounding sphere. 1990.

[7] Wikipedia contributors. Smallest-circle problem — Wikipedia, the free encyclopedia, 2019. [Online ; accessed 3-January-2020].