凸包之最小矩框新解

國立台南藝術大學/國立台北大學
校長室/電機工程學系
詹景裕
gejan@ntpu.edu.tw

國立台北大學
電機工程學系
陳炫佑
s410387003@ntpu.edu.tw

摘要

凸包(Convex Hull)是指能在歐式幾何平面與空間中包覆一群散佈點的最小凸集合，且其面積或容積為最小。凸包在計算幾何中為一項重要的研究學問，被大量運用在圖形處理、地理資訊系統及軍武研發，可堪稱為指標性演算法。

本文中，我們以最小矩框的觀念提出全新獨創的凸包解法。藉由最小矩框取出其上的凸包節點，並以凸包節點所形成的多邊形來移除其餘非凸包點，使節點數由n降至$\frac{n}{\log n}$，接著將剩餘的節點代入任意時間複雜度為$O(n \log n)$之凸包演算法，便可在線性期望時間$O(n)$獲得凸包，其中n為節點總數。

關鍵字:凸包、最小矩框

1 緒論

凸包是數學和幾何學中一個重要的議題。許多著名議題和其相關，如Voronoi diagrams [2]，三角剖分 [6]。這些方法在很多領域中運用:影像處理 [8]，地理資訊系統(GIS) [7]，軍事用途比方無人駕駛的地面載具路徑規劃及巡弋飛彈在多面體上的路徑規劃 [9]等。

在1996年，Barber等人提出Quick Hull Algorithm [3]，運用凸包點組成多邊形，除去多邊形內非凸包點的方式，快速減少圖形中的點，解決凸包問題。但在極端狀況下的時間複雜度為$O(n^2)$，而我們選擇用最小矩框法[3]和Graham's Scan [4]來減少極端狀況下需要的時間。

2 基本概念及背景

本節我們以最小矩框法 [1]和 Graham's Scan 為基礎，改善 Quick Hull Algorithm 降低解決凸包問題所需的時間。

最小矩框法由 AKL 等人在 1978 年提出，藉由找到 x 及 y 座標的最大及最小值以建立最小矩框。且利用矩框邊界必為凸包點的特性，移除內部的非凸包點，我們藉此減少 Graham's Scan 排序需計算的節點，以減少時間複雜度。

Graham's Scan 由 Graham 在 1972 年提出，是現今相當有效率解決凸包問題的辦法，藉由極座標的角度排序與所有節點平均點的距離解決凸包問題，其演算法時間複雜度為 $O(n \log n)$，其中 n 為節點數。在本演算法中，我們以 Graham's Scan 處理剩餘的非凸包點。

本演算法中，我們用旋轉矩陣建立不同的座標軸，以多種角度的最小矩框移除其內部的非凸包點。使用 Graham's Scan 處理剩餘的非凸包點並完成凸包，時間複雜度在平均狀況下為$O(n)$。

3 演算法及其圖解

在本節介紹演算法，本演算法用到最小矩框、旋轉坐標軸等概念，利用移除點集合中央非凸包點的方式，節省 Graham's Scan 排序所需的時間。能將 Graham's Scan 的時間複雜度$O(n \log n)$在平均分布狀況下減少至$O(n)$，而在極端情況下時間複雜度也同為$O(n \log n)$。Table 1 列出演算法中使用的變數。

Table1 The notations

G	The set of graph; $G = (V, E)$
V	The set of vertices; $V = \{v_i(x_i, y_i) \mid 0 \le i \le n-1\} = \{v_i(r_i, \theta_i) \mid 0 \le i \le n-1\}$
E	The set of edges; $E = \{e_{i,j}(v_i, v_j) \mid 0 \le i \le n-1, 0 \le j \le n-1\}$
G^{CH}	The set of convex hull; $G^{CH} = (V^{CH}, E^{CH})$
V^{CH}	The set of vertices on the convex hull; $V^{CH} = \{v_i^{CH}(x_i^{CH}, y_i^{CH}) \mid 0 \le i \le n-1\} = \{v_i^{CH}(r_i^{CH}, \theta_i^{CH}) \mid 0 \le i \le n-1\}$
E^{CH}	The set of edges on the convex hull; $E^{CH} = \{e_{i,j}^{CH}\ (v_i^{CH}, v_j^{CH}) \mid 0 \le i \le n-1,\ 0 \le j \le n-1\}$
n	The number of points.
S	The set of all points
S^{CH}	The set of points on convex hull
S^{UCH}	The set of points that are not on convex hull
S^{UK}	The set of undetermined points $S^{UK} = S - S^{CH} - S^{UCH}$
R	The set of points on the smallest enclosing rectangle; R^T: The points on the top edge of the rectangle R^B: The points on the bottom edge of the rectangle R^R: The points on the right edge of the rectangle R^L: The points on the left edge of the rectangle
E^P	The set of edges on the polygon in the rectangle
θ_i	The rotate angle
k	Number of rotations, normally $k \le 90$ for the first quadrant in a plane angle, so that a full rotation is 360 degrees.

