應用字串圖形辨識法於包含問題之研究

[1]朱倬梁
明道大學資訊傳播學系
[1]副教授
chujl@mdu.edu.tw

[2]李天明
明道大學資訊傳播學系
[2]助理教授
lee_tenmin@mdu.edu.tw

[3]洪碧君 徐千翔
明道大學資訊傳播學系
[3]研究生
s1199110@yahoo.com.tw
pikachu7604300@yahoo.com.tw

摘要

判斷平面上的點是否在多邊形內是一個相當基本的問題，但係諸多應用領域之基礎技術。我們特別稱此問題為「包含問題」，即「inclusion problem」。

「包含問題」的應用主要用於電腦繪圖領域。近年來全球定位系統、無線感測網路盛行，本技術和這些系統結合後，正朝向自動導引及虛擬圍牆等影像辨識領域發展。

關於「包含問題」的研究，大體上係將多邊形可分成兩類：凸多邊形(convex polygon)及凹多邊形(concave polygon)，分別發展應用技術。判斷平面上的點是否在凸多邊形內，對電腦而言是項輕而易舉的事。只須檢查該點是否位於多邊形各邊線之同側即可。然而要判斷該點是否在凹多邊形內就顯得相當棘手。

本文依據Fong and Chu 於1990年發表的用以辨識多邊形與線段間四種關係(線型)：外線、內線、割線、邊線的方法，人稱字串圖形辨識法(a string pattern recognition approach)。設計演算法將任意形狀的簡單多邊形予以三角形化，使多邊形變成三角形的組合。成功地將「辨識測試點是否在多邊形內」的複雜問題，轉變成「辨識測試點是否在三角形內」的簡單問題。經實例驗証，本法有效且簡單易行。

關鍵字：包含問題、字串圖形辨識法、凸凹多邊形、多邊形三角形化

1.前言

探討平面上的點是否在多邊形內的方法，可分成兩類：

- ✧ **逐點判斷演算法**：參考文獻[4]提出從測試點出發，分別作垂直朝上、朝下兩條射線。若朝上射線與多邊形邊線的交點數為奇數，且朝下射線與多邊形邊線的交點數亦為奇數，則該點在多邊形內。另有文獻[2]以類神經網路演算法來執行這個觀點。逐點判斷演算法的缺點是當應用於「繪圖著色」過程時，需對畫面上每一像素作一次「在不在多邊形內」之判斷，以便著色。畫面上的像素往往有上百萬個，計算會相當耗時。
- ✧ **掃描線演算法**：本法係先將多邊形分解成凸多邊形的集合，然後判斷掃描線掃過多邊形後，有那些線段落在凸多邊形內。落在凸多邊形內的線段，其所含的像素點，統統包含於多邊形內。所以本法有不需逐點判斷的優點。詳情請參閱[1]。不過應用[1]的方法，需要特別注意「奇點」的問題。本文援用字串圖形辨識法[3]將多邊形分解成三角形的集合。過程中不需考慮「奇點」的問題，這是本文的優點。

1.1 基本概念介紹

頂點　多邊形邊線的端點稱為多邊形的頂點(vertex)。頂點與頂點間用直線相連會連成

簡單多邊形。

簡單多邊形：多邊形的邊線僅和相鄰兩邊線相交，且相交在頂點。本文僅討論簡單多邊形。

凸多邊形：多邊形的所有內角均小於180°。

凹多邊形：多邊形的內角有大於180°者。

頂點與邊線之序向均為逆時鐘方向：如圖1所示，多邊形的頂點排列順序為：p_1, p_2, p_3；邊線的方向分別為：$\overrightarrow{p_1p_2}, \overrightarrow{p_2p_3}, \overrightarrow{p_3p_1}$

