ALGAV
RAPPORT DE PROJET

Résolution du problème de rectangle et cercle minimum

Veyrack LIN

January 12, 2020

Table des matières

1 Introduction

Dans ce rapport, nous étudions le problème des conteneurs englobant un ensemble de points. En effet ce problème est récurrent dès que l'on s'intéresse aux collisions entre deux objets. Lorsque l'on souhaite savoir si deux objets sont en collision sur un plan 2D il faut savoir si au moins un pixel d'un des deux objet est inclus dans le second objet. Il est assez trivial d'effectuer cette vérification si les objets sont des cercles ou des rectangles. Par exemple pour les cercles il suffit de savoir si la distance entre le point et le centre du cercle est inférieure au rayon de celui-ci. Dans des cas plus complexes, l'objet n'est pas réduit à un cercle ou à un rectangle. On peut alors englober cet objet avec une enveloppe convexe, mais la collision reste difficile à détecter. Il serait inenvisageable de comparer chaque pixel des deux objets afin de savoir s'ils sont ou non en collision. Pour palier à ce problème nous allons voir deux méthodes qui permettent d'obtenir des conteneurs englobant l'objet ainsi que leur qualité en tant que tel. Dans la première partie, nous verrons les rectangles minimum. Puis, dans le seconde partie, les cercles minimum.

2 Toussaint

2.1 Algorithme

Nous allons commencer avec l'algorithme de Toussaint. Ce dernier permet d'obtenir un rectangle d'aire minimum contenant un ensemble de points.

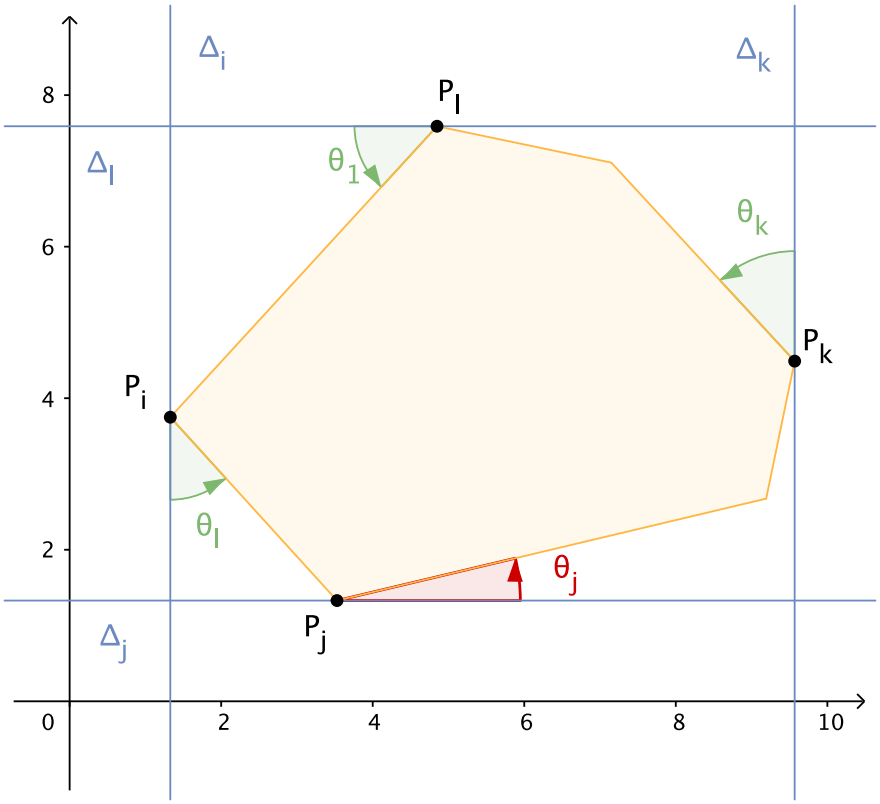

Figure 1: Fonctionnement de l'algorithme de Toussaint (1)

Le principe est de partir de l'enveloppe convexe formée par l'ensemble de points puis de générer autour de cette enveloppe des rectangles à la suite en effectuant des rotations jusqu'à obtenir le rectangle d'air minimum. (2) (3)
Son implémentation est la suivante :

1. Trouver les points d'abscisse et d'ordonnée minimale et maximale. On appellera ces points P_i,P_j,P_k,P_l

2. A partir de ces points, construire les droites horizontales et verticales passant par P_i,P_j,P_k,P_l nommées Δ_i,Δ_j,Δ_k,Δ_l.