3.1 演算法

本節介紹演算法的架構，包含 3 個 Function 和 1 個 Algorithm，Function 分別是 *FindSmallest EnclosingRectangle* 用於建立最小矩框 *Remove PointsInsidePolygon* 移除凸包點形成之多邊形內的多餘節點，*RotationOfMatrix* 取得新的座標軸來尋找更多的凸包點。Algorithm 最小矩框之凸包演算法則統整上述 Function 並呼叫時間複雜度為 $O(n \log n)$ 的凸包演算法 *Graham's Scan* 來獲得凸包。

Function 1 *FindSmallestEnclosingRectangle*()

Step 1 建立矩框

Step 1.1 從未確認的點集合S^{UK}中找到x及y座標的最大值和最小值的點，以建立最小包圍矩框。

Step 1.2 將矩框上四個邊的點分別放進矩框R的上、下、右、左 4 個矩框邊陣列內(R^T、R^B、R^R、R^L)。

Step 2 將四個陣列中分別保留最大和最小的點。

Step 3 得到矩框和矩框上的凸包點將其標示為凸包的點集合S^{CH}。

END{*FindSmallestEnclosingRectangle()*}

Function 2 *RemovePointsInsidePolygon*()

Step 1　用四個矩框邊陣列中的點數，得知多邊形邊數。

Step 2　用多邊形的邊E^P做不等式，得多邊形的區域方程式。

Step 3　將點集合S^{UK}代入區域方程式，以移除多邊形內的非凸包點。

END{*RemovePointsInsidePolygon()*}

Function 3 *RotationOfMatrix*()

Step 1　輸入S^{UK}和 S^{CH}的點座標和旋轉角度θ_i。

Step 2　所有點代入旋轉矩陣後取得新座標軸。

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

END{*RotationOfMatrix()*}

Algorithm: 最小矩框之凸包演算法()

Step 1　Call Function 1 *FindSmallest-EnclosingRectangle* 找到矩框，將其存到四個陣列內。

Step 2　Call Function 2 *RemovePoints-InsidePolygon* 用矩框上的點建立多邊形，將多邊形內部的點排除運算。

Step 3　Call Function 3 *RotationOfMa-trix* 旋轉坐標軸後回到 Step 1，直到旋轉 k 次或$S^{UCH} \leq n - (n/\log n)$。

Step 4　Call *Graham's Scan*(或任何時間複雜度為$O(n \log n)$的凸包演算法)，將可能剩餘的非凸包點排除。

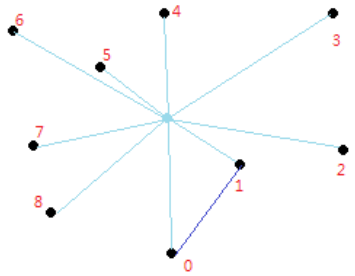
END{最小矩框之凸包演算法()}

3.2　演算法圖解

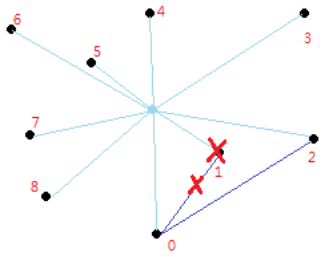
在本節中用圖形來呈現演算法的過程。Fig. 2 中，(a)輸入點集合(b)建立最小矩框，標出矩框上的點(c)將最小矩框上的點連成多邊形(d)移除非凸包的點(e)逆時針旋轉坐標軸建立新的最小矩框(f)找到多邊形，並將非凸包點移除(g)重複步驟(b)，直到旋轉 k 次或$S^{UCH} \leq n - (n/\log n)$。