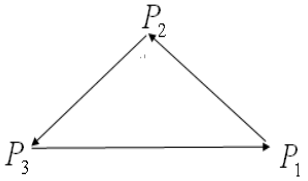


圖 1. 頂點及邊線皆逆時鐘方向排序

1.2 測試點 p_t 和有向直線間的關係

判斷測試點在有向直線的左邊或是右邊的方法：如圖 2，將 p_t 的 x、y 軸座標代入公式(1)，本公式之推導，請參閱[2]。

$$p_{tx}w_x + p_{ty}w_y + b = n \qquad (1)$$

$$w_x = p_{2y} - p_{1y} \qquad (1\text{-}1)$$

$$w_y = p_{1x} - p_{2x} \qquad (1\text{-}2)$$

$$b = p_{2x}p_{1y} - p_{1x}p_{2y} \qquad (1\text{-}3)$$

若 $n = 0$，則 p_t 點在直線上

若 $n > 0$，則 p_t 點在直線的右邊

若 $n < 0$，則 p_t 點在直線的左邊

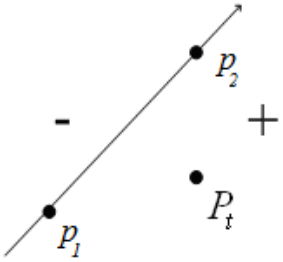


圖 2. 測試點與有向直線的關係

1.3 辨識測試點在三角形內部

1.3.1「點」在三角形內部

圖3中，測試點 p_{t1} 在三角形邊線 $\overrightarrow{p_1p_2}$，$\overrightarrow{p_2p_3}$，$\overrightarrow{p_3p_1}$ 之同側(左側)，代表 p_{t1} 在三角形內部。

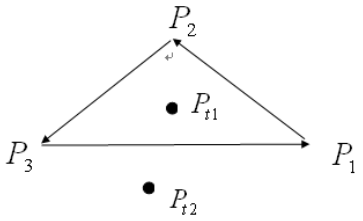


圖3 「點」在三角形內或外部的示意圖

1.3.2「點」在三角形外部

圖3中，測試點 p_{t2} 在邊線 $\overrightarrow{p_1p_2}$，$\overrightarrow{p_2p_3}$，之左側，而在 $\overrightarrow{p_3p_1}$ 之右側。只要有一邊不同側，即表 p_{t2} 在三角形外部。

2. 主要內容

2.1 Fong & Chu 字串圖形辨識法簡介[3]

本法介紹如何用字串圖形辨識法辨識平面上多邊形和線段間的四種關係：外線、內線、割線、邊線。

2.1.1 線段延長成直線並分成三段不同區域

如圖 4，將線段 $\overrightarrow{p_1p_2}$ 延長成一直線，可分成三段不同區域：

✧ a 段：從 p_2 點延長出去的半線(含 p_2 點)

✧ b 段：從 p_1 點延長出去的半線(含 p_1 點)

✧ 0 段：即線段 $\overrightarrow{p_1p_2}$ 本身(不含 p_1 及 p_2 點)

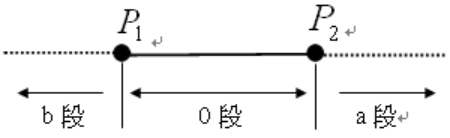


圖 4 將直線分成三段：a 段、b 段及 0 段

2.1.2 直線與多邊形相交產生字串

如圖 5，線段延成直線 $\overleftrightarrow{p_1p_2}$ 後，會和多邊形的邊線相交。交點和多邊形的頂點分別按照下列規則作標記：

✧ 交點標記為〝a〞：交點位於 a 段

✧ 交點標記為〝b〞：交點位於 b 段

✧ 交點標記為〝0〞：交點位於 0 段。

✧ 頂點標記為〝+〞：若多邊形的頂點位於直線 $\overleftrightarrow{p_1p_2}$ 右側，則將該頂點標記為〝+〞。

✧ 頂點標記為〝－〞：若頂點位於直線 $\overleftrightarrow{p_1p_2}$ 左側，則將該頂點標記為〝－〞。

以任一交點為起始點，沿多邊形邊線朝逆時鐘方向走一圈，且按序將標記寫下，即得一字串。從圖 5，可得循環字串 a--b++++

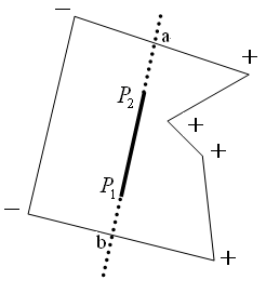


圖 5 直線與多邊形相交產生字串

2.1.3 正分枝、負分枝與交換枝

正分枝(positive branch)： a，b 間夾有〝+〞和〝0〞之字串，稱為正分枝。其中〝+〞的個數至少 1 個以上。

負分枝(negative branch)：a，b 間夾有〝-〞和〝0〞之字串，稱為負分枝。其中〝-〞的個數至少 1 個以上。

交換枝(switch branch)： 字串：+0－或－0+，稱為交換枝。

2.1.4 辨識四種線型的方法

平面上多邊形和線段之間存在下列四種線型(或稱關係)，其辨識規則如下：

✧ **外線(External Line)：**

正分枝的個數=偶數

負分枝的個數=偶數

圖 6 產生字串：a---b+b----a+，其中負分枝的個數為 2，即：a---b 和 b----a。正分枝的個數為 0，分枝個數均為偶數，故 $\overline{p_1p_2}$ 為外線。