3. Trouver l'angle θ minimum entre les droites Δ et les côtés de l'enveloppe convexe. (4)

4. Une fois l'angle minimum trouvé, il faut effectuer la rotation des quatre droites Δ de l'angle θ (5)

5. Avec ces quatre nouvelles droites on crée un nouveau rectangle.

6. On calcule l'aire du nouveau rectangle et si elle est inférieure à l'ancienne on stocke ce nouveau rectangle.

7. On répète les étapes 3-6 pour tout les cotés de l'enveloppe convexe.

Certains choix d'implémentation ont été effectués afin de faciliter celle-ci :
-Pour les droites nous utilisons comme définition un *Point* ainsi qu'un $Vecteur$ directeur.
-Pour les rectangles, nous pouvons soit, utiliser quatre points ordonnés qui correspondent aux quatre coins du rectangle, ou alors quatre droites ordonnées et calculer les intersections pour obtenir le rectangle.
-Pour calculer l'angle on utilise cette formule : $\theta = \arccos(\frac{\vec{u}.\vec{v}}{||\vec{u}||.||\vec{v}||})$
avec $\vec{u}.\vec{v}$ le produit scalaire des deux vecteurs
et $||\vec{u}||$ la norme du vecteur (6)

2.2 Analyse

Afin de tester la performance de ce programme, voyons tout d'abord la complexité théorique du temps d'exécution de cet algorithme. L'algorithme de Toussaint effectue lors de son exécution en premier lieu le calcul de l'enveloppe convexe via l'algorithme de *Graham* avec son *pixelsort*. Cela est réalisé en $O(n)$.

Par la suite on calcule les points extrêmes (c'est à dire les points d'abscisse et d'ordonnée minimale et maximale) en parcourant une fois l'ensemble des points. Nous avons donc un $O(n)$.

Ensuite, on effectue des calculs basiques pour chaque coté de l'enveloppe convexe, ce qui nous donne une complexité de $O(n)$

Au final, nous avons une complexité en $O(n + n + n) = O(n)$

En pratique, en utilisant une base de test crée contenant 1664 instances dont les points vont de 256 à 299 776, nous obtenons cette courbe :

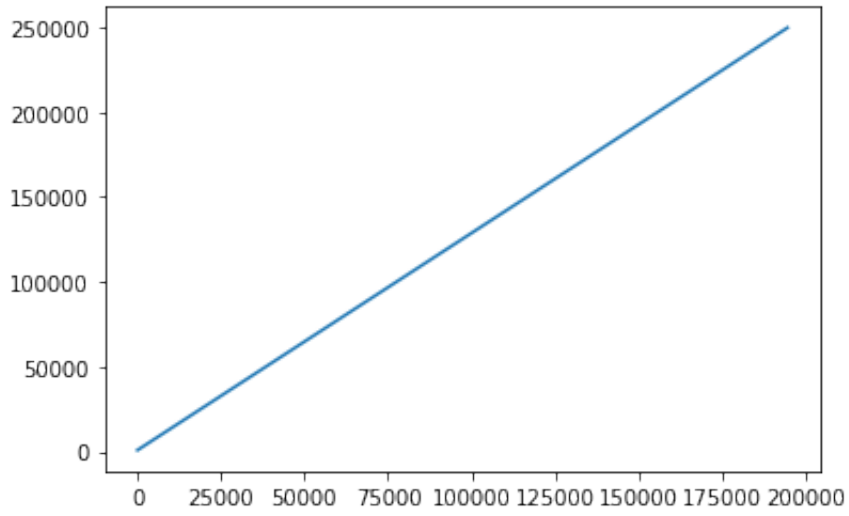

Figure 2: Temps d'exécution(ms) en fonction du nombre de points

On remarque donc que l'algorithme a effectivement un temps d'exécution linéaire.

