國立交通大學

生醫工程研究所

碩 士 論 文

運用於果蠅腦神經之多重立體資料顯影系統

A Multi-volume Rendering Visualization System for Fly Brain Neuron Volume Data

研 究 生：葉謹慰

指導教授：荊宇泰　教授

中 華 民 國　一百 年 九 月

運用於果蠅腦神經之多重立體資料顯影系統

A Multi-volume Rendering Visualization System for Fly Brain Neuron Volume Data

研 究 生：葉謹慰　　Student：Chin-Wei Yeh

指導教授：荊宇泰　　Advisor：Yu-Tai Ching

國 立 交 通 大 學

生 醫 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Biomedical Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年九月

運用於果蠅腦神經之多重立體資料顯影系統

學生：葉謹慰　　指導教授：荊宇泰 教授

國立交通大學生醫工程研究所

摘　　要

近年來隨著腦科學的研究不斷興起，用來輔助研究工作的多重立體資料顯影技術漸漸成為趨勢，早期的立體資料顯影技術多著重於單一立體資料之顯影上，而多重立體資料顯影有別於單一立體資料顯影在於如何讓使用者能夠在數量龐大的立體資料中，分辨各立體資料並輕易的理解其幾何上的相對關係。本論文使用表面立體資料顯影中之 Alpha Shapes 演算法來建構果蠅腦中的神經元件，並用 OpenGL 內建之圓柱體及球體來建構神經群，實現一個多重立體影像之顯影系統。

A Multi-volume Rendering Visualization System for Fly Brain Neuron Volume Data

Student: Chin-Wei Yeh Advisor: Prof. Yu-Tai Ching

Institute of Biological Engineering

National Chiao Tung University

ABSTRACT

The multi-volume rendering technique becomes a trend since the research of brain come up recently. In the past years, volume rendering technique was focused on rendering of single volume data. However, the most different between multi-volume rendering technique and single volume rendering technique is how to make a clear scene of the huge volume data for users, so that they can easily figure out the relationship of the neuron and the neuron components. We use a common surface rendering technique, Alpha-shapes, to implement the neuron components and CYLINDER and SPHERE function in OpenGL to construct neuron.

誌　謝

能在兩年的時間內完成這篇論文，首先要感謝我的指導教授 荊宇泰教授，在他這兩年的不厭其煩的指導下收穫良多，實驗室的博班學長秉璋、昌杰也都很熱心的幫忙，在實驗過程卡住時總能及時提供協助，新科博班生大威的熱心午餐總能讓我們免去大熱天曬太陽的煩惱，生活知識家科科也隨時能夠解決 lab 的大小事，跟我互相 cover 的寰宇，還有同屆的心宇、小由，大家常常互相勉勵，畢業的千千、政宇以及碩一的學弟妹阿淳、欣裕、琪琪、輝哥和隔壁 lab 的 CC、雅蓁、阿誠、小節也都惠我良多，我們親愛的隔壁 lab 之花兼 office 達人筱莉，在我碩一的時候每天一大早聽我吵著要休學，還有我們家的胖咪跟夯蕃蕃，每天準時回家陪他們，讓我碩二生活過得相當健康，以及最重要的我家二老，多年來工作不愁吃穿，一路念到研究所，尤其是省吃儉用的還天天替我禱告的媽媽，非常感謝！ 感謝上帝讓這麼多天使在我身旁，在我有所困難時都能及時得到幫助，讓我有足夠的動力能夠往前衝！！

目錄

圖目錄

表目錄

第一章　緒論

近年來 3D 影像顯影技術廣泛地應用在生物醫學及工程界，其主要目的為模擬真實世界中的人事物，並應用於教學、研究與建造等用途上，讓人們能夠透過它來了解原事物的構造與狀態。由於電腦科技發展突飛猛進，使得 3D 影像顯影技術的成本降低，且也越來越趨近於真實事物，大大的降低每次操作的成本。早期的 3D 影像顯影技術多是單一的立體資料顯影，但在腦神經系統中，由於神經系統多而複雜，若只用單一個立體資料來表示，將會讓畫面顯得相當雜亂而無法辨識，有鑑於此，本論文將針對腦神經系統做多重立體資料顯影(multi-volume rendering)之研究。

多重立體資料顯影即一次讀取多個立體資料並將其建成立體模型投影顯示出來，倘若立體資料的數量龐大，又沒有使用適當的演算法，將會讓畫面與用單一立體資料來表示神經系統一樣，讓人無法清楚辨別神經彼此之間的關係，且系統效能將會因為資料量的因素變得非常低落。本系統旨在將使用者有興趣的神經及神經元件清楚的呈現出來，並能夠及時的操作系統不須等待太長的時間即可看見結果。

