

# 國立交通大學

## 資訊科學與工程研究所

### 碩士論文

基於擴散曲線之點陣圖自動向量化

Automatically Vectorizing Raster Image based on  
Diffusion Curves

研究生：池品軒

指導教授：林文杰 教授

共同指導教授：莊榮宏 教授

中華民國 102 年 9 月

基於擴散曲線之點陣圖自動向量化  
Automatically Vectorizing Raster Image based on Diffusion Curves

研究生：池品軒

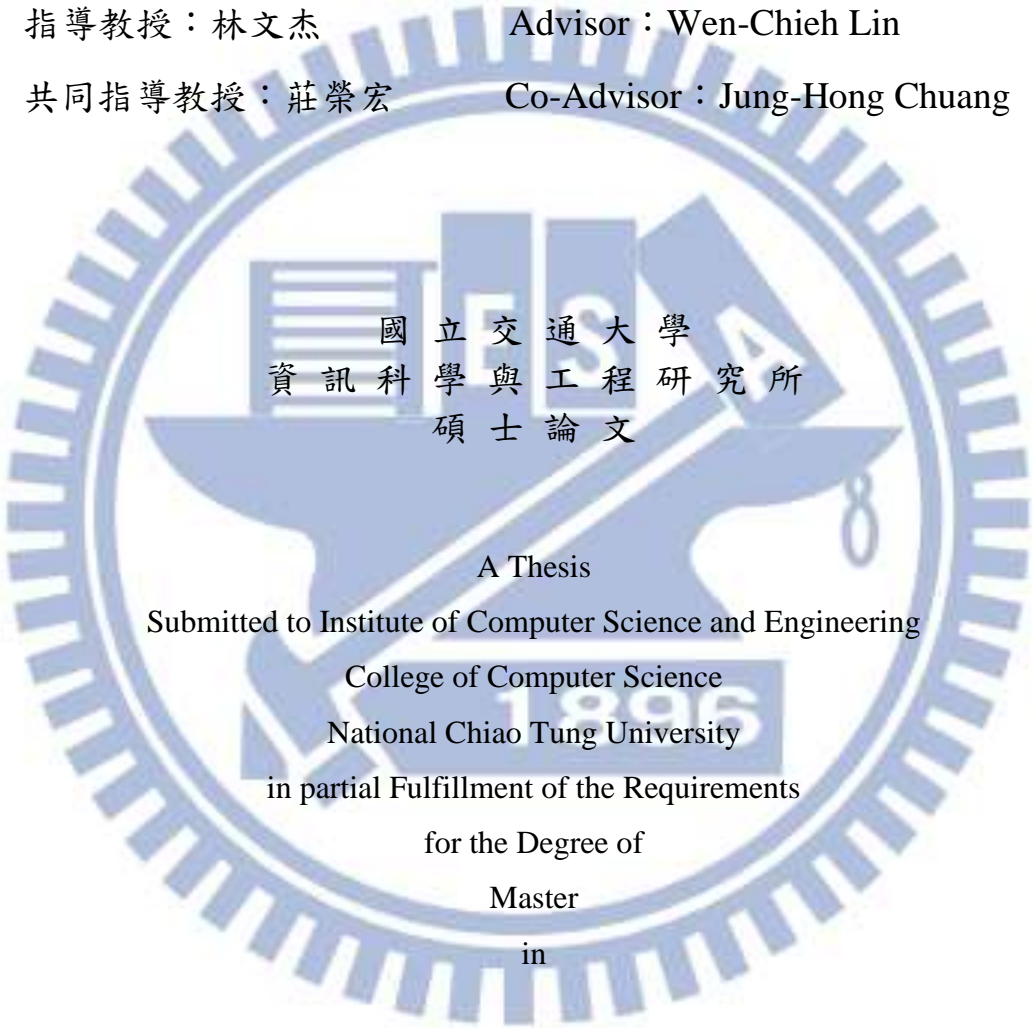
Student : Pin-Shen Chih

指導教授：林文杰

Advisor : Wen-Chieh Lin

共同指導教授：莊榮宏

Co-Advisor : Jung-Hong Chuang



國立交通大學  
資訊科學與工程研究所  
碩士論文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2013

Hsinchu, Taiwan, Republic of China

中華民國 102 年 9 月

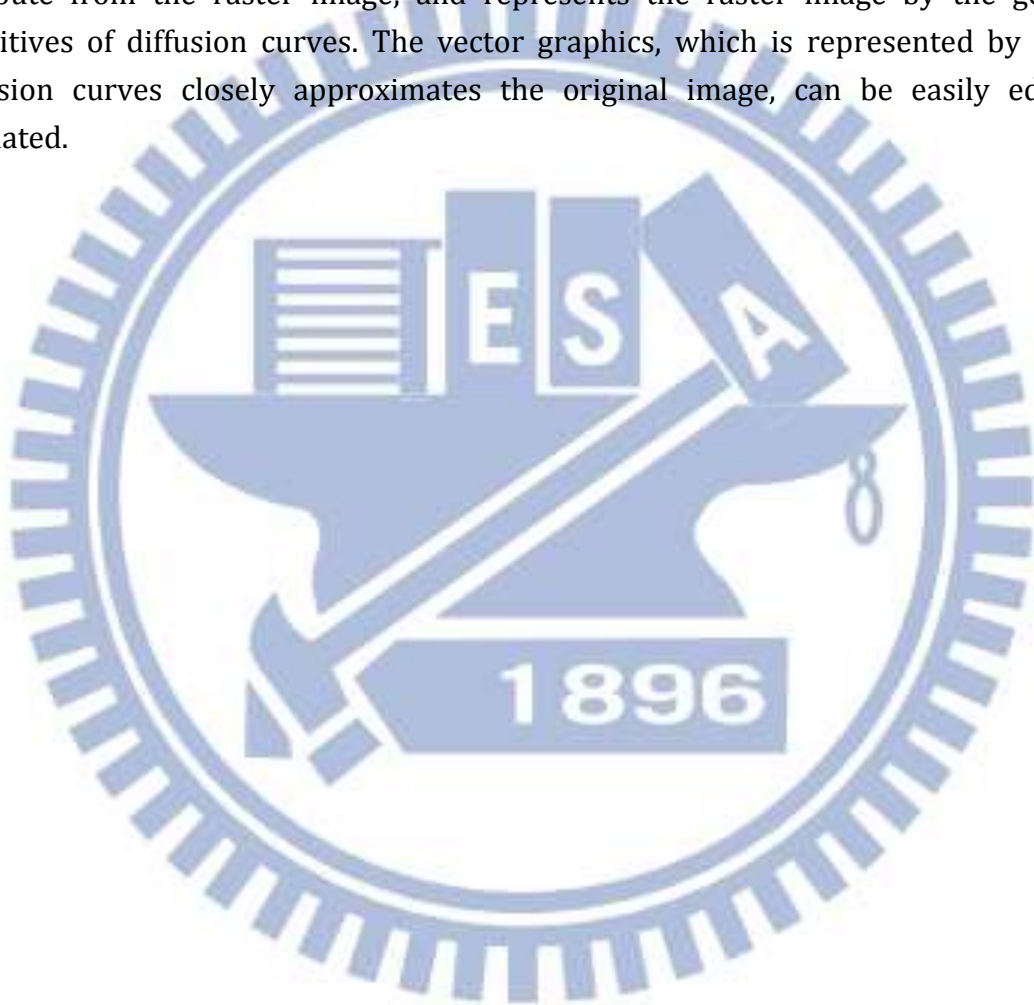
## 中文摘要

擴散曲線是一種新的向量基元，這種基元被繪製時會將圖片上的區域分開，而且可以在曲線兩側定義顏色。這些顏色會從曲線上平滑地向兩側擴散，遂至整張圖片。本論文基於擴散曲線的特性，提出了一個點陣圖向量化的自動化方法。我們從點陣圖中萃取輪廓、顏色及模糊資訊，並轉換成擴散曲線所需要的幾何基元。我們的方法可以將點陣圖轉換成一組擴散曲線，這組擴散曲線所表示的向量圖會很接近原始的點陣圖內容，而且可以很容易地編輯或製作成動畫。



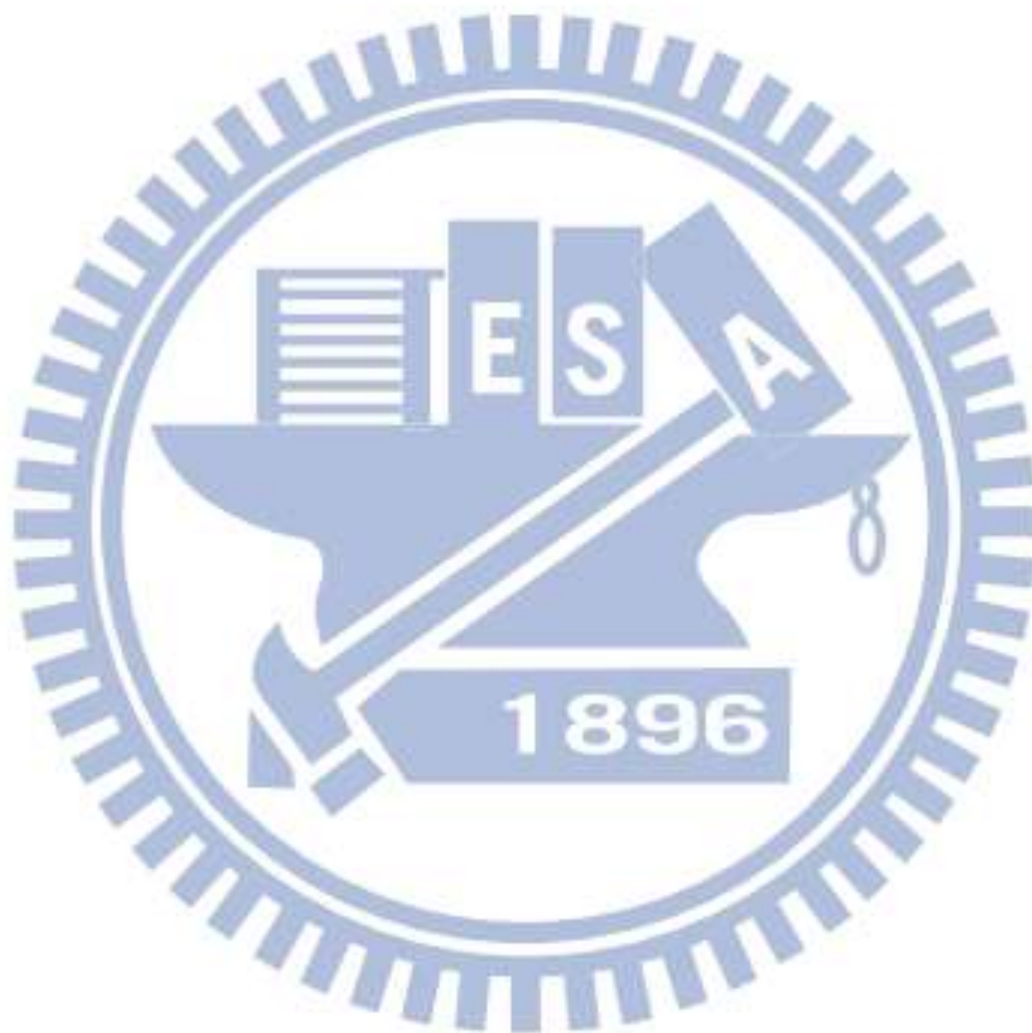
## ABSTRACT

Diffusion curve is a novel vector-based primitive. It partitions the space through which it is drawn and define colors on each side. These colors smoothly diffuse outwards from each side until they cover the entire image. In this thesis, we propose a method to automatically vectorize a raster image. Our method extracts the contour, color and blur attribute from the raster image, and represents the raster image by the geometry primitives of diffusion curves. The vector graphics, which is represented by a set of diffusion curves closely approximates the original image, can be easily edited or animated.



