

基於四元樹之凸包新解

國立台南藝術大學

校長室

詹景裕

gejan@ntpu.edu.tw

國立臺灣海洋大學

電機工程學系

呂紹緯

b0119swleu@gmail.com

國立臺灣海洋大學

電機工程學系

蔡詠如

lmdmda@hotmail.com

美國底特律莫西大學

電機工程學系

雒超民

luoch@udmercy.edu

摘要

凸包是指在歐氏幾何平面上，能包覆一群散佈於平面上的最小凸集合，其表面積為最小。簡單來說，可想像成，在一個平面木板上釘著許多的釘子，而一條剛好能包著所有釘子的橡皮圈，所包覆的面積就是所有釘子被包覆的最小面積，而橡皮圈就是釘子的凸包。

而我們提出以四元樹資料結構，以遞迴的方式，先將平面上的點劃分，先形成許多子凸包，再將子凸包往上合併成一個完整的凸包來解決凸包問題。

關鍵字:凸包、四元樹。

1. 緒論

凸包是應用數學和計算幾何中的主要研究問題之一。而眾所周知的一些問題也與凸包問題有關，例如，Voronoi Diagram [1]，Delaunay triangulation [5]。凸包也應用在很多不同的領域如：圖像處理[7]，地理信息系統（GIS）[4]，無人地面車輛路徑規劃和二次表面巡航導彈任務規劃等軍事系統[8]等。

在1996年由Chan提出的Chan's algorithm方法[2]，以及1997年由Preparata和Hong提出以分而治之演算法(divide and conquer)[6]，都是以合併凸包的方式，解決凸包問題。而我們這次提出以四元樹資料結構為概念[9]，結合分而治之演算法(divide and conquer)，來完成凸包。

分而治之演算法解決凸包問題時，因為需要排序所有點，再將點劃分成兩等份，形成兩個凸包後找公切線，將兩凸包合成一個大凸包，所以其時間複雜度是 $O(n \log n)$ ， n 為所有點的數目，如果不排序則轉變成Chan's algorithm方法，將歐氏幾何平面上所有點隨機劃分成多個可能重疊的凸包，再合併成一個完整的凸包。Chan's algorithm的時間複雜度為 $O(n \log M)$ ， M 為凸包的頂點數目，但是將重疊的凸包合併成大凸包需要耗時的運算，因此演算法實際執行速度相當的慢。因此，我們使用四元樹的資料結構，省去分而治之演算法的排序時間，也使凸包不會重疊，只需合併不重疊的凸包，演算法更為簡單且實際執行速度快於Chan's algorithm。

2. 背景與基本概念

本章節我們將介紹四元樹和分而治之演算法(divide and conquer)。

分而治之演算法(divide and conquer)，用遞迴的方法將複雜的問題分割成兩個或更多的子問題，直到子問題不能再進一步的分割為止，再將子問題一一的解決，而原問題的解即子問題的解合併。

由Preparata及Hong提出的分而治之演算法，以合併的方式來解決凸包問題，將歐氏幾何平面上所有的點做排序，做等份的劃分，再求子凸包們的公切線，連接公切線，

合併子凸包，以完成凸包問題，這方法就是把問題劃分成多個子問題，一一解決子問題，即完成問題，也不會讓平面上的點有漏網之魚的可能。

四元樹資料結構是由 Finkel 與 Bentley 在 1974 年所提出的[3]，是指將平面分成四個子區塊，可以以條件遞迴分割下去，越分越細。而資料可以為線段、點、三角形等。

3. 演算法及其圖解

在本節中，我們介紹以分而之治演算法為基礎，結合四元樹資料結構的概念，從四元樹最底層開始，以每層子凸包合併的方式，解決凸包問題。

3.1 演算法

Function 1.1 建構四元樹資料結構()

Step 1 網格內的點超過四點，即往下分裂子樹

Step 2 當鄰近的手足(sibling)網格內都沒有節點，即往上合併為父節點(parent node)

END{建構四元樹資料結構()}

Function 1.2 水平合併凸包()

Step 1 找下公切線

Step 1.1 找到左凸包上 x 軸最大的頂點，與右凸包上 x 軸最小的頂點，分別為公切線段的左右端點。

Step 1.2 固定左凸包頂點，逆時針找右凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

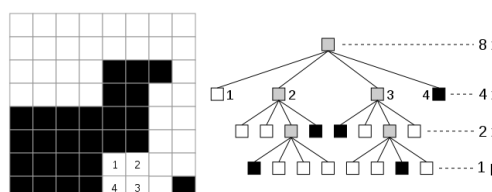


Fig. 1 四元樹資料結構 (摘錄於[11])