Pour le test de qualité du rectangle par rapport à l'enveloppe convexe, nous utilisons la formule donnée : qualité $= \frac{aireRectangle}{airePolygone} - 100\%$

L'aire du polygone obtenu à partir de l'ensemble de points se calcule de la façon suivante en prenant les points du polygone triés dans le sens trigonométrique usuel notés $(x_1, y_1), (x_2, y_2), (x_3, y_3)...(x_n, y_n)$

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

Grâce à cette formule on peut donc évaluer la qualité et ainsi nous obtenons les données suivantes en utilisant la base de test de Varoumas comprenant 1664 instances de 256 points :

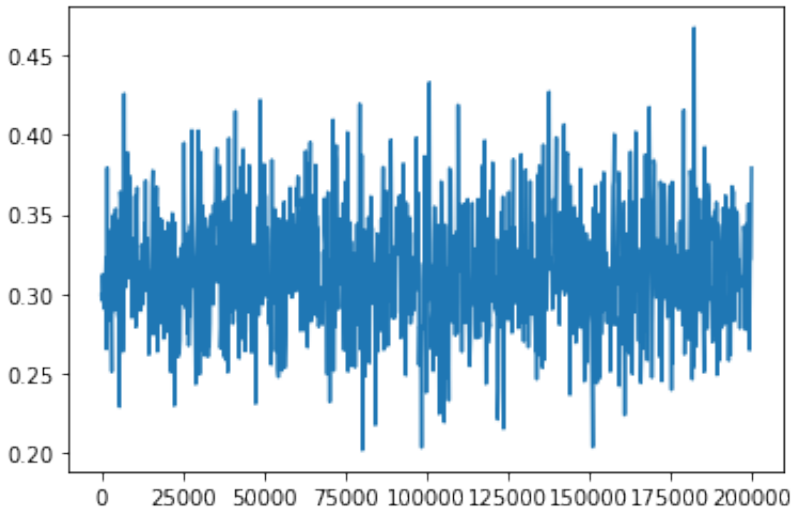

Figure 3: Qualité du conteneur en fonction des instances

La qualité du rectangle englobant est donc plus ou moins compris entre 0.25 et 0.40. Cela signifie qu'en moyenne l'aire du rectangle est entre 25 et 40% supérieure à celle de l'enveloppe convexe.

3 Ritter

3.1 Algorithme

Pour l'algorithme de Ritter, il s'agit aussi d'obtenir un conteneur minimum, mais cette fois ci avec un cercle contenant tous les points.(7) (8)