## 誌謝

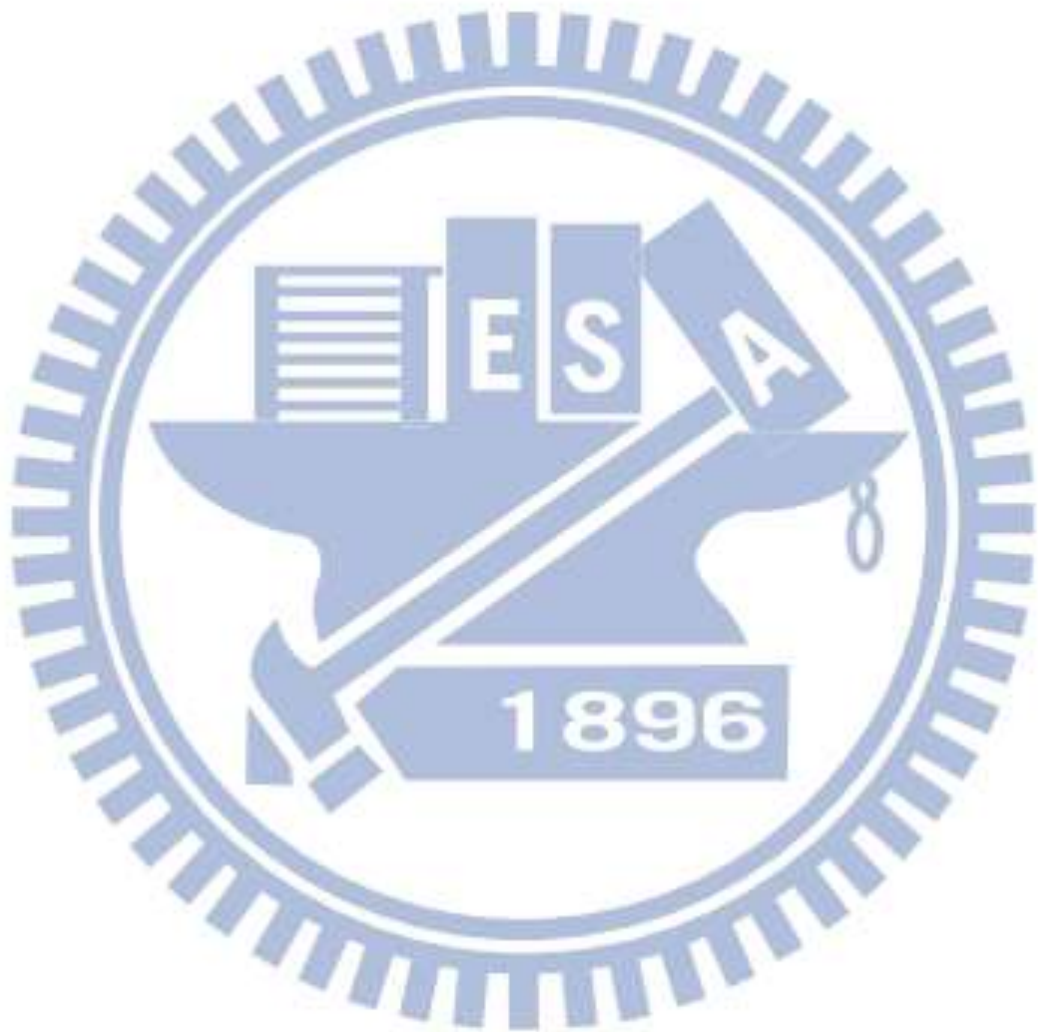
我要感謝我的指導教授林文杰博士的耐心指導。並且感謝 GPL 和 CAIG 實驗室的同儕們的幫助及指教，另外也感謝許多朋友的協助及鼓勵。最後，我要感謝家人不斷地給我支持，讓我可以全力地完成碩士學位。



# 目錄

中文摘要.....	1
ABSTRACT.....	2
誌謝.....	3
目錄.....	4
中英文對照表.....	6
第一章 緒論.....	8
第二章 文獻回顧.....	10
2.1 向量圖形工具.....	10
2.2 擴散曲線應用.....	10
第三章 方法.....	17
3.1 概觀.....	17
3.2 擴散曲線.....	18
3.2.1 資料結構.....	18
3.2.2 顯像方法.....	18
3.2.2.1 顏色來源.....	20
3.2.2.2 顏色擴散.....	21
3.2.2.3 模糊資訊擴散.....	21
3.2.2.4 綜合結果.....	22
3.3 輪廓偵測.....	23
3.3.1 方向梯度量值.....	23
3.3.2 亮度、顏色及紋理梯度.....	24
3.3.3 多重解析度的強度組合.....	26
3.4 輪廓追蹤.....	27
3.4.1 摩爾鄰居演算法.....	27
3.4.2 轉換成貝茲樣條曲線.....	28
3.5 萃取顏色及模糊資訊.....	29
第四章 結果.....	31
第五章 貢獻與未來展望.....	38

5.1	貢獻 .....	38
5.2	未來展望 .....	38
<b>參考資料 .....</b>		<b>40</b>

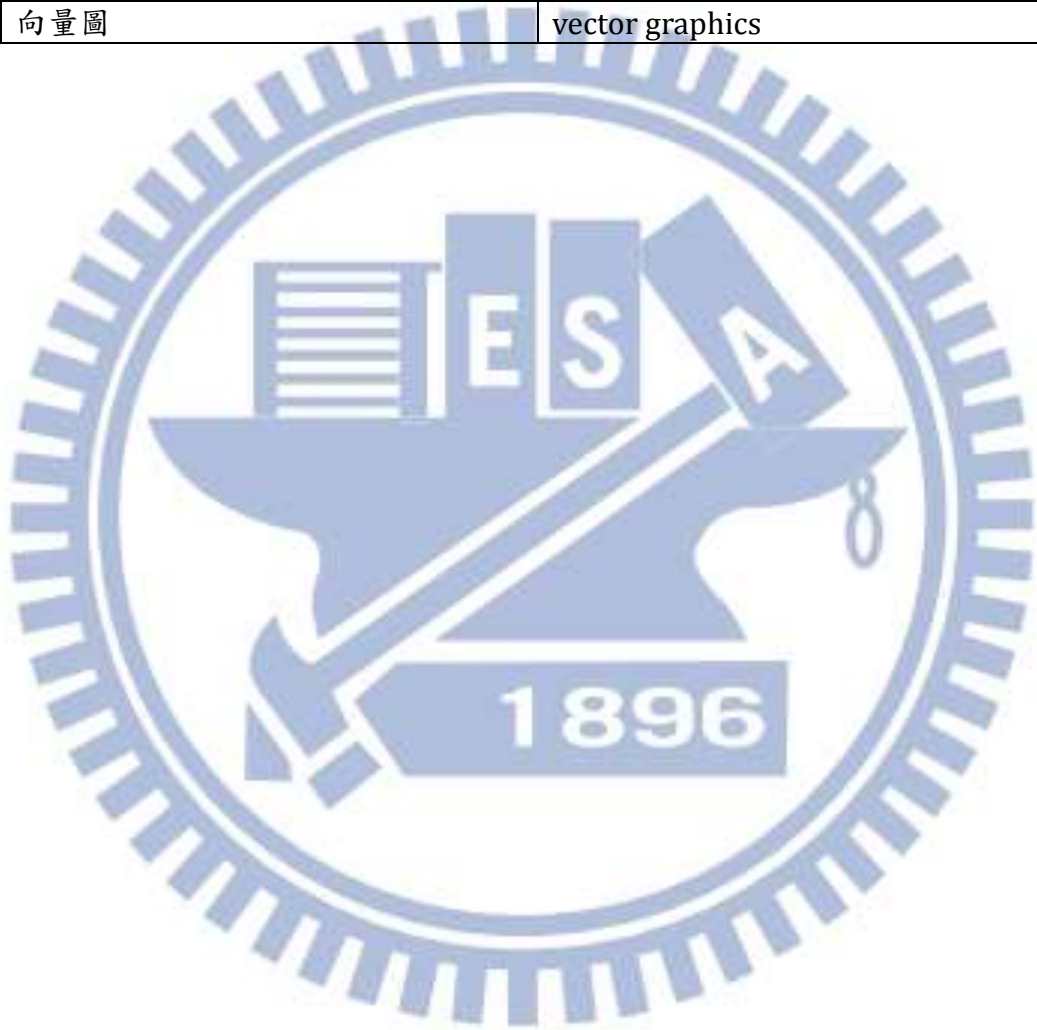


## 中英文對照表

中文	英文
貝茲曲線	Bezier curve
貝茲樣條曲線	Bezier spline
模糊	blur
模糊控制點	blur control point
亮度	brightness
通道	channel
顏色控制點	color control point
輪廓偵測	contour detection
輪廓追蹤	contour tracing
控制點	control point
卷積	convolution
三次的	cubic
擴散曲線	diffusion curve
擴散曲面	diffusion surface
散度	divergence
萃取	extract
伽柏雜訊	Gabor noise
高斯導數濾波器	Gaussian derivative filter
圖形處理器	GPU
梯度	gradient
梯度域	gradient domain
漸層網格	Gradient mesh
關鍵影格	key frame
拉普拉斯解答器	Laplacian Solver
線性內插	linear interpolation
摩爾鄰居演算法	Moore neighborhood algorithm
非真實感繪製	Non-photorealistic rendering
方向梯度量值	Oriented gradient magnitude
參數化	parameterization
基元	primitive
像素	pixel
像素鏈	pixel chain
泊松方程式	Poisson equation
折線	Polyline
二次的	quadratic
點陣圖	raster graphics
柵格化	rasterization



光線追蹤	ray tracing
解析度相依	resolution-dependence
解析度獨立	resolution-independence
分割	segmentation
光譜分群	spectral clustering
紋理元素	texon
閾值	threshold
拓撲結構	topology-preserving
三角化	triangulation
向量圖	vector graphics



# 第一章

## 緒論

在電腦圖學的領域中，透過電腦所生成的繪圖有兩種截然不同的類別，分別是向量圖(vector graphics)及點陣圖(raster graphics)。向量圖是一種由一群幾何基元(primitive)所構成的數位影像。這些基元—像是點、線、曲線、多邊形等等，都是基於數學方程式所構成。而點陣圖則是另一種的描述圖片的表示方式，它使用像素(pixel)陣列來定義影像，每一個像素都可以儲存不同顏色資訊。

這兩種表示方式都各有其優缺點。向量圖最大的優點是可以自由地縮放圖片大小，而且不會損失圖片的品質。原因是向量圖只需要簡單地將構成的基元乘上縮放的比例，就可以產生不同大小的結果。如圖 1 所示，當向量圖被放大時，每個基元仍然保有自己的特徵。這種性質被稱為解析度獨立(resolution-independence)。相反地，點陣圖則具有解析度相依(resolution-dependence)的性質，若將點陣圖放大，則會損失圖片的品質。

向量圖的另一個優點是很容易編輯。由於向量圖是由幾何基元所組成，因此可以很容易地透過幾何變換或修改顏色來編輯向量圖中的物體。另一方面，點陣圖中的像素彼此之間並沒有明確的相關性，因此要編輯點陣圖就沒有那麼直觀。若要編輯圖片中的某個物體，只能透過修改每個獨立的像素來達成。如圖 1 所示，當輸入的圖片(a)轉換成點陣圖(b)後，如果要在點陣圖中獲得眼睛的資訊，只能透過圈選包含眼睛在內的臉部區域(c)；但在向量圖(d)中，可以很簡單地將整個眼睛搬移到新的位置，完全不會影響到臉部的資訊(e)。此外，容易編輯的特性也讓向量圖在製作動畫上比點陣圖容易。只要在修改每一個關鍵影格(key frame)上的幾何基元，就可以讓美術人員迅速地做出動畫。

雖然有上述這些優點，但向量圖的繪製工具對於表現複雜顏色組成的資訊，只能提

供有限的支援。舉例來說，向量圖中的物體的形狀及顏色，無法很有效地呈現一些特殊風格的筆觸，像是複雜的陰影變化、光澤反射效果，以及各種紋理的顏色變化等等。使用點陣圖就可以很容易地捕捉並表現複雜的影像，因此通常被用來表現照片或寫實影像。