Step 1.3 固定右凸包頂點，順時針找左凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

Step 1.4 連接最後找到的兩頂點，形成下公切線。

Step 2 找上公切線

Step 2.1 找到左凸包上 x 軸大的頂點，與右凸包上 x 軸最小的頂點，分別為公切線段的左右端點。

Step 2.2 固定左凸包頂點，順時針找右凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

Step 2.3 固定右凸包頂點，逆時針找左凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

Step 2.4 連接找到的兩凸包頂點，形成上公切線。

END{水平合併凸包()}

Function 1.3 垂直合併凸包()

Step 1 找左公切線

Step 1.1 找到上凸包上 y 軸最小的頂點，與下凸包上 y 軸最大的頂點，分別為公切線段的左右端點。

Step 1.2 固定上凸包頂點，逆時針找下凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

Step 1.3 固定下凸包頂點，順時針找上凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

Step 1.4 連接找到的兩凸包頂點，形成左公切線。

Step 2 找右公切線

Step 2.1 找到上凸包上 y 軸最小的頂點，與下凸包上 y 軸最大的頂點，分別為公切線段的左右端點。

Step 2.2 固定上凸包頂點，順時針找下

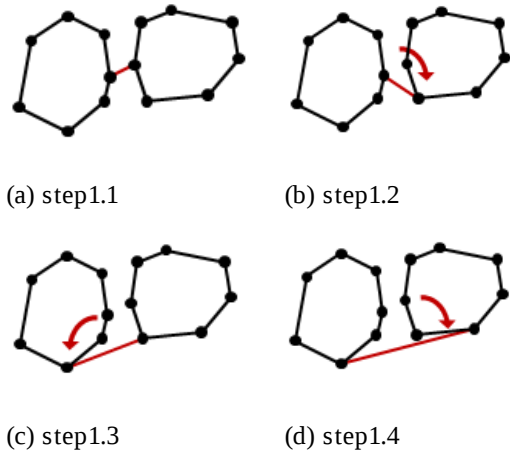


Fig. 2 水平合併凸包 Function2.2(找下公切線)
(摘錄於[10])

凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

Step 2.3 固定下凸包頂點，逆時針找上凸包上的頂點，直到無法相連(公切線會穿越凸包)為止。

END{垂直合併凸包()}

演算法： 以四元樹為基礎之凸包演算法()

Step 1 由歐氏幾何平面上 n 個點裡，找出 x 軸最大值以及最小值的點， y 軸最大值以及最小值的點，形成最小的矩框。

Step 2 將最小矩框剖分形成 $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ 的網格。

Step 3 Call Function 2.1 依網格內節點數量，建構四元樹資料結構。

Step 4 合併凸包。從四元樹最底層子節點開始，往上依序交錯水平 (Call Function 2.2) 及垂直 (Call Function 2.3) 合併凸包，直到剩下最頂層，並得到最後單一凸包。

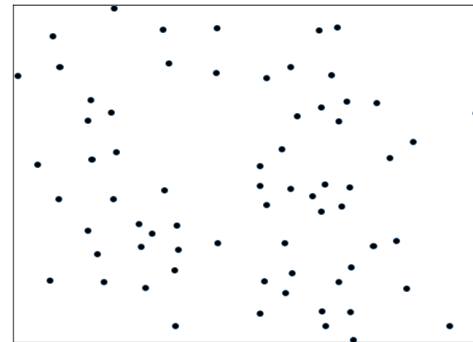
END{以四元樹為基礎之凸包演算法()}

3. 2. 演算法圖解

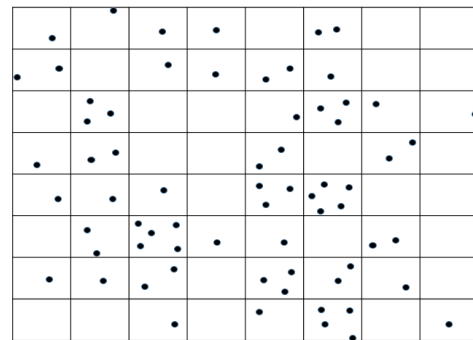
凸包演算法的圖解步驟為以下的 Fig.3，Fig. 3(a)- Fig.3(b)先形成最小矩框，用網格劃分所有節點，Fig. 3(c)以四元樹概念，當網格裡的節點超過四點，則往下分割成子樹，亦或者往上形成父節點，接著 Fig.3(d)-(k) 從四元樹資料結構最底層開始，以交錯水平及垂直的方式合併凸包，依序層層向上合併，最後在頂層完成最後凸包。