目前常見的立體資料顯影技術非常多種，大致上可分為直接立體資料顯影(direct volume rendering)，例如 ray tracing、3D texture-based 及表面立體資料顯影(surface rendering)例如 marching cubes 及 Alpha shapes 等，本論文選擇表面立體資料顯影的 Alpha shapes 演算法來實作。 Alpha shapes 常用於細胞蛋白質結構的模擬，為一個能給一群散佈在空間中之點集定義「形狀」的方法。此外，為了能夠讓使用者清楚的分辨神經之間的關係，實作中加入了物件透明化、光照明暗、客製化色彩設定及神經元件的隱藏等效果。其中，物件透明化使用了常見的 alpha blending 演算法，讓分布在其他物體中的物件也能被觀察而不被遮蔽，運用此演算法時，電腦繪製幾何元件的順序必定要是離視點由遠而近，故在將物件的幾何元件送至 GPU 前要先做過排序的動作；光照明暗則讓物件看起來更加立體逼真，

使用的是目前最常被使用的 Phong lighting model[1] 搭配 Phong shading，要使用此效果，必須提供每個頂點的法向量資訊，通常頂點的法向量資訊不會在原始資料裡頭，所以必須在繪製之前將頂點法向量算出並存於頂點的資料結構中；客製化色彩設定則讓使用者更能區分不同的物件之分布，本系統是選用 QT 的 Qcolor 函式來完成視窗化調變顏色的功能。

本篇論文結構大致如下：第一章是介紹論文的動機、目的及使用的方法，第二章是介紹立體影像顯影技術的基本原理，第三章是介紹在本系統中用來讓多重立體資料顯影所表達的資訊看起來更清晰的效果，第四章為實作成果及系統效能，第五章為未來研究的方向，最後則是參考文獻。

第二章　體積影像顯影技術

2.1 立體資料顯影

現今社會人類為了模擬或做研究，時常將物體做三維的取樣後，再將其重建回原始樣貌。取樣出來的三維資料稱做立體資料(Volume data)，而將其重建並繪出原始樣貌的方法就稱為立體資料顯影(Volume rendering)。

立體資料顯影(Volume rendering)係將立體影像資料以投影轉化成平面影像的一種方式。現今的體積影像顯影技術大致可分為直接體積影像顯影(Direct volume rendering)以及表面體積影像顯影(Surface volume rendering)。直接立體資料顯影是模擬 X 光線穿透物體而衰減的物理現象，由一個決定的視點，沿著其方向對體積影像做積分運算而得的投影，此方法模擬真實世界的光學原理，最後所呈現的結果就像常見的 X 光片一樣，光線較能穿透的地方灰階值較低，相反的，光線較難穿透的部位，灰階值較高，能夠表現出細部區域的狀態，如下圖：

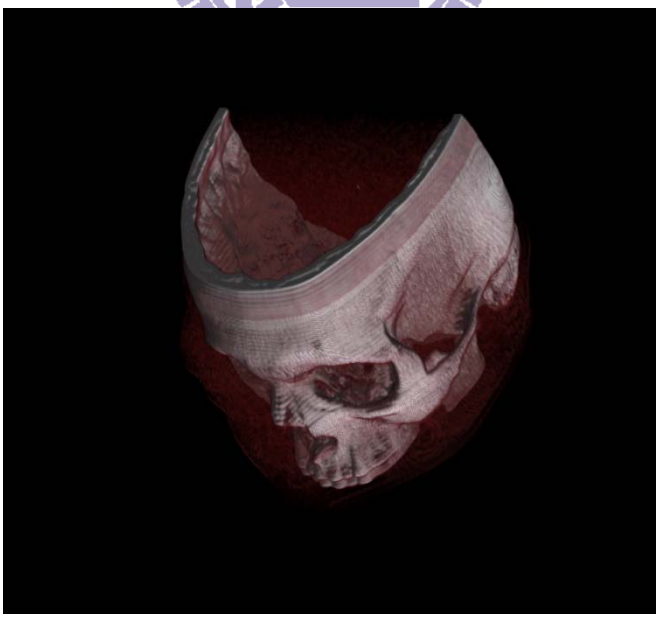

圖2.1　直接立體資料顯影

(資料來源：Wikipedia, http://en.wikipedia.org/wiki/Volume_rendering)

表面體積影像顯影則是將使用者所需求的部分，藉由計算機幾何學及 3D 圖學的演算法，從立體資料中計算出該表面的多邊形，然後再將這些多邊形加以顯現，使用者最後看到的為一個物體的表面，如下圖：