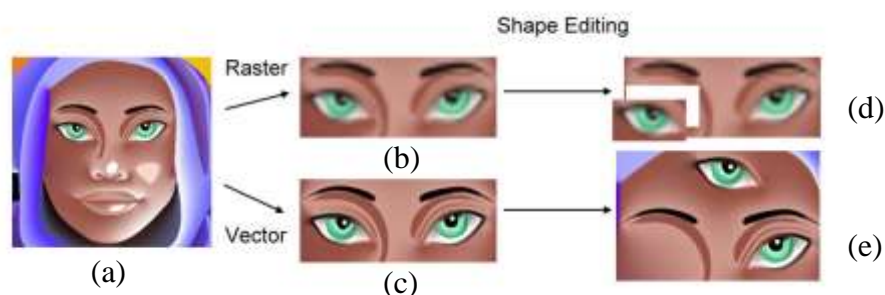


圖 1 點陣圖 v.s 向量圖

擴散曲線(diffusion curve)是 Orzan 等人[1]所提出的一個新的向量圖形基元。這是一種在曲線上給定顏色，然後從曲線上向被分割的兩側擴散顏色的曲線。由於這種擴散方式可以將顏色擴展到整張圖片，因此在沒有幾何基元的區域上，仍然會因為周圍擴散曲線的擴散效果，而有著顏色的變化。由於擴散曲線有這樣的特性，因此比起一般的向量圖表示式，能表現更為複雜的顏色組成。

本論文基於擴散曲線的特性，提出了一個點陣圖向量化的自動化方法。首先，我們透過高性能的輪廓偵測(contour detection)器，從點陣圖中萃取(extract)輪廓資訊。將取得的輪廓資訊透過輪廓追蹤(contour tracing)演算法，轉換成擴散曲線的主要構成元素——貝茲曲線(Bezier curve)；接下來，我們從點陣圖中萃取擴散曲線兩側的顏色及模糊(blur)資訊；最後，我們將這些取得這些的資訊轉換成擴散曲線，再使用以圖形處理器(GPU)為基底的顯像系統，對以擴散曲線為基底的向量圖進行顯像。透過我們的自動化方法所產生的向量圖，其結果會很接近原始的點陣圖內容。

## 第二章

### 文獻回顧

在向量圖的表示方法上，有許多各式各樣的方法被提出。在此我們就與本文有關的相關文獻一一介紹。首先，針對前人所提出的向量圖形工具，在 2.1 討論。擴散曲線各種應用，則在 2.2 討論。

#### 2.1 向量圖形工具

長久以來，向量圖的基元一直都是以固定顏色、線性或輻射梯度為主。雖然美術人員可以使用這些簡單的工具來繪製向量圖，但是要表現出較複雜的梯度或模糊效果就會受到限制。Lecot 等人[2]所提出的 ArDeco 系統則使用較為複雜的梯度系統。如圖 2 所示，這個系統會先把輸入的影像分割(segmentation)成多個區域，區域內以線性或二次梯度來計算顏色。雖然區域內的顏色變化近似原本圖片的顏色，但是區域與區域之間的邊緣太過銳利，使得算出來的圖片有很明顯的分割痕跡。



圖 2 Lecot 等人[2]的 ArDeco 系統的向量圖製作方式。首先需要將圖片分割成多個區域(B)，之後計算出每個區域的顏色變化(C)，最後得到的結果如(D)。(圖片來源: [2])

在目前常使用繪圖軟體(例如：Corel Draw、Illustrator等)則是引入了由 Sun 等人 [3]所提出的漸層網格(Gradient mesh)，如圖 3 所示，這種向量工具可以半自動化地協助美術人員設定每個網格上的顏色，而四角網格中間的顏色值便會平滑地內插四周的網格顏色。但是，若要將輸入的影像轉成漸層網格，美術人員必須要有足夠的技巧及耐心，因為他們必須要精準地將網格的分佈位置對應到圖片上，而這通常需要花上許久的時間。儘管目前網路上有提供許多教學及輔助工具來幫助美術人員製作漸層網格，但網格的複雜度及數量，往往會需要美術人員不斷地練習才能畫出有效的網格及表現出精準的顏色取樣，以符合原本圖片的效果。

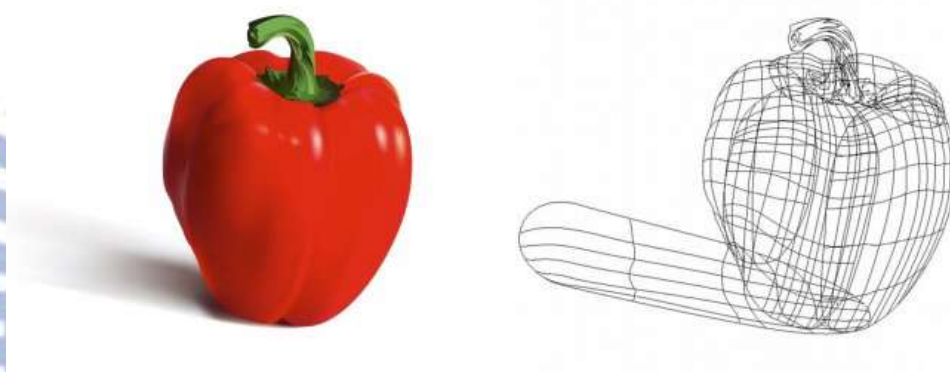


圖 3 Sun 等人[3]的漸層網格的向量圖及對應網格。此一向量圖一共使用 340 個頂點(意即同樣數量的顏色控制點)。(圖片來源: [3])

Lai 等人[4]提出了一種以拓撲結構(topology-preserving)為基礎的漸層網格，並提出了一個自動生成的圖形向量化方法。如圖 4 所示，這個方法將使美術人員不再需要花時間練習並設計，它可以自動地將圖片三角化(triangulation)，並分析圖片的幾何結構將其參數化(parameterization)，最後透過這些資訊自動地生成漸層網格。但是，這些生成漸層網格的步驟都無法即時運算，並且需要較大的儲存空間。

McCann 等人[5]則提出了梯度域(gradient domain)的即時繪圖系統。這套系統設計了梯度筆刷及梯度複製工具，如圖 5 所示，美術人員可以很容易地透過筆刷來修改圖片上的梯度值，進而製作出特殊風格的繪畫；也可以使用複製工具來複製圖片中的物件，並且貼在圖片的其他位置。這種方法其實是透過修改圖片上的梯度值來進行編輯，並沒有使用基元來組成圖片中的物件，所以美術人員在編輯上需要花上相當長的時間才能畫出想要的結果。另外，這種系統與一般的向量儲存格式相比，需要相當龐大的記憶體來

儲存圖片的梯度值。

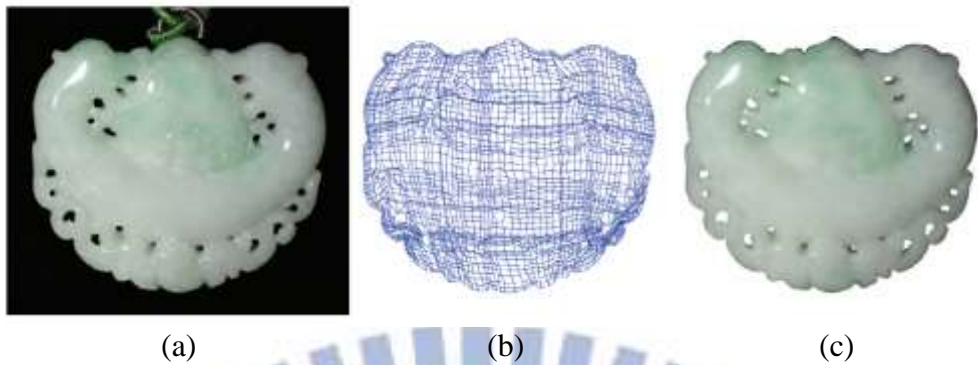


圖 4 Lai 等人[4]所提出的自動生成的拓撲結構漸層網格。(a)為輸入影像，(b)為自動生成的漸層網格，(c)為使用生成的漸層網格顯像的結果。此一自動化生成的計算時間大約為2.5分鐘。(圖片來源: [4])



圖 5 McCann 等人[5]的梯度域即時繪圖系統。美術人員從一張空白的畫布開始，利用系統提供的梯度筆刷工具作畫。繪製時間：30分鐘。(圖片來源: [5])

Selinger [6]提出了另一種名叫 Potrace 的自動向量化方法。如圖 6 所示，這種方法能將一張給定的點陣圖自動轉換成平滑的輪廓曲線，並且以向量的表示式來儲存。但是這種方法會將原本點陣圖上的一條曲線計算成兩條輪廓，產生雙重邊緣的現象。如圖 7 所示，本篇論文也使用此演算法來嘗試自動生成擴散曲線，但是顯像的結果也會產生很明顯的雙重邊緣。此外，若要使用此方法來將彩色影像轉換成向量圖，該演算法需先將點陣圖轉換成灰階再做處理，如此一來便會失去顏色的變化資訊，所計算出來的輪廓較不完整。

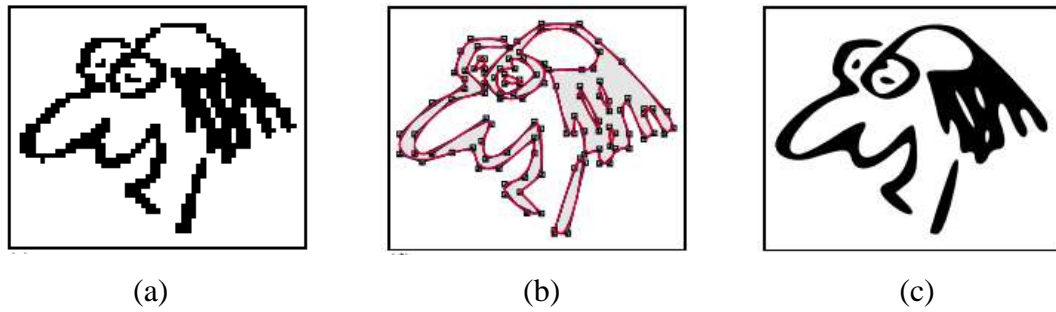


圖 6 Selinger [6] 的自動向量化方法，首先偵測給定的點陣圖的邊緣資訊，並轉換成向量曲線，再進行上色。(a)為給定的點陣圖，(b)為計算出來的向量曲線，(c)為顯像結果。(圖片來源: [6])

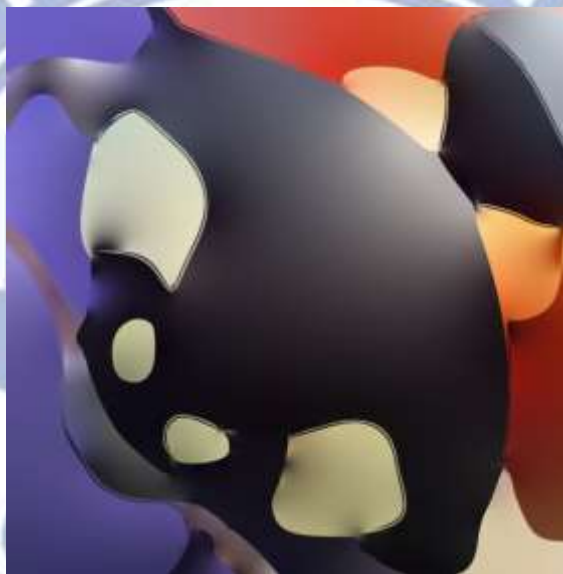


圖 7 使用 Potrace 所計算的擴散曲線顯像結果。原本點陣圖上的一條輪廓被計算成兩條，產生雙重邊緣的現象。

## 2.2 擴散曲線應用

Orzan 等人[1]所提出的擴散曲線，使用非常簡單的基元來表示向量圖形，因此過去幾年有許多人相繼提出了與擴散曲線相關的應用。Jeschke 等人[7]提出了以圖形處理器為主的拉普拉斯解答器(Laplacian Solver)，來加速擴散曲線的顯像過程，達到即時運算的結果。本論文所生成的擴散曲線亦用此方法來加速顯像。

Bowers 等人[8]則提出了以光線追蹤(ray tracing)來計算擴散曲線的擴散。這種方式延伸了擴散曲線的基元，加入了梯度、紋理、透明度，甚至是美術人員可以自行設定