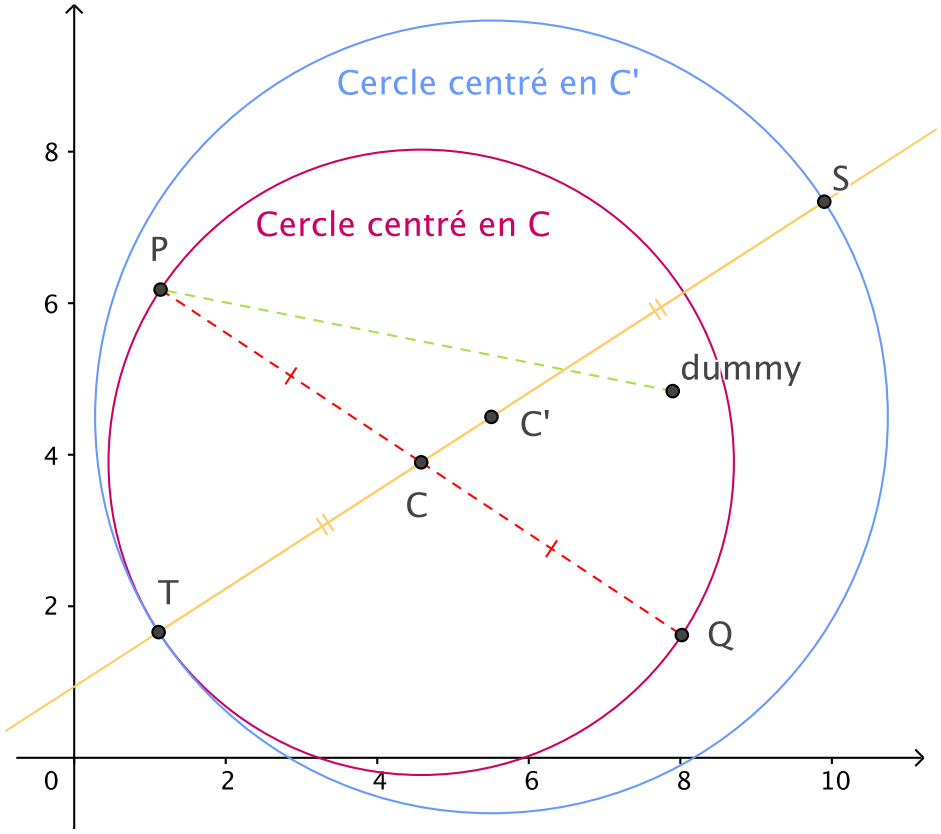

Figure 4: Fonctionnement de l'algorithme de Ritter

Pour parvenir à cela nous procédons ainsi :

1. Choisir un point aléatoire dans l'ensemble des points, nommé *dummy*.
2. Parcourir l'ensemble pour trouver le point P de distance maximum par rapport au point *dummy*.
3. Parcourir de nouveau l'ensemble pour trouver le point Q de distance maximum par rapport au point P.

4. Calculer le point C, centre du segment PQ.
5. Construire le cercle de centre C de rayon $|PC|$ (Il passe donc par P et Q).
6. Parcourir l'ensemble afin d'enlever tout les points qui sont contenus dans le cercle précédemment construit.
7. Si l'ensemble n'est pas vide, on choisit de nouveau un point aléatoirement nommé S.
8. Construire la droite passant par S et C. Cette dernière coupe le cercle en deux points. Prenons T le point le plus éloigné de S. (9)
9. A partir du segment ST, calculer le point C' centre de celui ci. (10)
10. Construire le cercle de centre C' et de rayon $C'S$.
11. Répéter les étapes de 6 à 10 jusqu'à qu'il ne reste plus de point dans l'ensemble.

3.2 Analyse

Dans cette analyse théorique de l'algorithme de Ritter nous pouvons voir que dans ce dernier on parcourt dans un premier temps à deux reprises l'ensemble des points afin de trouver un point de distance maximum par rapport à un autre point. Nous avons donc déjà un $O(2n)$. Ensuite, on effectue une boucle qui retire certains points de l'ensemble un nombre c fois. Pour retirer un point on doit parcourir l'ensemble. Cela se fait en $O(n)$. Cette boucle se fait donc en $O(cn)$. Au final la complexité de l'algorithme est en $O(2n + cn) = O(cn)$.

Lors de l'expérience pratique, en utilisant une nouvelle fois la même base de test que celle pour l'algorithme de Toussaint, on obtient le graphe suivant :

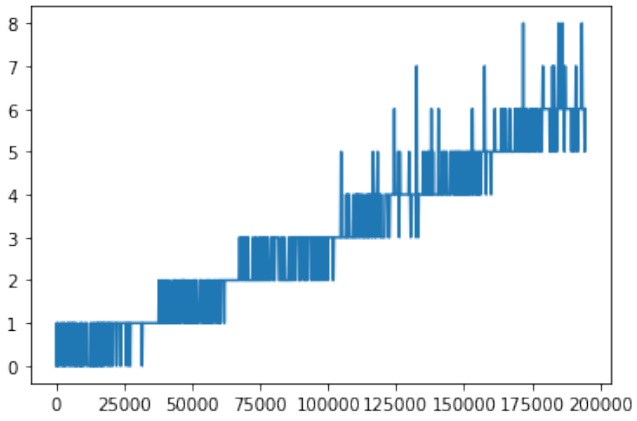

Figure 5: Temps d'exécution(ms) en fonction du nombre de points

Le temps d'exécution de cet algorithme est donc aussi en temps linéaire mais avec un coefficient bien moins élevé que celui de l'algorithme de Toussaint. Nous allons les comparer dans la partie suivante.