4　效能分析

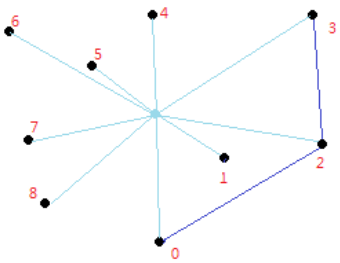
本節分為凸包之正確性和時間複雜度一一闡述如下。



(a)　角度排序

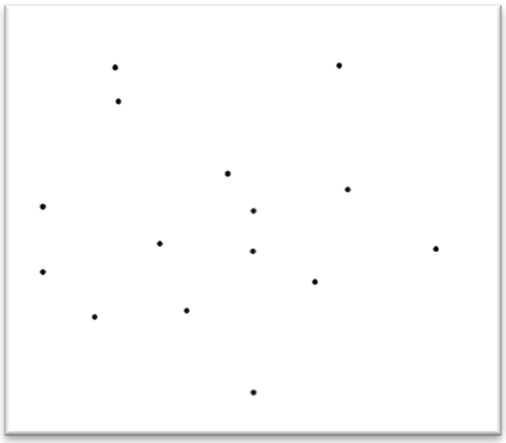


(b)　判斷是否凹陷

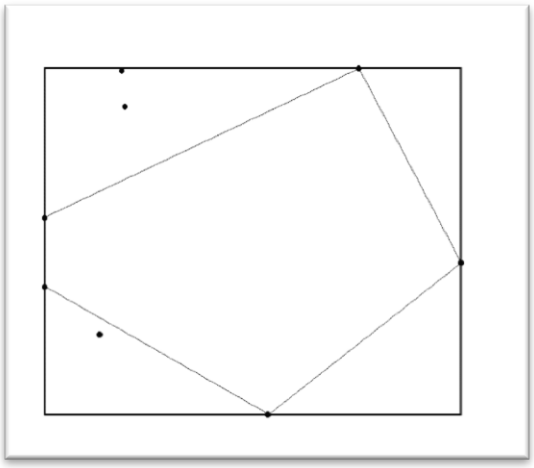


(c)　修正凸包

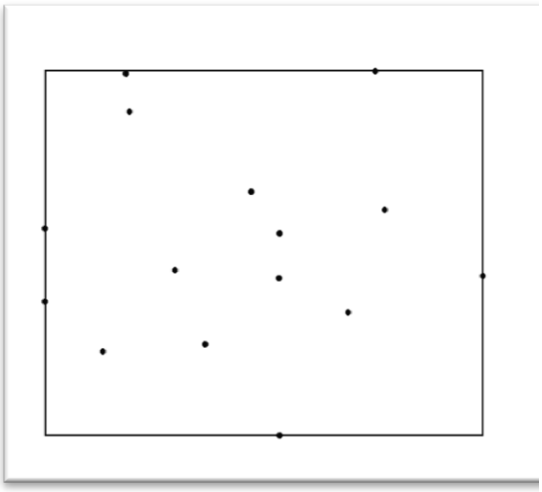
Fig.1 Graham's Scan



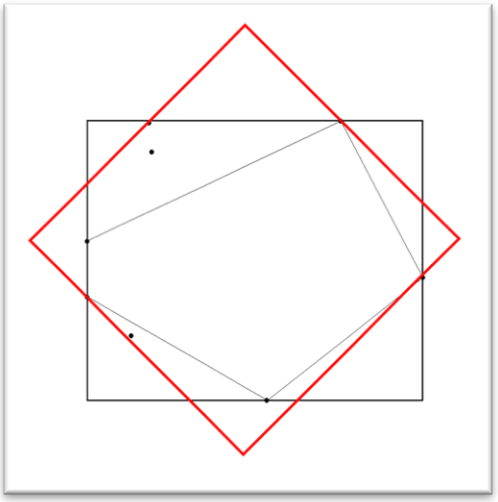
(a) 範例凸包



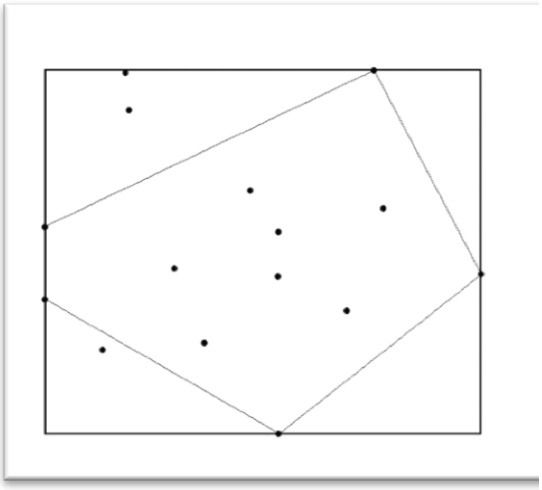
(d) 除去內部的點



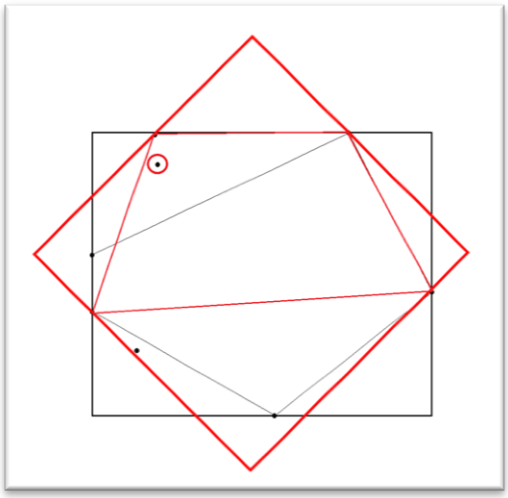
(b) 最小矩框及其凸包點



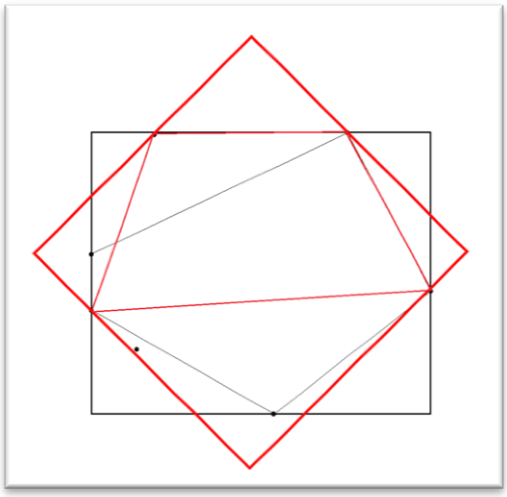