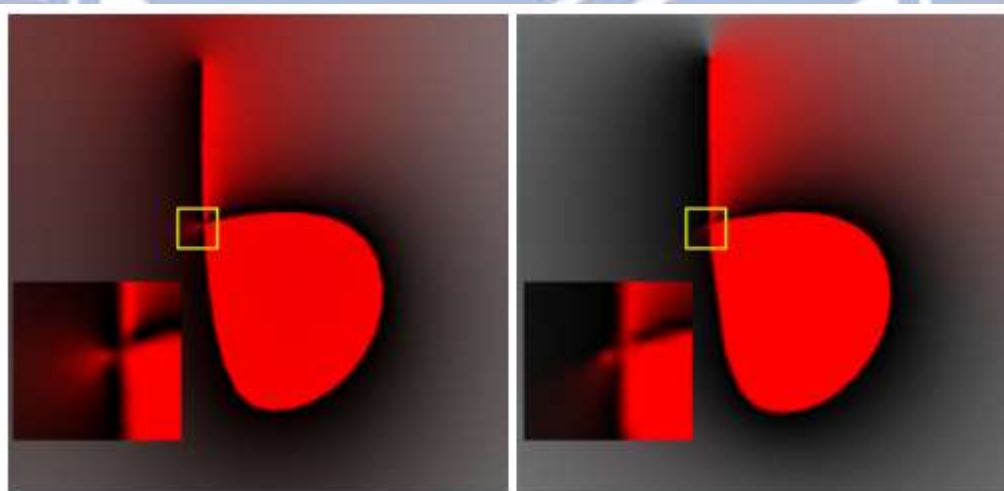
的權重資訊，讓擴散曲線能產生更豐富的效果，如圖 8 所示。除此之外，這個方法還改善了擴散曲線在交會點顯著的人工合成現象，如圖 9 所示。



(a)

(b)

圖 8 擴散曲線的顯像方法比較。(a)為 Orzan 等人[1]的方法，(b)為 Bowers 等人[8]的方法。(b)的眼睛部分的光影變化較(a)豐富。(圖片來源: [8])



(a)

(b)

圖 9 擴散曲線的顯像方法於交會點的比較。(a)為 Orzan 等人[1]的方法，(b)為 Bowers 等人[8]的方法。(b)的交會點部分所產生的人工合成現象略為不明顯。(圖片來源: [8])

Winnemöller 等人[9]將擴散曲線應用於紋理的設計上。首先讓使用者編輯擴散曲線以設計紋理元素，之後加入了法線及紋理座標等資訊，將紋理元素重複地拼貼在圖片中的衣服或掛布上面。使用者可以使用這個系統快速地將設計好的不同紋理元素拼貼在



同一張圖片中的同一塊區域，如圖 8 所示。由於增加了法線的資訊，因此可以產生更逼真的細節及陰影效果。Sun 等人[10]及 Jeschke 等人[11]則將擴散曲線延伸到3D物體的紋理貼圖上，由於向量圖的特性，這些紋理貼圖都是解析度相依，而且可以隨著3D物體曲面的變化產生相對應的結果，如圖 9、圖 10 所示。



圖 10 Winnemöller 等人[9]的拼貼方法。(a)使用者可以在圖片中指定要拼貼紋理元素的區域。(b)使用者可以很快地將不同的紋理元素應用在同一張圖片上。(圖片來源: [9])



圖 11 Sun 等人[10]將擴散曲線應用在3D曲面的紋理貼圖。(圖片來源: [10])



圖 12 Jeschke 等人[11]將擴散曲線應用在3D曲面的顯像上。(圖片來源: [11])

Takayama 等人[12]提出了擴散曲面(Diffusion Surfaces)的概念，是以擴散曲線作延

伸的一種 3D 曲面。與擴散曲線類似，只要在曲面兩邊定義一連串的控制顏色點，便可以產生自動擴散的立體模型。這種方法可以產生非真實感繪製(Non-photorealistic rendering, NPR)風格的立體模型，如圖 11 所示。

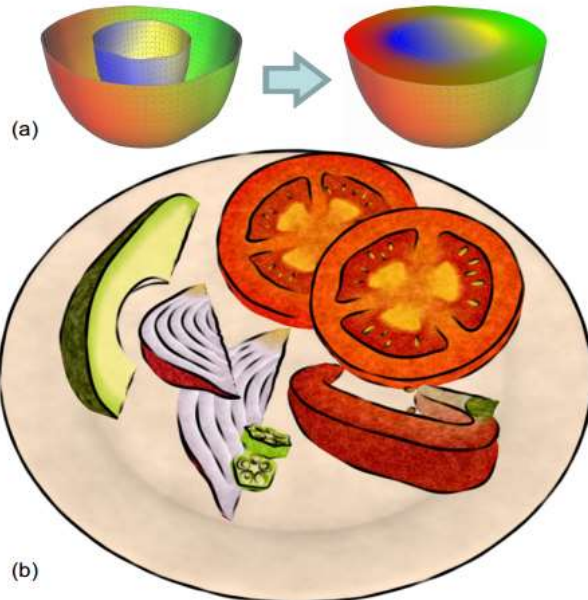


圖 13 Takayama 等人[12]所提出的擴散曲面。(a)給定曲面兩面的顏色值，就可以將顏色擴散到整個立體模型。(b)利用擴散曲面所生成的NPR風格的3D模型。(圖片來源:[12])

Hnaidi 等人[13]將擴散曲線的擴散方法延伸應用到地形的生成上，如圖 11 所示。由於擴散曲線的顏色是從顏色控制點往外擴散生成的，所以只要用相同地方法，給定地形的主要輪廓曲線，以及曲線上頭的梯度值，整塊地形每個位置的梯度也可以從曲線上向外擴散而得。

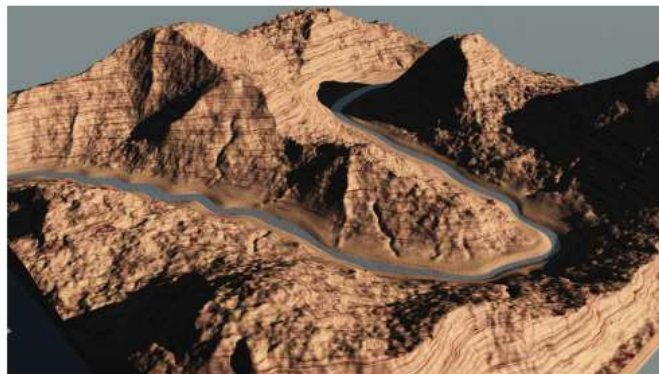


圖 14 Hnaidi 等人[13]使用擴散曲線的擴散方法所生成的複雜地形結果。(圖片來源:[13])

# 第三章

## 方法

本章節將描述點陣圖自動向量化的方法。首先，我們會在 3.1 介紹整個方法的概觀。接下來，由於本方法是將點陣圖轉換成擴散曲線的向量表示式，因此我們先在 3.2 介紹擴散曲線的結構及顯像方法。最後，我們將闡述整個自動向量化的演算法，包含在 3.3 說明輪廓偵測器， 3.4 說明輪廓追蹤演算法， 3.5 說明顏色及模糊資訊的萃取。

### 3.1 概觀

我們的目標是要將點陣圖自動轉換成擴散曲線，並且讓產生的向量圖盡可能地接近原本的點陣圖。為了達到此目的，我們從點陣圖中萃取形狀、顏色及模糊資訊，並透過這些資訊產生擴散曲線。圖 15 顯示我們的方法架構圖。首先，我們的系統輸入一張由使用者所給定的點陣圖。接下來，針對這張點陣圖進行輪廓偵測以獲得輪廓資訊。由於產生出來的輪廓資訊是以像素陣列的方式儲存，因此我們再透過輪廓追蹤演算法，將輪廓資訊轉換成貝茲曲線。至於擴散曲線所需要的顏色及模糊資訊，則是沿著輪廓從點陣圖上萃取。當上述這些資訊都獲得了以後，我們便生成擴散曲線，並使用以圖形處理器為基底的顯像系統進行顯像。

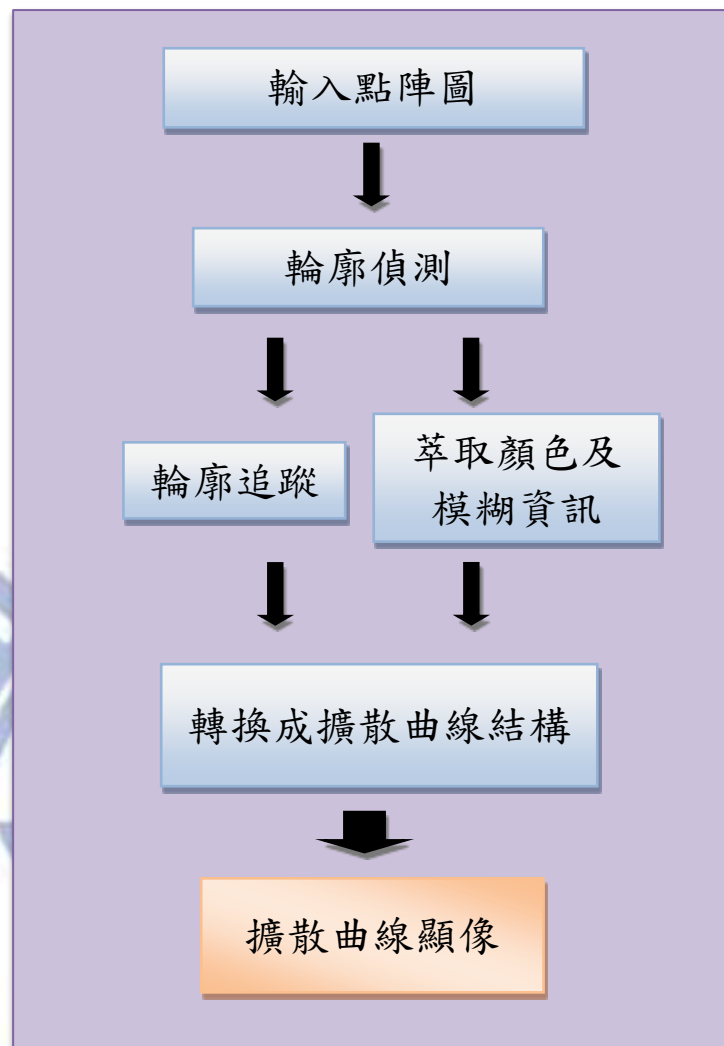


圖 15 方法架構圖。

## 3.2 擴散曲線

擴散曲線是由 Orzan 等人[1]所提出的一個新的向量圖型基元。在這一節中，我們將介紹擴散曲線的資料結構，以及我們如何透過這個幾何基元有效地顯像成一張圖片。

### 3.2.1 資料結構

如圖 16，擴散曲線的結構包含底下這些基礎元素：

1. 有著一組控制點(control point)的三次方貝茲樣條曲線(cubic Bezier spline)，如圖 16(a)。

2. 兩組顏色控制點(color control point)  $C_l$  及  $C_r$ ，表示曲線左右兩側的顏色限制條件，如圖 16(b)。
3. 一組模糊控制點(blur control point)  $\Sigma$ ，定義曲線左右兩側的顏色變化的平滑程度，如圖 16(c)。

在擴散曲線上的顏色及模糊屬性，可以沿著曲線產生豐富的顏色變化。透過控制點與控制點之間的線性內插，可以將曲線上的每個位置都有顏色及模糊屬性。各種屬性的控制點可以獨立儲存，因為他們彼此之間沒有相關性。因此每一條擴散曲線都可以由四個獨立的陣列來儲存所有的控制點資訊。