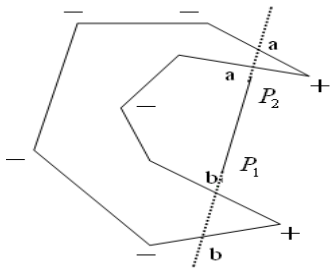


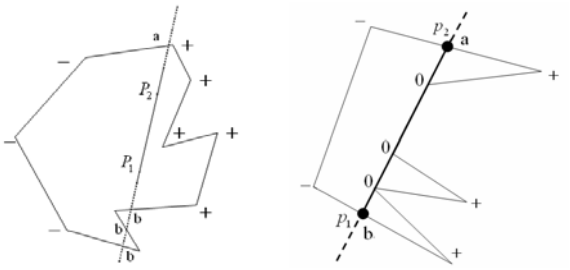
圖 6 $\overline{p_1p_2}$ 為外線之辨識圖

✧ **內線(Interior Line)：**

正分枝的個數=奇數

負分枝的個數=奇數

圖 7(a)產生字串：a---b+b-b+++++，其中負分枝的個數為 1(a---b)，正分枝的個數亦為 1(b+++++a)，故 $\overline{p_1p_2}$ 為內線。



(a) (b)

圖 7 $\overline{p_1p_2}$ 為內線之辨識圖

圖 7(b)產生字串：a--b+0+00+，其中負分枝的個數為 1，即：a---b。正分枝的個數亦為 1，即：b+0+00+a。分枝個數均為奇數，故 $\overline{p_1p_2}$ 為內線。

✧ **邊線(Edge Line)：**

正分枝的個數=偶數(奇數)

負分枝的個數=奇數(偶數)

圖 8 產生字串：a--b+b+ba+，其中負分枝的個數為 1，即：a--b。正分枝的個數為 0。分枝個數一奇一偶，故 $\overline{p_1p_2}$ 為內線。

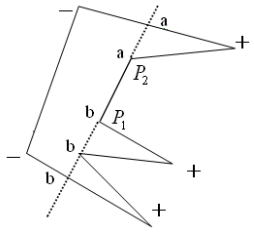


圖 8 $\overline{p_1p_2}$ 為邊線之辨識圖

✧ **割線(Intersecting Line)：**

字串中含交換枝者。

圖 9 產生字串 a-0+b-b++。字串含交換枝：－0+，故 $\overline{p_1p_2}$ 為割線。

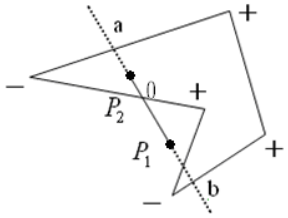


圖 9 $\overline{p_1p_2}$ 為割線之辨識圖

2.2 用字串法分解多邊形成三角形的聯集

2.2.1 多邊形三角形化的程式步驟

步驟 1：輸入尚未三角形化的多邊形，**列出頂點序列**：$a_0a_1a_2.....a_n$。

說明：a_0 到 a_n 表多邊形的頂點，係沿著邊線依逆時鐘方向排序。此序列是循環序列，即 a_0 接 a_1、a_1 接 a_2、…、a_n 接 a_0。

步驟 2：選取開始點 P。

說明：程式一開始，P=a_0。

步驟 3：檢查三角形 $\Delta pp_{next}p_{last}$ 是否成形。

說明：1. 程式起始，$p=a_0$，$p_{next}=a_1$，$p_{last}=a_n$。$\therefore \Delta pp_{next}p_{last}=\Delta a_0a_1a_n$