4. 效能分析

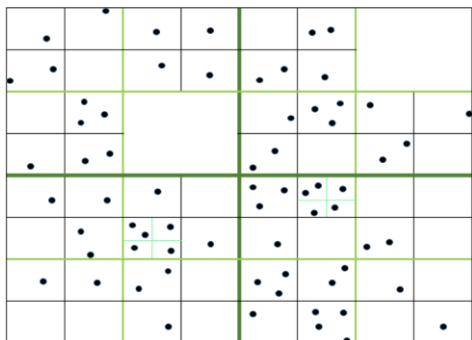
本章節為凸包正確性及時間複雜度之敘述。



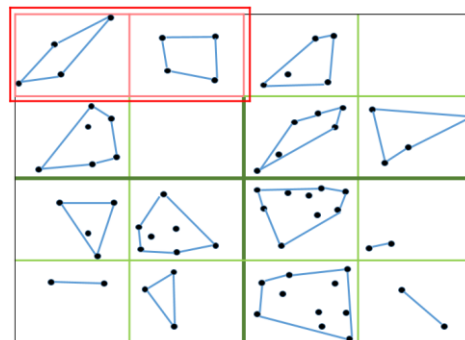
(a)形成最小矩框



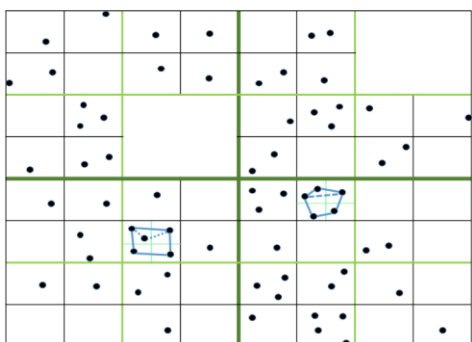
(b)劃分為 $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ 的網格