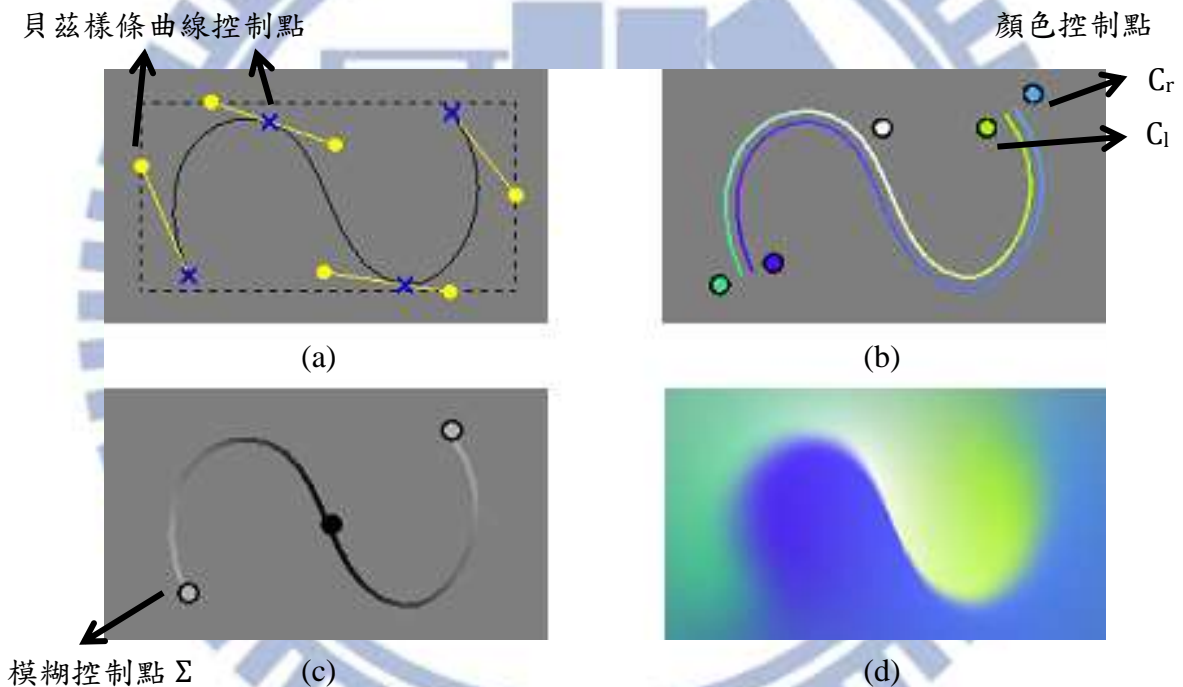


圖 16 擴散曲線。(a)三次方的貝茲樣條曲線，是擴散曲線的基本組成。黃色及藍色點代表控制點。(b)顏色控制點，曲線的左右兩側皆定義顏色，並沿著曲線做線性內插。(c)模糊控制點，同樣地沿著曲線做線性內插。(d)為經過顏色擴散及模糊後的擴散曲線結果。(圖片來源: [1])

### 3.2.2 顯像方法

擴散曲線的顯像方法如圖 17，一共分成三個步驟：

1. 產生一個柵格化(rasterization)的顏色來源影像。位在貝茲樣條曲線兩側上的像素的顏色取決於顏色屬性，剩餘的像素則未上色。
2. 對顏色來源影像進行擴散，反覆這個過程直到整張圖片都有顏色。
3. 使用擴散曲線上的模糊屬性，將運算好的圖片進行模糊化。

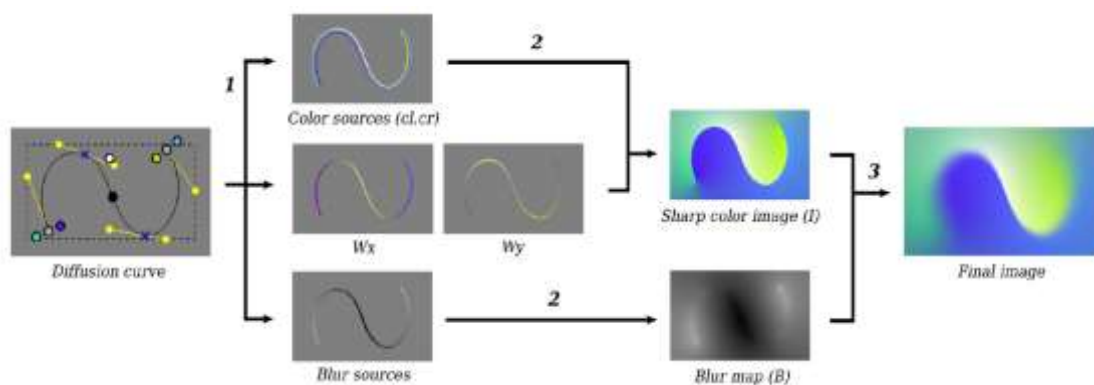


圖 17 擴散曲線顯像方式。階段1：透過柵格化獲得顏色及模糊資訊，以及梯度域  $W = (W_x, W_y)$  資訊。階段2：將顏色及模糊資訊擴散到整張圖片。階段3：將得到的顏色圖片透過模糊貼圖進行模糊化，得到最終的結果。(圖片來源: [1])

### 3.2.2.1 顏色來源

為了要將顏色從擴散曲線向外擴散，首先要對曲線上的顏色來源  $C$  進行顯像。由於顏色控制點  $C_l$  及  $C_r$  上的顏色值已經確定，因此曲線上其他位置的顏色值就直接從  $C_l$  及  $C_r$  向曲線行進的兩個方向做線性內插即可。但由於曲線左右兩側的顏色位置非常接近，如果直接在曲線所在位置著上內插後的顏色，之後的顏色擴散時會產生不自然的視覺效果。因此我們將顏色著在曲線外一段距離  $d$  (這裡我們採用  $d = 3$  像素) 的地方，確保不會有上述的問題發生。最終得到計算完成的顏色來源  $C$ ，如圖 18(a) 所示。

另外，我們也計算整張圖片上每個像素的顏色梯度值，作為擴散時的限制條件。我們將梯度域  $W$  分成  $x$  方向的  $W_x$  及  $y$  方向的  $W_y$ 。對於在曲線以外的像素來說， $W_{x,y} = 0$ ；而對於每個在曲線上的像素來說， $W_{x,y} = (C_l - C_r) * N_{x,y}$ ，其中  $N_{x,y}$  代表此像素在該曲線上的法向量。計算結果如圖 18(c)(d) 所示。

最後，我們也計算曲線上的模糊數值。在模糊控制點  $\Sigma$  位置上的模糊數值已經確定，因此曲線上其他位置的模糊數值只要從模糊控制點向曲線行進的兩個方向作線性內

插即可。最終得到一張模糊資訊來源  $\sigma$ ，如圖 18(b) 所示。

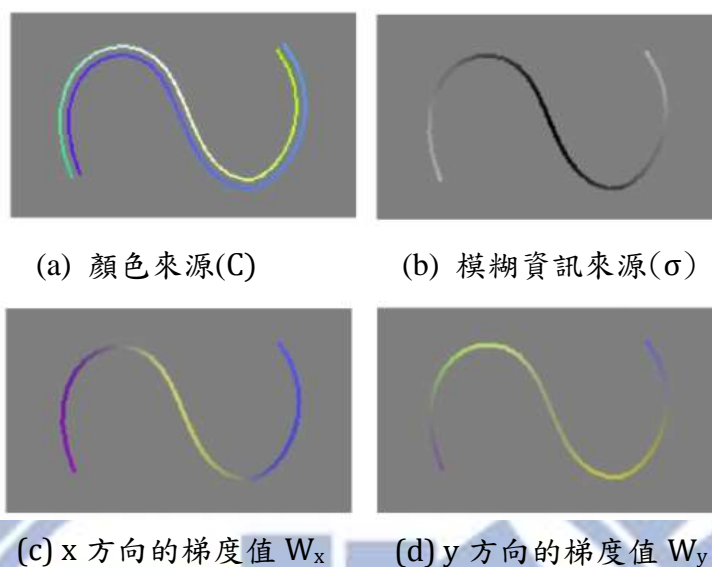


圖 18 柵格化所得擴散曲線所需要的各種基礎元素。(圖片來源:[1])

### 3.2.2.2 顏色擴散

有了 3.2.2.1 所計算的顏色來源  $C$  及梯度域  $W$  之後，接下來就可以對整張圖片進行擴散。我們使用類似於 Pérez 等人[14]的方法，透過解出底下的泊松方程式 (Poisson equation) 將顏色擴散到整張圖片：

$$\Delta I = \text{div } W$$

如果  $(x, y)$  像素有顏色值， $I(x, y) = C(x, y)$

$\Delta$  及  $I$  為拉普拉斯及散度(divergence)運算子。

由於計算泊松方程式的解需要很大的線性系統，因此我們採用 Jeschke 等人[7]所提出的以圖形處理器為主的拉普拉斯解答器(Laplacian Solver)，來加速計算的過程，已完成擴散曲線顏色的顯像。顏色顯像的結果如圖 19(a) 所示。

### 3.2.2.3 模糊資訊擴散

與 3.2.2.2 的方法類似，我們同樣也針對之前所計算的模糊資訊進行擴散。與顏色擴散不太一樣的是，曲線上的模糊數值不像顏色資訊需要向外移動一段距離，因此沒有

任何梯度的限制。計算方法如下：

$$\Delta B = 0$$

如果 $(x,y)$ 像素在曲線上， $B(x,y) = \sigma(x,y)$

其中  $B$  為欲計算的模糊貼圖， $\sigma(x,y)$  為在該點上的模糊數值。

同樣地，我們也使用Jeschke 等人[7]的以圖形處理器為主的拉普拉斯解答器來計算此泊松方程式，計算結果如圖 19(b) 所示。

### 3.2.2.4 綜合結果

當顏色及模糊的顯像結果都已經計算完畢以後，最終就是將兩張圖片進行相乘得到最終的結果，如圖 19(c) 所示。

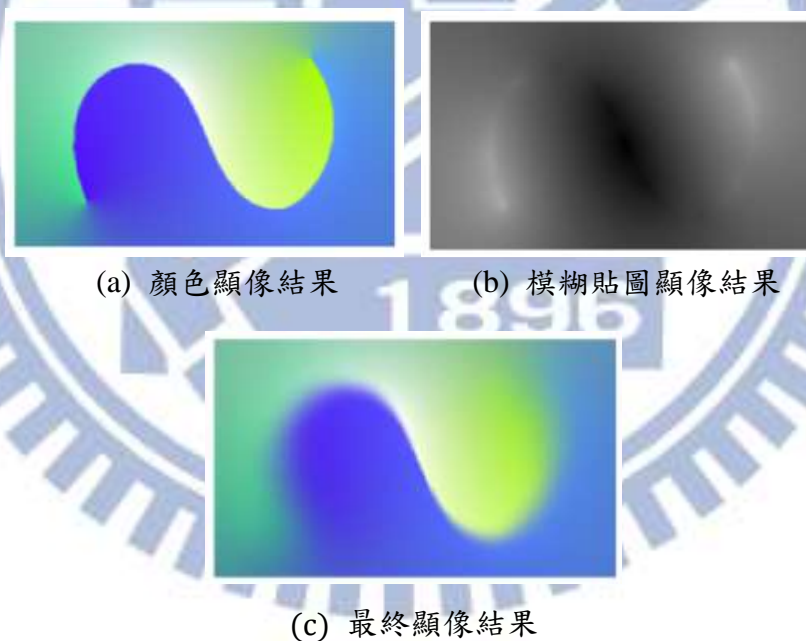


圖 19 使用泊松方程式進行擴散所產生的顏色貼圖結果(a)及模糊貼圖顯像結果(b)。以及將兩張圖片相乘所得到的最終顯像結果(c)。(圖片來源: [1])



### 3.3 輪廓偵測

由於擴散曲線是由控制點、顏色控制點及模糊控制點組成，因此需要從點陣圖中萃取這些資訊。針對控制點的位置資訊，我們採取輪廓偵測的方式。目前存在著許多方法來找出圖片中的輪廓資訊。Arbelaez 等人[15]提出了一個高效能的輪廓偵測器，可以偵測圖片中的局部及全域的影像資訊。這個輪廓偵測器使用光譜分群(spectral clustering)的方法，將圖片中的每個像素，根據其周圍一定範圍內的亮度(brightness)、顏色及紋理的梯度量值來進行分群。