(e) 逆時針旋轉 45 度建立新的最小矩框



(c) 將矩框上的點連成多邊形

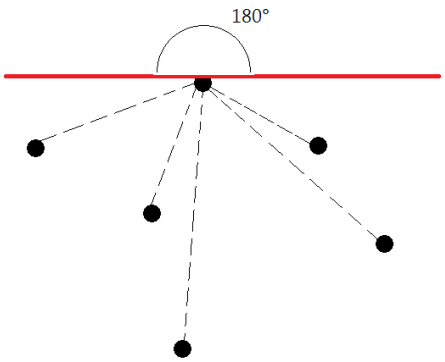


(f) 形成新的多邊形

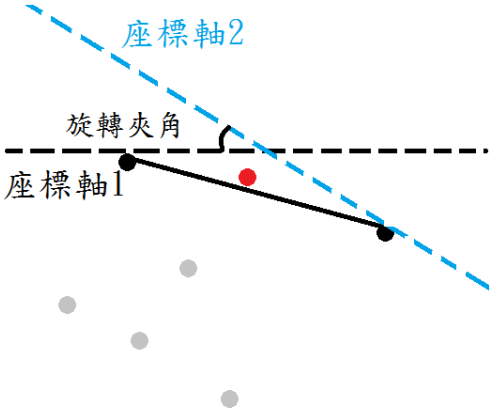


(g) 移除非凸包點後重複步驟直到旋轉次數或節點數符合要求

Fig.2 凸包演算法步驟圖解



(a) 最小矩框上的點



(b) 旋轉時被遺漏的節點

Fig.3 最小矩框法正確性之圖解

4.1 凸包之正確性

演算法中，Step 1 利用最小矩框上的點不可能與任意兩節點形成凹角(夾角>180°)的特性，尋找凸包點，如 Fig.3(a)。接著根據凸包定義，任意數個凸包點所包圍之多邊形內的點均不可能為凸包點，將非凸包點排除運算。然而旋轉座標軸時可能有部份凸包點無法被找到，如 Fig.3(b)中紅點，在演算法中使用 Graham 避免凸包形成時有節點被遺漏，基本上仍然不脫離 Graham 演算法的範疇，差別在時間複雜度，可由 Graham 演算法的正確性證明本演算法之正確性。