Pour le test de qualité du cercle minimum, nous utilisons la même formule que définit précédemment ainsi que la base de test qui a été crée. Avec ceci nous obtenons donc ce graphe :

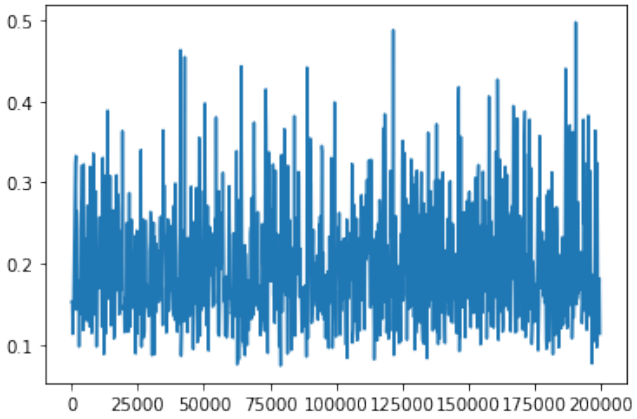

Figure 6: Qualité du conteneur en fonction des instances

La qualité du cercle englobant est donc plus ou moins compris entre 0.10 et 0.30. Cela signifie qu'en moyenne l'aire du rectangle est entre 10 et 30% supérieure à celle de l'enveloppe convexe.

4 Comparaison

Après avoir expliqué, réalisé et analysé ces algorithmes nous allons voir la comparaison de ceux-ci. Nous avons vu que les deux algorithmes étaient en $O(n)$ pour la complexité du temps d'exécution. Malgré cela, leurs coefficients sont différents ce qui rend leur temps d'exécution bien différent aussi comme nous le voyons sur ce graphe :

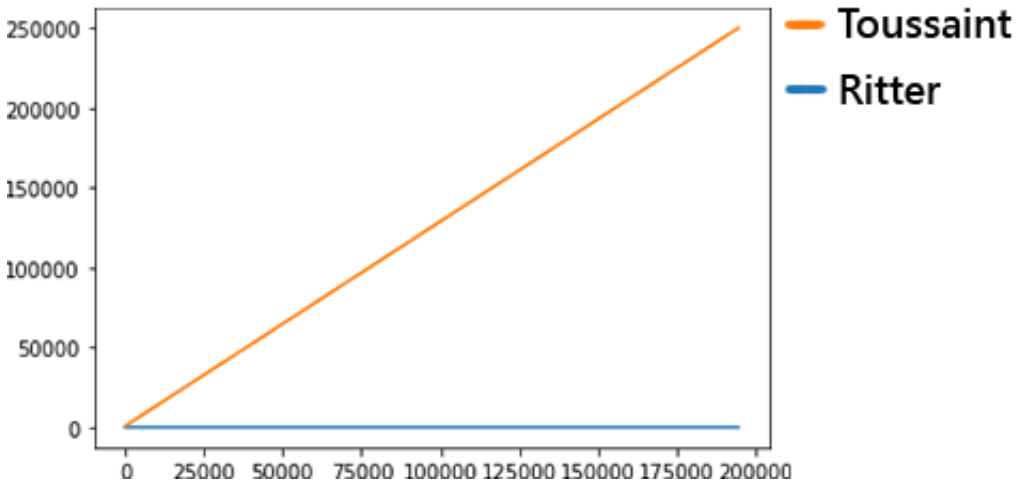

Figure 7: Temps d'exécution(ms) en fonction du nombre de points

Nous pouvons donc voir que l'algorithme de Ritter est bien plus efficace en termes de temps d'exécution que celui de Toussaint. En effet le calcul de l'enveloppe convexe ralentit considérablement le déroulement de l'algorithme.

Quant à la qualité en tant que conteneur de ces objets, lorsque l'on compare les deux algorithmes voila ce qui en découle :