2. 三角形 $\Delta pp_{next}p_{last}$ 成形，表示 $\Delta pp_{next}p_{last}$ 包含於多邊形內。

3. 若 $\overline{p_{next}p_{last}}$ 是內線或邊線，則三角形 $\Delta pp_{next}p_{last}$ 成形。

若三角形不成形，回到**步驟 2**，更新 $p=p_{next}$。

若三角形成形，則進入**步驟 4。**

步驟 4：將三角形 $\Delta pp_{next}p_{last}$ 儲存入庫。

步驟 5：從頂點序列中刪除 p，形成新序列。

步驟 6：檢查新序列的個數是否大於 2。

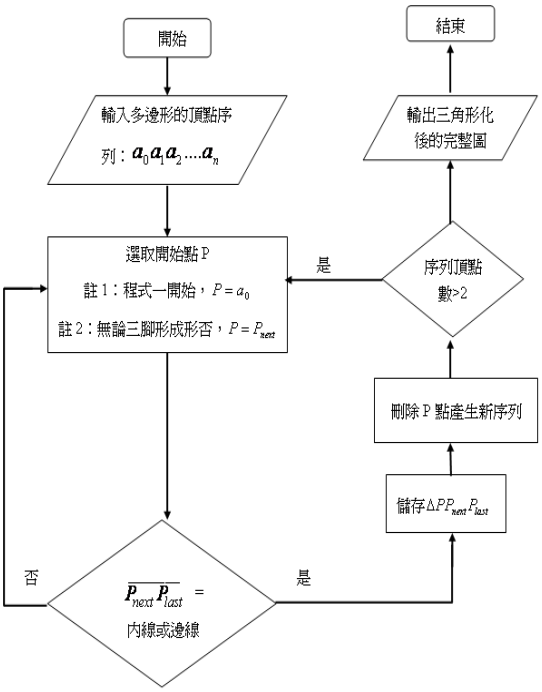
「是」，回到**步驟 2**，**更新** $p=p_{next}$。

「否」，進入**步驟 7**，

步驟 7：輸出完整三角形化後的多邊形。

步驟 8：結束。

2.2.2 多邊形三角形化的程式流程圖



2.2.3 辨識測試點是否在多邊形內

多邊形被分解成三角形的組合，且一一納入儲存庫後。辨識測試點 p_t 是否在多邊形內的問題，就轉變成辨識 p_t 點是否在儲存庫裡某一三角形內部的問題。只要將儲存庫裡之三角形逐一呼叫出來，按照 1.3 節介紹的方法，測試 p_t 點是否在三條邊線之同側，即可辨識 p_t 點是否在內部。若測出 p_t 點在某一三角形的內部，則