#### 3.3.1 方向梯度量值

Arbelaez 等人[15]所提出的輪廓偵測器的基本構成，是從一張圖片中計算各種強度(包含亮度、顏色及紋理)的方向梯度量值(Oriented gradient magnitude)  $G$ 。方向梯度量值的計算方式，如圖 20 所示。首先，在圖片中的每個位置  $(x,y)$  上放置一個圓盤(a)，並且以方向  $\theta$  為分界分成兩半。針對每個半圓，我們擷取被覆蓋的所有像素的強度值，並使用直方圖進行統計。該位置  $(x,y)$  的方向梯度量值  $G$  的計算方式，是根據兩半圓  $g$  及  $h$ ，算出  $\chi^2$  距離(卡方距離)：

$$G(x,y) = \chi^2(g,h) = \frac{1}{2} \sum_i \frac{(g(i) - h(i))^2}{g(i) + h(i)}$$

當每個位置的梯度量值都計算完畢後，我們再使用二階 Savitzky-Golay 濾波器[16]來強化區域最大值，並且平滑化在方向  $\theta$  時多重偵測到的最大值。

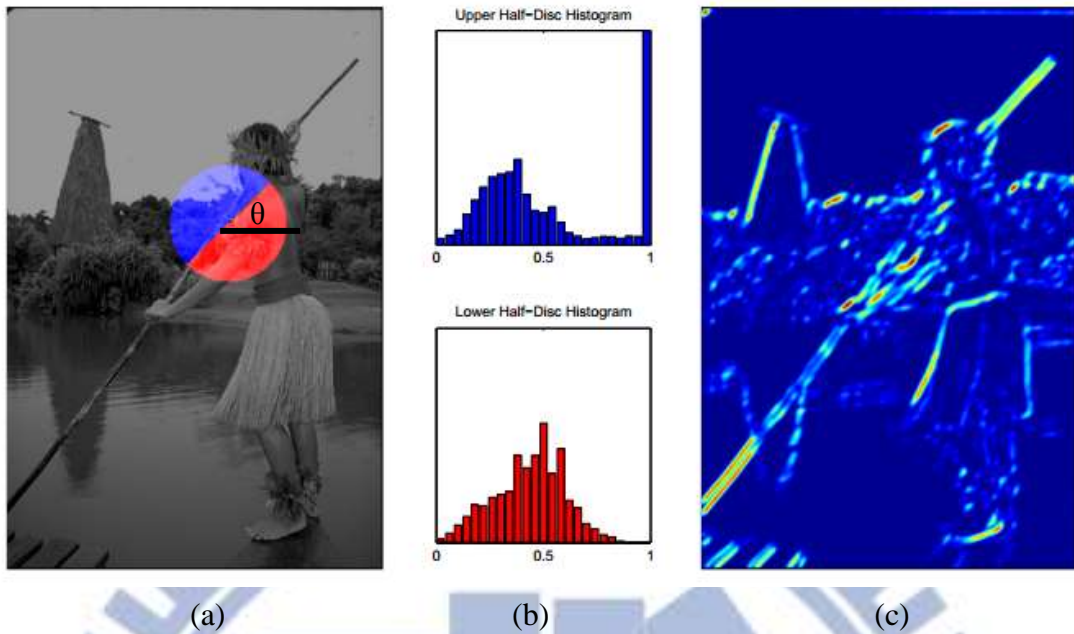


圖 20 Arbelaez 等人[15]方向梯度量值計算方法。給定一張圖片，此方法計算在方向  $\theta$  時某個位置的方向梯度量值。(a)顯示在位置  $(x,y)$  上的圓盤，藍色及紅色半圓表示  $g$  及  $h$ 。(b)的兩張直方圖即為  $g$  及  $h$  兩半圓所統計的強度值。(c)顯示計算  $X^2$  距離後，再使用二階 Savitzky-Golay 濾波器[16]所產生的方向梯度量值。這個例子使用半徑為5像素的圓盤，方向  $\theta = \frac{\pi}{4}$  來計算。(圖片來源: [15])

### 3.3.2 亮度、顏色及紋理梯度

有了方向梯度量值的計算方法之後，接下來就針對圖片的各種特徵來計算不同的通道(channel)的梯度量值。首先，我們將給定的圖片轉換成 CIE Lab 顏色空間，也就是亮度  $L$ 、顏色  $a$  及顏色  $b$  三個通道，並且使用 3.3.1 的計算方法算出三個通道的方向梯度量值  $G$ ，如圖 21 所示。

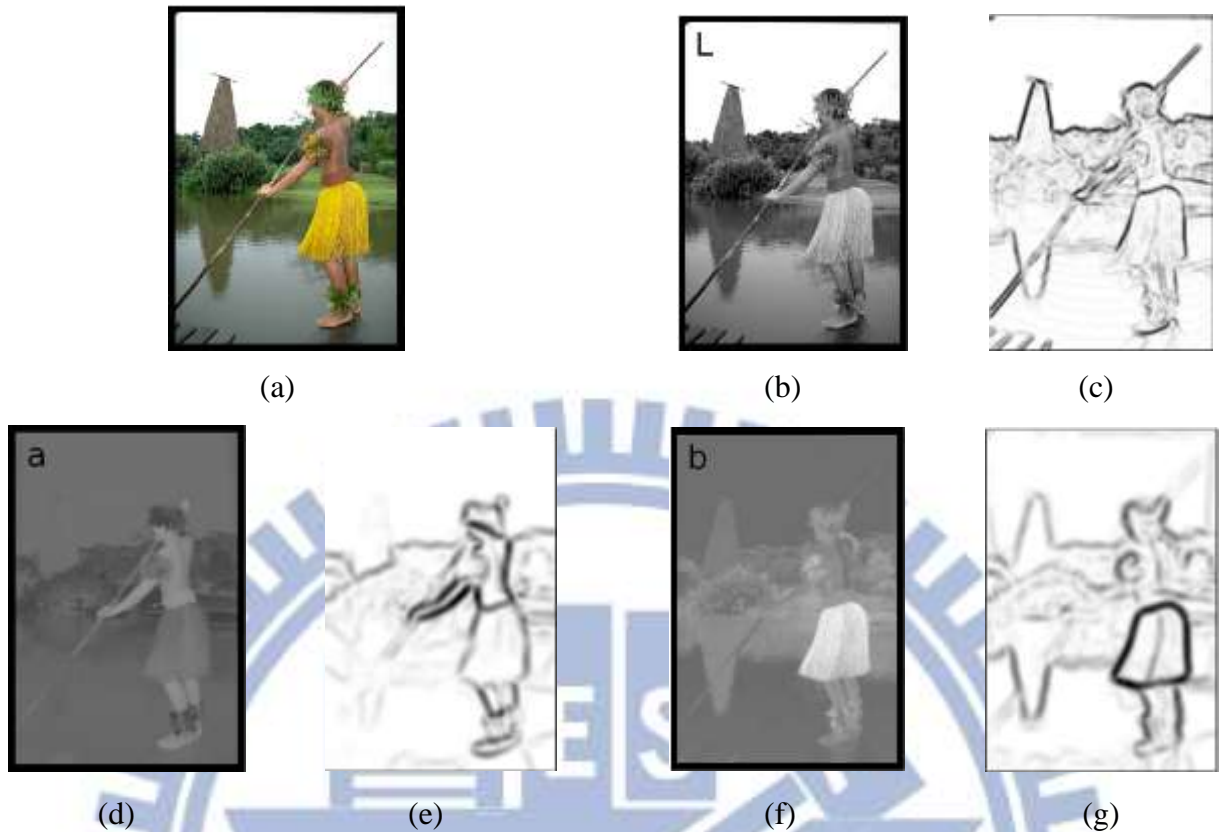


圖 21 利用Arbelaez 等人[15]的方法所計算的亮度 L、顏色 a 及顏色 b 三個通道的方向梯度量值。(a)為輸入的圖片，(b)(d)(f)為轉換成 CIE Lab 後的 L、a、b 三個通道的數值，(c)(e)(g)為三個通道所各自計算的方向梯度量值。(圖片來源: [15])

除了顏色的資訊之外，我們也考慮了紋理的資訊，因此第四個通道就是紋理通道。我們使用濾波器及分群方法，將每個像素指定到特定的紋理元素(texton) ID。首先，我們先將圖片轉成灰階，再與圖 22 的 17 個高斯導數濾波器(Gaussian derivative filter)進行卷積(convolution)。如此一來，每個像素都會對應到一個17維度的向量，我們再使用K-means分群法對這些向量進行分群，如此每個像素都會對應到一個分群過後的紋理元素 ID。最後，我們將圖片中的每個像素，用其對應 ID 的紋理元素代替。接下來我們就可以針對這張圖片，以 3.3.1 的計算方法算出方向梯度量值，如圖 23 所示。



圖 22 Arbelaez 等人[14]所提出的17個不同的高斯導數濾波器。(圖片來源: [15])

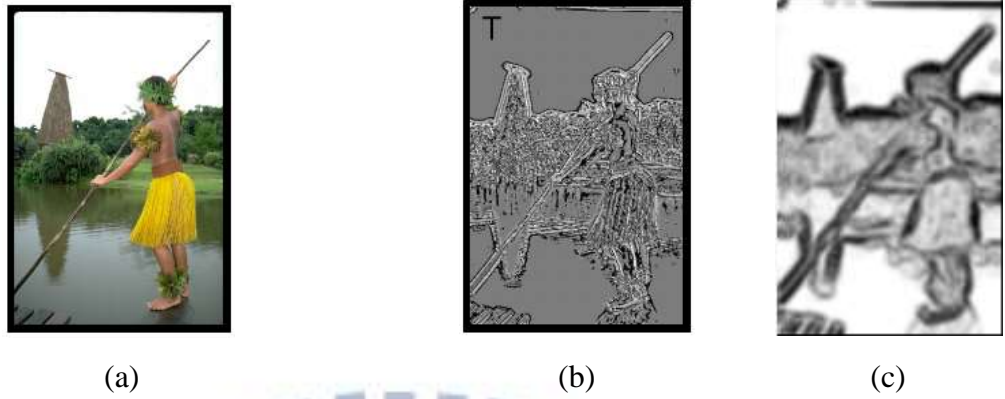


圖 23 Arbelaez 等人[15] 使用紋理元素代替原有像素的結果。(a)為輸入的圖片，(b)為取代的結果。有了紋理的資訊之後，便可以計算出方向梯度量值，結果如(c)。(圖片來源: [15])

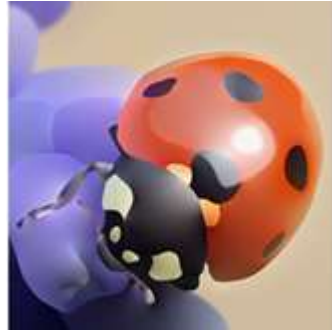
### 3.3.3 多重解析度的強度組合

為了能夠偵測更詳細的結構資訊，我們計算三種圓盤大小  $(\lceil \frac{\tau}{2} \rceil, \tau, 2\tau)$  的方向梯度量值。不同的通道也使用不同的圓盤大小(亮度通道的  $\tau$  為 5 像素，顏色及紋理通道的  $\tau$  則是 10 像素)。最終，我們將所有的資訊全部加總起來，得到一個單一的方向數值：

$$mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta)$$

其中， $s$  是指不同的圓盤大小  $(\lceil \frac{\tau}{2} \rceil, \tau, 2\tau)$ ， $i$  是指不同的通道(亮度  $L$ 、顏色  $a$ 、顏色  $b$ 、紋理  $T$ )， $G_{i,\sigma(i,s)}(x, y, \theta)$  則是位置  $(x, y)$  上，以方向  $\theta$  所計算出來的方向梯度量值。 $\alpha_{i,s}$  則是在不同解析度下，四個通道各自的權重值，Arbelaez 等人[15] 找來測試者將各種圖片進行手動的輪廓偵測，並透過訓練算出  $\alpha_{i,s}$ 。

當計算出最終的方向梯度量值之後，我們再使用光譜分群的方法，得到最終的輪廓偵測結果。如圖 24 所示。



(a)



(b)