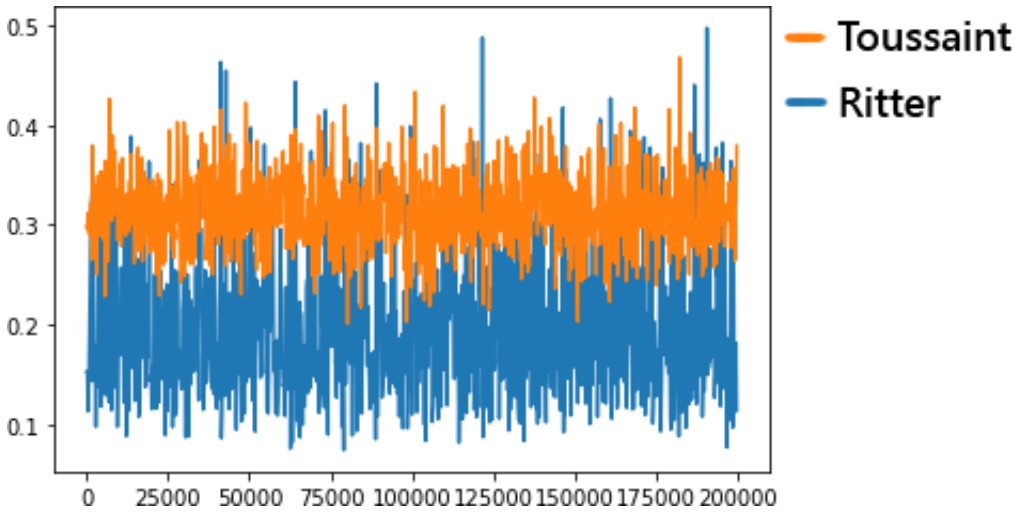

Figure 8:

On peut voir qu'en moyenne le rapport de la qualité de l'algorithme de Toussaint est plus grande que celle de Ritter, c'est à dire que le pourcentage de différence est plus élevé. Ainsi, l'algorithme de Ritter fourni un meilleur conteneur minimum que Toussaint.
Malgré cela, il arrive que l'algorithme de Toussaint soit plus efficace que celui de Ritter. En effet dans certains cas si la forme de l'enveloppe convexe se rapproche d'une ellipse où la largeur est nettement supérieure à la hauteur (ou inversement) alors le rectangle minimum est plus optimisé par rapport au cercle.

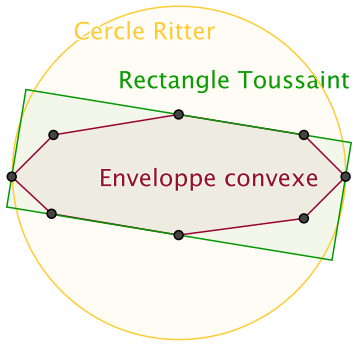

Figure 9: Cas où Toussaint est meilleur que Ritter

5 Conclusion

Nous avons pu voir durant ce projet portant sur les conteneurs minimum d'un ensemble de points deux algorithmes différents donnant des objets englobant l'ensemble des points. D'une part nous avons vu l'algorithme de Toussaint fournissant un rectangle minimum et d'autre part celui de Ritter fournissant un cercle minimum. Ces deux algorithmes sont utilisés dans de nombreux domaines à la place des enveloppes convexes pour certaines raisons. En effet l'enveloppe convexe est un très bon conteneur, mais le problème réside dans son efficacité. Nous préférerons donc perdre en qualité de conteneur, mais gagner en rapidité d'exécution en utilisant les algorithmes vus au cours de ce projet. Par ailleurs, d'après les analyses effectuées pour ces algorithmes on peut s'apercevoir que l'algorithme de Ritter est plus efficace que celui de Toussaint, car ce dernier nécessite un pré-calcul de l'enveloppe convexe. On préférera donc utiliser l'algorithme de Ritter.

References

[1] https://vincentcordobes.github.io/ritter-toussaint/.

[2] https://geidav.wordpress.com/2014/01/23/computing-oriented-minimum-bounding-boxes-in-2d/.

[3] http://www.mathwords.com/a/area_convex_polygon.htm.

[4] https://fr.wikihow.com/calculer-l'angle-entre-deux-vecteurs.

[5] https://en.wikipedia.org/wiki/Rotation_(mathematics).

[6] https://fr.wikipedia.org/wiki/Norme_(mathématiques).

[7] https://fr.wikipedia.org/wiki/Problème_du_cercle_minimum.

[8] https://www.developpez.net/forums/d1553357/logiciels/microsoft-office/excel/probleme-cercle-minimum/.

[9] https://stackoverflow.com/questions/563198/how-do-you-detect-where-two-line-segments-intersect.

[10] https://www.kartable.fr/ressources/mathematiques/methode/determiner-les-coordonnees-du-milieu-dun-segment/3537.