p_t 點在該多邊形的內部，否則在外部。

3. 實例驗證

舉例驗證本辨識法是否有效，以圖10(a)之多邊形為例，按照前節所述步驟逐步執行，其中 p_t 點之坐標為 $p_t(3,1)$。

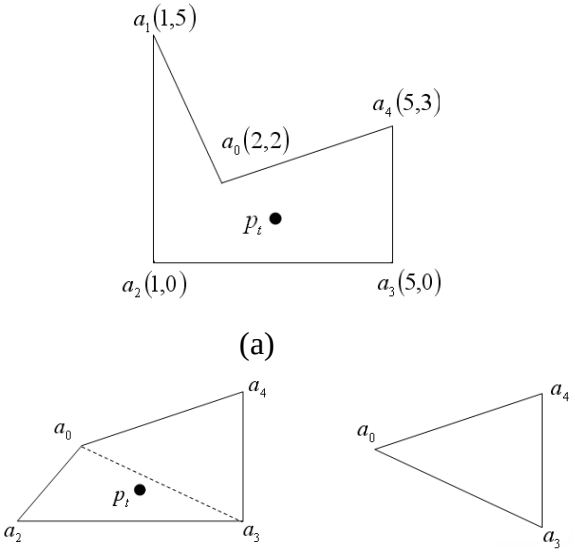


圖 10 分解多邊形 $a_0a_1a_2a_3a_4$ 成三角形

3.1 逐步進行分解多邊形步驟

步驟 1： 多邊形之**頂點序列**：$a_0a_1a_2a_3a_4$。

步驟 2：開始點 P=a_0。

步驟 3：檢查 $\Delta a_0a_1a_4$ **是否成形。**

因 $\overline{a_4a_1}$ 是外線，三角形不成形，回**步驟 2**

步驟 2，更新 $p=a_1$。

步驟 3：檢查 $\Delta a_1a_2a_0$ **是否成形。**

因 $\overline{a_2a_0}$ 是內線，三角形成形，進入**步驟 4**

步驟 4：將三角形 $\Delta a_1a_2a_0$ **儲存入庫。**

步驟 5：從頂點序列中刪除 a_1**，形成新序列：**
$a_0a_2a_3a_4$。此時多邊形如圖10(b)。

步驟 6：檢查新序列頂點個數是否大於 2。

「是」，回到**步驟 2**

步驟 2，更新 $p=a_2$。

步驟 3：檢查 $\Delta a_2a_3a_0$ **是否成形。**

因 $\overline{a_3a_0}$ 是內線，三角形成形，進入**步驟 4**

步驟 4：將三角形 $\Delta a_2a_3a_0$ **儲存入庫。**

步驟 5：從頂點序列中刪除 a_2**，得到新序列：**
$a_0a_3a_4$。此時多邊形如圖10(c)。

步驟 6：檢查新序列的個數是否大於 2。

「是」，回到**步驟 2**

步驟 2，更新 $p=a_3$。

步驟 3：檢查 $\Delta a_3a_4a_0$ **是否成形。**

因 $\overline{a_4a_0}$ 是邊線，三角形成形，進入**步驟 4**

步驟 4：將三角形 $\Delta a_3a_4a_0$ **儲存入庫。**

步驟 5：從頂點序列中刪除 a_3**，得到新序列：**
a_0a_4。

步驟 6：檢查新序列的個數是否大於 2。

「否」，進入步驟 7

步驟 7:輸出完整三角形化後的多邊形如圖 11。

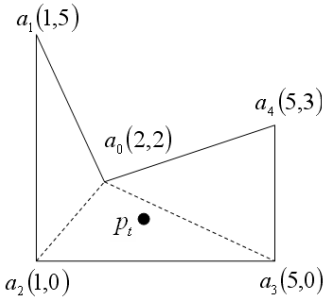


圖 11 多邊形被分解成三個三角形

步驟 8：程式結束。

3.2 辨識測試點在三角形內

執行完3.1節的分解步驟後，圖10(a)所示之多邊形被順利分解成三個三角形：$\Delta a_1a_2a_0$，$\Delta a_2a_3a_0$，$\Delta a_3a_4a_0$。

按照1.2節的公式，計算「點」在直線的左側或右側，且用 1.3節介紹的方法測試「點」是否位於三角形內。得知 p_t 點確實在 $\Delta a_2a_3a_0$ 內，因此 p_t 點在多邊形 $a_0a_1a_2a_3a_4$ 內。

4. 結論

本文應用字串圖形辨識法〔3〕,試圖對「包含問題」作出供獻。藉著線段截割多邊形所產生的字串來確認線段的四種線型(外線、內線、邊線、割線)。利用辨識線型的能力，本文設計完成分解多邊形成三角形的演算步驟。於是將辨識點在多邊形內的困難問題，轉變成辨識點在三角形(凸多邊形)內的簡單問題。從實例驗證，可看出本法確實簡單易行。未來發展方向有二：

1. 發展最佳多邊形分解法：本論文援用字串圖形辨識法[3]，將多邊形分解成三角形的集合。在分解過程中，發現該法極具協助開發最佳多邊形分解法(凸多邊形個數最少)的潛能。可達減少計算量，提升效能之目的。
2. 開發簡易的「比較」法，取代「計算」法：本法運作最耗時的地方是當尋求線段與多邊形之交點時，需作大量的運算，以便對交點作標記來產生字串。為提高系統的效率，宜發展簡易的「比較」法來協助判定會不會有這個交點，交點位於那一區段，而不是真的去「計算這個交點的精確位置」。

參考文獻

[1] 唐榮錫等編著“**計算機圖學**”網奕資訊，2005，第7-3到7-14頁。

[2] 朱倬梁、蔡永勝、柯志忠“**辨識點在多邊形內的類神經網路演算法**”TAAI 2009 第十四屆人工智慧與應用研討會 p117.

[3] David Yu-Shan Fong and Jwo-Liang Chu “**A String Pattern Recognition Approach To Polygon Clipping**” Pattern Recognition Vol.23. No. 8. pp879-892 ， 1990,

[4] Shamos M.L, “**computational geometry**”, unpblished Ph.d. thesis, Yale University, May 1978