4.2 時間複雜度

我們的演算法中，預期時間複雜度為$O(n)$，其中n為二維平面上所有的節點，在Step 1 中建立最小包圍矩框需尋找所有的點，時間複雜度為$O(n)$。Step 2 以矩框上的點建立多邊形，尋找所有點以將多邊形內的非凸包點排除運算，時間複雜度為$O(n)$，因此多邊形邊數為常數。以及Step 3 將所有點n帶入旋轉矩陣取得新座標軸，最多旋轉常數項次，時間複雜度亦為$O(n)$。最後一步中，Graham's Scan 的時間複雜度為$O(n\log n)$，但平均狀況下最小矩框法將使節點數降為$\frac{n}{\log n}$，所以預期時間複雜度為$O(n)$。

總結 Steps 1-4，我們得出演算法的時間複雜度為$O(n)$。

5 結論

我們以最小矩框的概念提出的全新凸包解法。從點集合座標取得最小矩框後，藉由取出最小矩框上的凸包節點，移除凸包節點所形成多邊形內部的非凸包點。預估在平均狀況下，一到兩次運算即可排除大部分節點。使節點數由n降至$\frac{n}{\log n}$，接著將剩餘的節點代入任意時間複雜度為$O(n\log n)$之凸包演算法，便可在線性期望時間$O(n)$獲得凸包，其中n為節點總數。

未來研究將撰寫程式並完成執行結果。

參考文獻:

[1] S. G. AKL and G. T. Toussaint, "A Fast Convex Hull Algorithm," *Information Processing Letter*, Vol. 7, No.5, pp. 219-222,1978

[2] K. Q. Brown, "Voronoi diagrams from convex hulls", *Information Processing Letters*, vol. 9, no. 5, pp. 223-228, 1979.

[3] C. Barber, D. Dobkin and H. Huhdanpaa, "The quickhull algorithm for convex hulls", *ACM Transactions on Mathematical Software*, Vol. 22, No. 4, pp. 469-483, 1996.

[4] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letters*, Vol. 1, No. 4, pp. 132-133, 1972.

[5] H. F. Jiang, "A Fast Convex Hull Algorithm of Planar Point Set", *AMR*, Vol. 706-708, pp. 1852-1855, 2013.

[6] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation", *International Journal of Computer and Information Sciences*, vol. 9, no. 3, pp. 219-242, 1980.

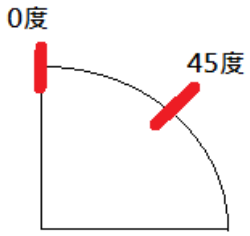
[7] I. Hong and A. T. Murray, "Efficient measurement of continuous space shortest distance around barriers", *International Journal of Geographical Information Science*, vol. 27, no. 12, pp. 2302-2318, 2013.

[8] P. L. Rosin, "Training cellular automata for image processing", *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 2076-2087, 2006.

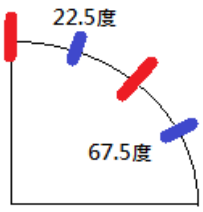
[9] C. C. Sun, G. E. Jan, S. W. Leu, K. C. Yang and Y. C. Chen, "Near-Shortest Path Planning on a Quadratic Surface With $O(n \log n)$ Time", *IEEE Sensors J.*, vol. 15, no. 11, pp. 6079-6080, 2015.

附錄: 演算法 Step 3 旋轉矩框的詳細說明

旋轉角度θ_i，因每次計算時的角度差距越大，越容易移除更多的點。所以我們選擇旋轉次數 k=1 時，旋轉角度θ_i = 45°；旋轉次數 k = 3 時，旋轉角度θ_i = 22.5°,45°,67.5°；以此類推，直到旋轉常數次或$S^{UCH} \leq n - (n/\log n)$，以使時間複雜度小於等於$O(n \log n)$。



(a) 旋轉次數k=1



旋轉次數k=3

Fig.4 旋轉角度θ_i