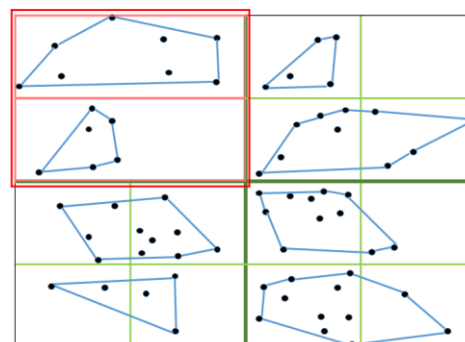
(c) 各網格的點往下分割或往上成樹



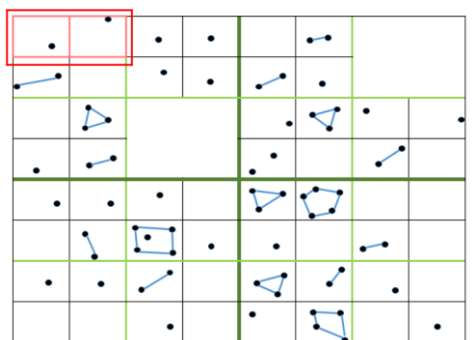
(g) 水平合併倒數第三層子節點



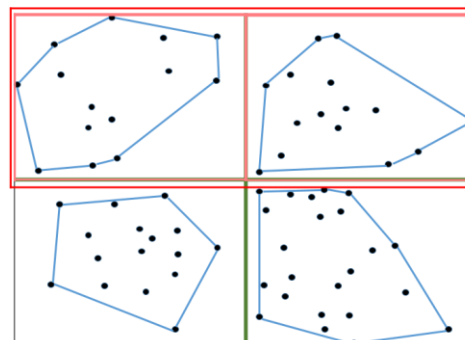
(d) 從四元樹最底層子節點合成小凸包



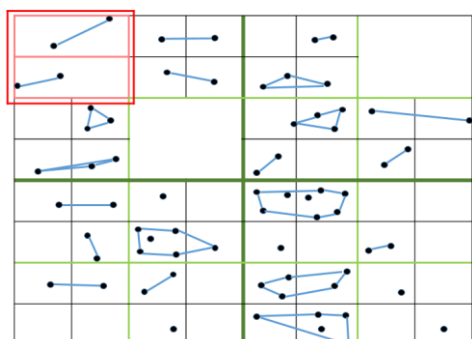
(h) 垂直合併倒數第三層子節點



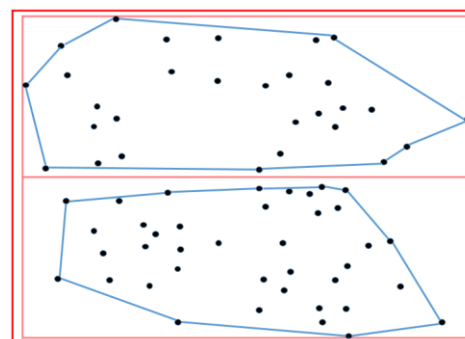
(e) 水平合併倒數第二層子節點



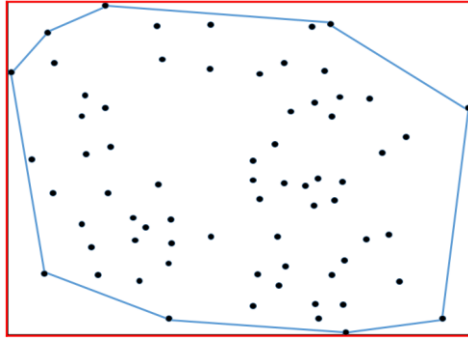
(i) 水平合併倒數第四層子節點



(f) 垂直合併倒數第二層子節點



(j) 垂直合併倒數第四層子節點



(k)完成凸包

Fig.3 演算法細部圖解

4.1 凸包之正確性

我們的方法，是將 divide and conquer 演算法，原本提出將平面上所有的點 n 分割的方式，以四元樹資料結構來分割，這樣能省去原本需要排序點的時間，而後面合併的部分已原本的 divide and conquer 演算法是一致的，所以不會有任何點遺漏，以確保正確性。

4.2 時間複雜度

在我們的演算法中，時間複雜度為 $O(n \log n)$ ，其中 n 為二維平面上所有的節點，在 Step 1 中尋找最小矩框，尋找所有的點，時間複雜度為 $O(n)$ 。Step 2 將矩框均分為 $\sqrt{n} \times \sqrt{n}$ 的網格，時間複雜度為 $O(n)$ ，Step 3 中建構四元樹需要 $O(n)$ 時間及 Step 4，每次往上合併凸包需要 $O(n)$ 時間，共需要 $O(\log n)$ 次。

總結 Steps 1-4，我們得出演算法的時間複雜度為 $O(n \log n)$ 。

5. 結論

我們提出以四元樹的資料結構，先將歐氏幾何平面上的點，以四元樹的資料結構做劃分，再求子凸包們的公切線，連接公切線，合併子凸包，以完成凸包問題。

比起先前的分而治之演算法，我們省去了一開始需要排序點的時間，降低時間複雜度，也不會使凸包重疊，只需合併不重疊的凸包，演算法更為簡單且實際執行速度快於 Chan's algorithm，且也不會有漏網之魚的可能。

參考文獻：

- [1] K. Q. Brown, "Voronoi diagrams from convex hulls", *Information Processing Letters*, vol. 9, no. 5, pp. 223-228, 1979
- [2] T. M. Chan. "Optimal output-sensitive convex hull algorithms in two and three dimensions," *Discrete and Computational Geometry*, Vol. 16, pp.361-368, 1996.
- [3] R. Finkel and J. L. Bentley, "Quad Trees:A Data Structure for Retrieval on Composite Keys," *Journal Acta Informatica.*, Vol.4, Issue. 1, pp.1-9, March 1974.
- [4] 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.
- [5] 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
- [6] F. P. Preparata, S.J. Hong. "Convex Hulls of Finite Sets of Points in Two and Three Dimensions, " *Commun. ACM*, vol. 20, no. 2, pp. 87-93, 1977.
- [7] P. L. Rosin, "Training cellular automata for image processing", *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 2076-2087, 2006.
- [8] 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.
- [9] Hanan Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, U.S.A., 2006.

- [10] 演算法筆記, Convex Hull: Divide and Conquer, Dec. 26, 2016, from <http://www.csie.ntnu.edu.tw/~u91029/ConvexHull.html>
- [11] 維基百科, 四叉樹, Dec. 30, 2016, from <https://zh.wikipedia.org/wiki/%E5%9B%9B%E5%8F%89%E6%A0%91>