圖 24 利用 Arbelaez 等人[15]的方法所計算出來的輪廓偵測結果。(a)為輸入的圖片，(b)為計算出來的輪廓結果。

### 3.4 輪廓追蹤

由 3.3 中所得到的輪廓資訊，仍然是點陣圖的資訊，仍然需要轉換成擴散曲線中的三次的貝茲樣條曲線。為了將輪廓的位置向量化，我們採用摩爾鄰居演算法(Moore neighborhood algorithm)來進行輪廓追蹤，每當找到一條最長的像素鏈(pixel chain)，便使用輪廓向量化方法找出與像素鏈差異性最小的貝茲樣條曲線。

#### 3.4.1 摩爾鄰居演算法

摩爾鄰居演算法是一種輪廓追蹤的方式，在此之前，我們先說明一個像素的摩爾鄰居。如圖 25(a) 所示，像素 P 周圍一共有八個相鄰的像素(P1 - P8)，這些像素便被稱為 P 的摩爾鄰居。

在開始進行輪廓追蹤之前，我們需要先將 3.3 所計算出來的輪廓資訊進行一些處理。由於輪廓資訊中每個像素的顏色值都有所不同，越接近黑色，表示是比較重要的輪廓線中的像素；反之接近白色則為較不重要的像素。我們定義一個閾值(threshold)參數  $T$ ，若每條輪廓線的顏色大於閾值，則代表這條線是會被轉換成貝茲樣條曲線。反之則會被捨棄。這麼做的原因是因為輪廓偵測出來的某些線條太過細微，一般人幾乎無法觀察出來，若將這些線條轉換成擴散曲線，那麼計算出來的結果便會有突兀的效果。

接下來介紹摩爾鄰居演算法。我們定義一個像素序列  $A$ ，用來儲存每一次萃取出來的像素鍊；我們定義  $M(s)$  為像素  $s$  周圍的摩爾鄰居， $p$  為當前邊界像素， $c$  為當前

像素所考慮的摩爾鄰居。

摩爾鄰居演算法的流程如下：

```
1  將像素序列 A 清空
2  從上到下左至右掃描整張圖片，直到找到一個大於閾值 T 的像素 s
3  將 s 放入 A 中
4  設定邊界像素為 p (p = s)
5  原路返回(移動到踏入 s 之前的像素)
6  設定 c 為 M(p) 的下一個順時針摩爾鄰居
7
8  While 如果 c 不等於 s
9    If c 與 p 的顏色值相同
10   設定 p = c
11   原路返回(將 c 設定為踏入 p 之前的像素)
12  else
13   設定 c 為 M(p) 的下一個順時針摩爾鄰居
14  end While
```

演算法 1 摩爾鄰居演算法

如圖 25(b) 所示，每當執行完一次摩爾鄰居演算法，便會找到下一個相離的像素，然後將原本圖片中，像素序列的每個像素的所在位置的顏色設成白色，表示已經造訪過這個像素。

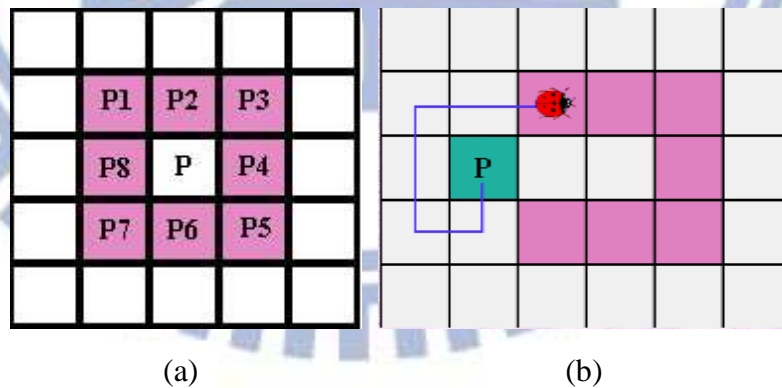


圖 25 (a)摩爾鄰居及(b)執行完一次摩爾鄰居演算法的結果。

### 3.4.2 轉換成貝茲樣條曲線

當演算法執行完畢以後，我們可以得到許多組像素鍊，接著就是要把這些像素鍊轉換成貝茲曲線。我們將連續的三個像素利用圖 26 的方法轉換成一條貝茲曲線。首先，

將第一及第三個點指定為貝茲曲線兩邊的控制點  $P_0$  及  $P_2$ 。接著將第二個點指定為當貝茲曲線參數  $t = 0.5$  時的位置，也就是  $P_{012}$ ，並且回算我們所需要的另外兩個控制點  $P_{01}$  與  $P_{12}$ ，如此一來便能將三個像素點向量化成貝茲曲線。最後，我們把所有轉換完成的貝茲曲線相連，以得到最終的貝茲樣條曲線。

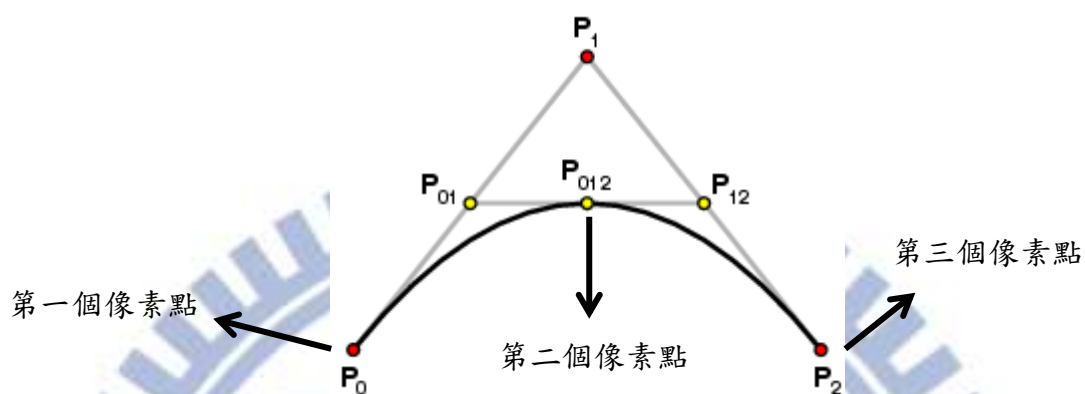


圖 26 三個點轉換成貝茲曲線的方法。我們將第一及第三個點指定為兩端的控制點  $P_0$  及  $P_2$ ，並且將第二個點指定為貝茲曲線參數  $t = 0.5$  時的位置。有了這些資訊，我們便可以回算出  $P_1$  的位置，進而算出  $P_{01}$  及  $P_{12}$  的位置。有了  $P_0$ 、 $P_{01}$ 、 $P_{12}$  及  $P_2$  的資訊，便可以構成一條三次方的貝茲曲線。

### 3.5 萃取顏色及模糊資訊

當點陣圖的輪廓已經轉換成貝茲樣條曲線之後，接下來便要從點陣圖中萃取顏色及模糊資訊。首先我們設定每條擴散曲線的顏色控制點及模糊控制點的位置，通常是與在擴散曲線上的控制點相同的位置上。接下來，從顏色控制點的位置往法線方向移動一段距離，擷取該點的顏色資訊做為顏色控制點的顏色值。以經驗來說，取法線方向 3 個像素的距離可以得到較好的效果，如圖 27(a) 所示。如此一來， $C_l$  及  $C_r$  的顏色值便根據曲線左右兩邊的法線方向來萃取。至於模糊資訊  $\Sigma$ ，則是直接從 3.3 的輪廓偵測的結果來萃取，如圖 27(b) 所示。

至此一來，擴散曲線所需要的各種屬性，包含貝茲樣條曲線控制點、顏色控制點及模糊控制點的資訊都已經萃取完成。接下來便可以直接對擴散曲線進行顯象。

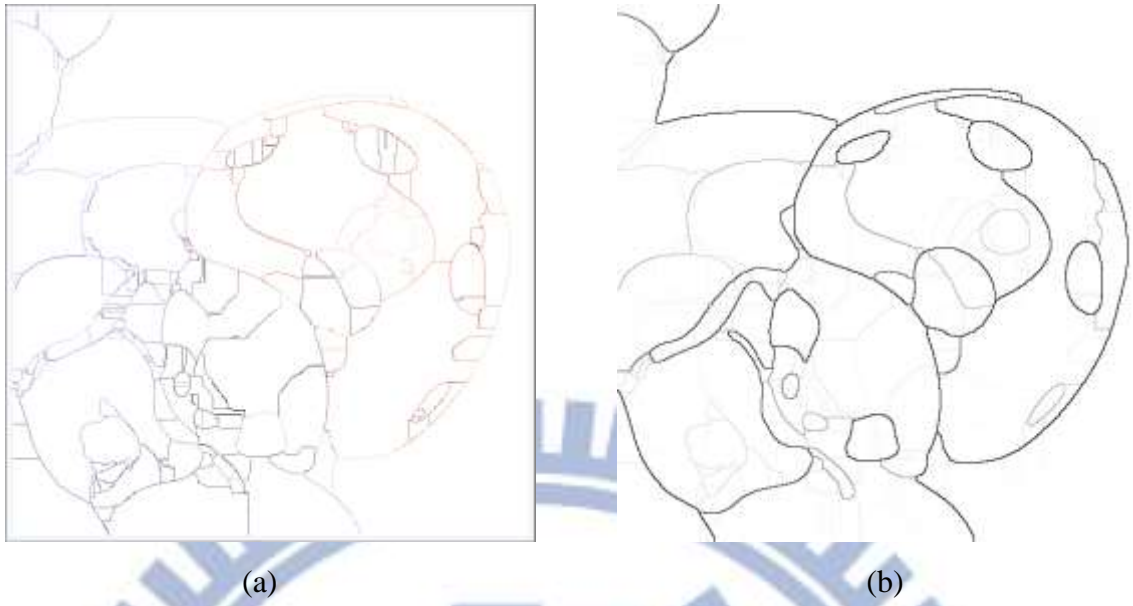


圖 27 萃取顏色及模糊資訊。(a)從點陣圖中萃取顏色資訊，之後儲存於顏色控制點  $C_l$  及  $C_r$ 。(b)從輪廓偵測中萃取模糊資訊，之後儲存於模糊控制點  $\Sigma$ 。





## 第四章

### 結果

我們提出了一個自動向量化的方法，可以將給定的點陣圖轉換成向量圖。如圖 28 所示，輸入的圖片(a)(b)(c)與我們的自動向量化方法所產生的結果(d)(e)(f)非常相近，而且我們所產生的向量圖沒有解析度相依的問題，這表示不論放大或縮小都能維持圖片的品質。

擴散曲線同樣也具有方便編輯的特性。如圖 29(b) 所示，使用者可以自行調整擴散曲線的控制點位置，讓圖片中的瓢蟲斑點有不一樣的造型變化。另外，使用者還可以修改顏色及模糊控制點的數值及位置，很容易地調整圖片中每個區塊的顏色變化，如圖 29(c) 所示。這些編輯的方式對剛接觸的使用者來說，可以很容易地學會，而且修改的結果可以即時地在畫面上顯像。因此，美術人員可以使用擴散曲線快速地修改圖片結果，或者在關鍵影格上調整控制點的資訊來製作動畫。

如圖 30 所示，對於畫家所繪製水彩畫(a)，或者非真實感繪製風格的圖片(b)(c)，透過我們的向量化方法所產生的向量圖，比起一般相片所產生的向量圖表現較好。我們認為最主要的原因是因為這些風格的圖片有比較明顯的輪廓，而且與一般相片相比，沒有較為複雜的材質變化。這些原因導致在進行輪廓偵測時，能夠完整且正確地擷取到重要的輪廓線條。而將我們的向量化方法應用在一般的相片時，就會有比較顯著的缺點。如圖 31 所示，在較為複雜的材質變化(例如草叢或樹葉的紋理)，我們的方法無法用擴散曲線來完整的呈現。此外，有些輪廓線條也無法有效地偵測。



(a)



(d)



(b)



(e)



(c)



(f)

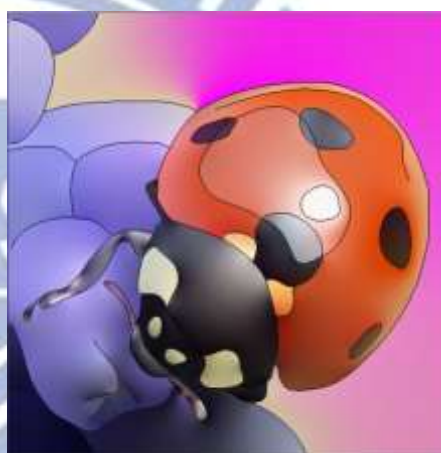
圖 28 點陣圖自動向量化的結果。(a)(b)(c)為輸入的圖片，(d)(e)(f)為我們的自動向量化所計算出來的向量圖。



(a)



(b)



(c)

圖 29 手動修改擴散曲線的顏色及線條資訊。(a)原本的擴散曲線向量圖。(b)使用者可以透過修改控制點的位置，來調整瓢蟲斑點的大小及位置。(c)使用者也可以透過修改瓢蟲最外層擴散曲線控制點的顏色，以得到不同的背景顏色。



(a)



(d)



(b)



(e)



(c)



(f)

圖 30 各種不同的風格的繪畫，透過自動向量化產生的結果。(a)為水彩畫風格的繪畫，(d)為透過我們的方法所產生的向量圖。(b)(c)為非真實感繪製風格的圖片，(e)(f)則分別為(b)(c)的向量化結果。



(a)

(b)

圖 31 一般相片使用我們的方法進行向量化結果。(a)為輸入的點陣圖，(b)為透過我們的向量化方法所產生的結果。結果顯示，較為複雜的材質部分無法用我們的擴散曲線完整的呈現，而且會遺漏許多輪廓的資訊。

在效能部分，我們使用 Jeschke 等人[7]所提出的以圖形處理器顯像擴散曲線的系統，在搭載NVIDIA Geforce GTX 460顯示卡的 Intel core 2 duo E7500 2.93GHz 的個人電腦上，輪廓偵測需花費1-2小時的計算時間，計算擴散曲線需花費20-30秒的計算時間，而擴散曲線的顯像則可以達到25-30fps的即時顯像效果。本篇大部分的圖片皆是以此系統進行顯像。

我們的方法與漸層網格[3]或 Potrace [6]相比，也有比較好的結果。圖 32 為我們的方法與漸層網格[3]的比較結果。漸層網格需要使用者不斷地練習，花費一定時間(一般來說，熟練的美術人員要花上30分鐘以上的時間)才能設計出想要的結果；相較之下，我們的方法不需要手動繪製及調整網格的頂點就可以自動完成。因此使用者不需要有任何經驗及美術知識就可以快速的得到向量圖。

圖 33 則是與Potrace[6]方法的比較。由於我們的方法採用Arbelaez 等人的[15]的輪廓偵測器，比起Potrace[6]使用Canny輪廓偵測器，輪廓的偵測效果較好，因此顯像結果上可以呈現更多細微的邊緣資訊。像是圖片中央的瓢蟲身上的光暈效果，Potrace[6]

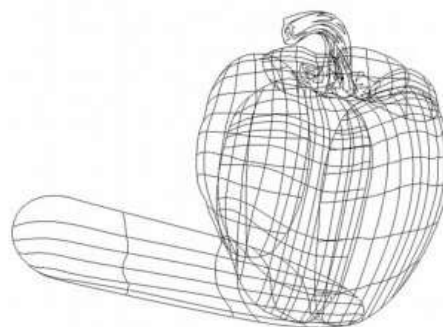
的方法無法完整呈現。



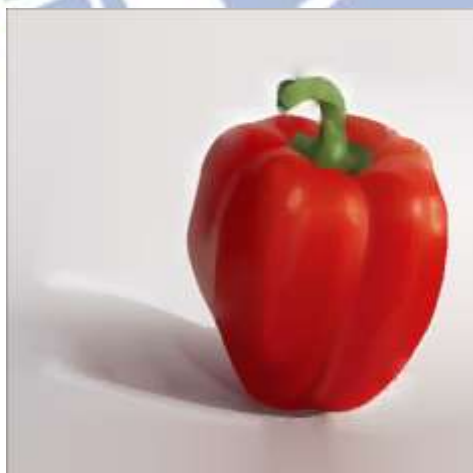
(a)



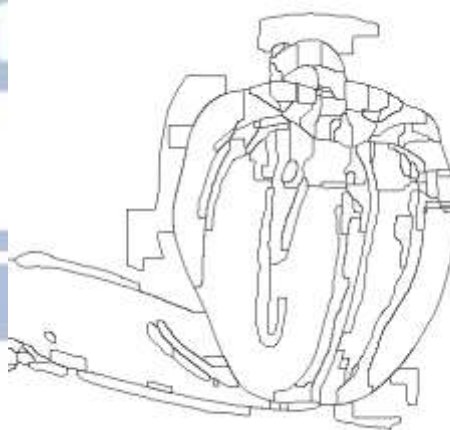
(b)



(c)

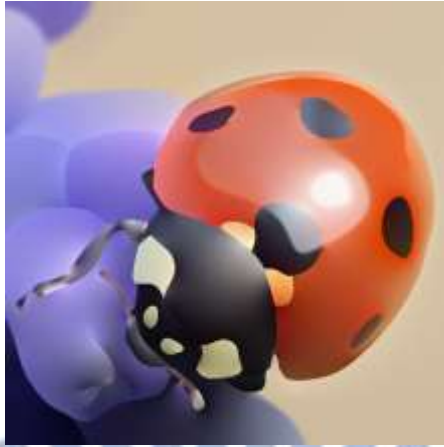


(d)

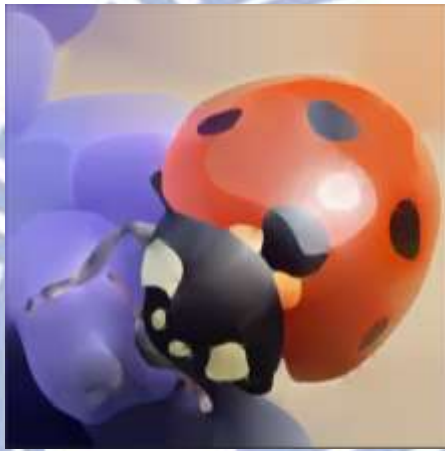


(e)

圖 32 我們的方法與漸層網格[3]的比較結果。(a)為輸入的圖片。(b)為漸層網格的結果，需要340個頂點構成，如(c)所示。(d)為我們的方法所計算的結果，只需要85條擴散曲線，如(e)所示。



(a)



(b)



(c)

圖 33 我們的方法與Potrace的比較結果。(a)為輸入的點陣圖，(b)為我們的向量化結果，(c)為Potrace的結果。

# 第五章

## 貢獻及未來展望

### 5.1 貢獻

過去以來，一直有許多點陣圖向量化的方法被提出。本篇論文提出了以擴散曲線為基底的點陣圖自動向量化演算法。

為了讓向量化的結果接近原始點陣圖，我們使用高性能的輪廓偵測器，從點陣圖中萃取輪廓資訊並將其向量化。並且從點陣圖中萃取擴散曲線兩側的顏色及模糊資訊。最後再對以擴散曲線為基底的向量圖進行顯像。

我們的演算法能自動將點陣圖轉換成向量圖，而且可以透過修改擴散曲線的控制點資訊，即時修改圖片的顏色、模糊及位置資訊。對於美術人員來說，使用此自動化方法可以快速地將圖形向量化，並且透過簡單的編輯製作成動畫。

### 5.2 未來展望

我們所使用的擴散曲線跟大部分的向量圖一樣，都無法表現像是紋理才有的細微顏色及光影變化。以過去的經驗來說，如果使用向量圖來表示紋理，將會需要大量的基元來描述紋理中細微的結構，這導致計算出來的向量圖相當地龐大，而且與原本的點陣圖相比，會產生很大的誤差。Jeschke 等人[17] 提出了使用伽柏雜訊(Gabor noise)來表現紋理變化的演算法。這個方法使用擴散曲線來表現圖片的整體結構，再透過調整伽柏雜訊的參數，讓產生出來的雜訊能產生區域性的紋理效果及光影變化。

輪廓偵測的數值高低代表著輪廓的重要程度，有些較為重要的輪廓在原本的點陣圖上並非具有很銳利邊緣，這代表著輪廓的重要性與模糊程度並無直接相關。如此一來，



本篇使用萃取輪廓偵測的結果來給定模糊控制點的數值的做法，其實並不夠妥善。若能從點陣圖中擷取更多有關模糊的資訊來設定模糊控制點的數值，效果應該會更好。

此外，本篇所使用的擴散曲線的基元，雖然可以讓使用者定義控制點的顏色，但卻無法讓使用者控制顏色的擴散程度。Bezerra 等人[18]所提出的擴散限制方法，可以有效降低控制點的數量，並且更有效地控制顏色的擴散程度及方向。



## 參考資料

- [1] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. "Diffusion curves: a vector representation for smooth-shaded images." *ACM Trans. Graph.* 27, 3, 92:1-92:8 (August 2008).
- [2] Gregory Lecot and Bruno Levy. 2006. "Ardeco: automatic region detection and conversion." In *Proceedings of the 17th Eurographics conference on Rendering Techniques (EGSR'06)*, 349-360.
- [3] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. 2007. "Image vectorization using optimized gradient meshes." *ACM Trans. Graph.* 26, 3, 11:1-11:7 (July 2007).
- [4] Yu-Kun Lai, Shi-Min Hu, and Ralph R. Martin. 2009. "Automatic and topology-preserving gradient mesh generation for image vectorization." *ACM Trans. Graph.* 28, 3, 85:1-85:8 (July 2009).
- [5] James McCann and Nancy S. Pollard. 2008. "Real-time gradient-domain painting." *ACM Trans. Graph.* 27, 3, 93:1-93:7 (August 2008).
- [6] Peter Selinger. 2003. "Potrace: a polygon-based tracing algorithm."
- [7] Stefan Jeschke, David Cline, and Peter Wonka. 2009. "A GPU Laplacian solver for diffusion curves and Poisson image editing." *ACM Trans. Graph.* 28, 5, 116:1-116:8 (December 2009).
- [8] John C. Bowers, Jonathan Leahey, and Rui Wang. 2011. "A ray tracing approach to diffusion curves." In *Proceedings of the Twenty-second Eurographics conference on Rendering (EGSR'11)*, 1345-1352.
- [9] H. Winnemöller, A. Orzan, L. Boissieux, and J. Thollot. 2009. "Texture design and draping in 2D images." In *Proceedings of the Twentieth Eurographics conference on Rendering (EGSR'09)*, 1091-1099.
- [10] Xin Sun, Guofu Xie, Yue Dong, Stephen Lin, Weiwei Xu, Wencheng Wang, Xin Tong, and Baining Guo. 2012. "Diffusion curve textures for resolution independent texture mapping." *ACM Trans. Graph.* 31, 4, 74:1-74:9 (July 2012).
- [11] Stefan Jeschke, David Cline, and Peter Wonka. 2009. "Rendering surface details with diffusion curves." *ACM Trans. Graph.* 28, 5, 117:1-117:8 (December 2009).
- [12] Kenshi Takayama, Olga Sorkine, Andrew Nealen, and Takeo Igarashi. 2010. "Volumetric modeling with diffusion surfaces." *ACM Trans. Graph.* 29, 6, 180:1-180:8 (December 2010).
- [13] Houssam Hnaidi, Eric Guérin, Samir Akkouche, Adrien Peytavie, Eric Galin. 2010. "Feature based terrain generation using diffusion equation." *Computer Graphics Forum* 29, 7, 2179--2186.
- [14] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. "Poisson image editing." *ACM Trans. Graph.* 22, 3 (July 2003), 313-318.
- [15] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. 2011. "Contour Detection and Hierarchical Image Segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 5 (May 2011), 898-916.

- [16] Abraham Savitzky, M. J. E. Golay. 1964. "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry - ANAL CHEM*, vol. 36, no. 8, pp. 1627-1639.
- [17] Stefan Jeschke, David Cline and Peter Wonka. 2011. "Estimating color and texture parameters for vector graphics." *Computer Graphics Forum* 30, 2, 523--532.
- [18] Hedlena Bezerra, Elmar Eisemann, Doug DeCarlo, and Joëlle Thollot. 2010. "Diffusion constraints for vector graphics." *In Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering (NPAR '10)*, 35-42.