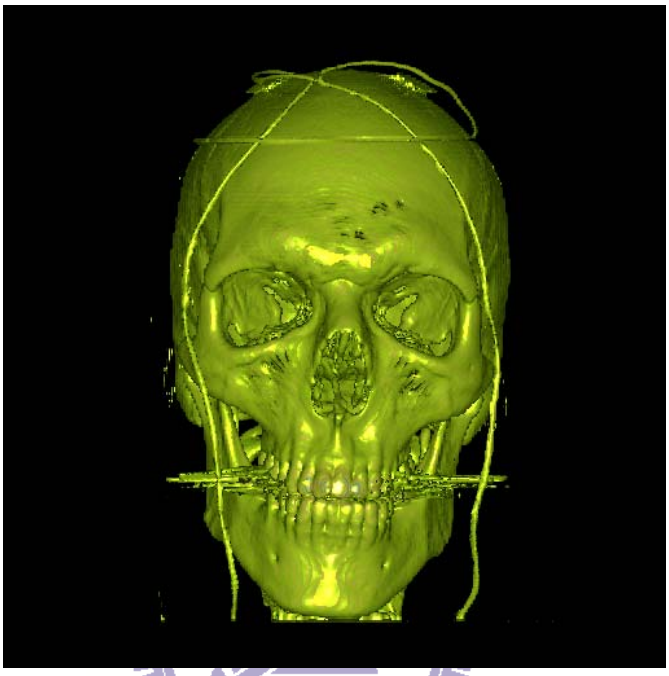

圖2.2　表面立體資料顯影

(資料來源：http://www.aravind.ca/IntelliVis)

由於在此系統實作內容為果蠅腦神經在腦中及其腦神經元件的分布狀況，只需要表面體積影像顯影技術來將腦神經元件的表面顯現出來。 表面體積影像技術至今已有許多成熟演算法，常見的有 Lorensen[2]所提出的 Marching Cubes、Edelsbrunner[3]的 Three-Dimensional Alpha Shapes。

2.2 Marching Cubes

Marchine Cubes 在 1987 年由 Lorensen 提出，為一常見的表面立體資料顯影演算法，使用者根據他們想要得到的表面資訊值(Isovalue)來對立體資料做存取，輸出的特定表面(Isosurface)由立體資料中的單位元素(voxel)根據表面資訊值來內插而得，作者提出 15 種基本的幾合樣式來拼湊出各種內插情況，如下圖：

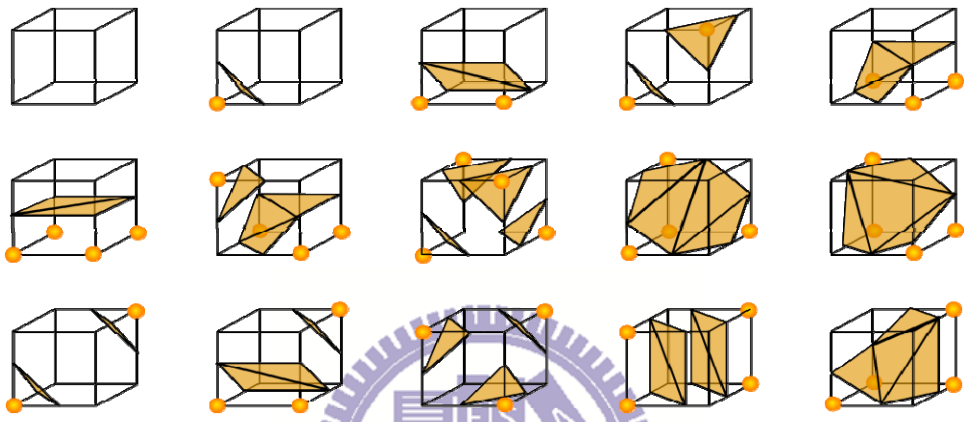

圖2.3　Marching Cubes的15種基本狀況

(資料來源：Wikipedia, http://en.wikipedia.org/wiki/Marching_cubes)

但此方法會出現一些模稜兩可的情況，後來有些人提出了改善方法，其中又以 Gregory M. Nielson[4] 的效果最佳，下圖為 Marching Cubes 所建立之模型：

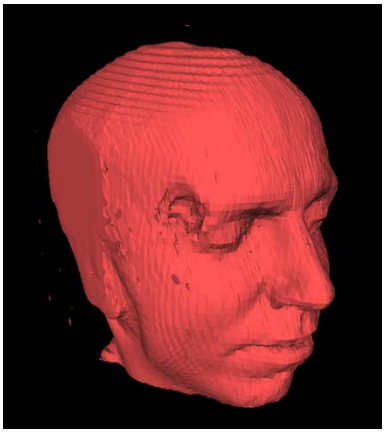

圖2.4　Marching Cubes建立之模型

(資料來源：Wikipedia, http://en.wikipedia.org/wiki/Marching_cubes)

2.3 Three Dimensional Alpha Shapes

在工程界或生物醫學界中，時常會將物體取樣成一組三維座標點集，必要時也必須將這些點集還原成原始物體的形狀，因此，如何給予這些點集一個屬於它們的形狀變的相當重要。 Alpha shapes 是一種建構于 Delaunay triangulation 上，用來給予空間點集「形狀」的方法。 假設有空間上的點集 S，當 Alpha shape 的 alpha 值為無限大時，所產生的形狀為 S 的 convex hull，隨著 alpha 值的遞減，S 的形狀會慢慢的出現凹陷，直到設定適當的 alpha 值，S 的原始形狀也將出現，當 alpha 值為 0 時，所產生的形狀將會是原來的點集合。 如圖 3.2 所示，隨著 alpha 值的變動，所產出的形狀也不一樣。

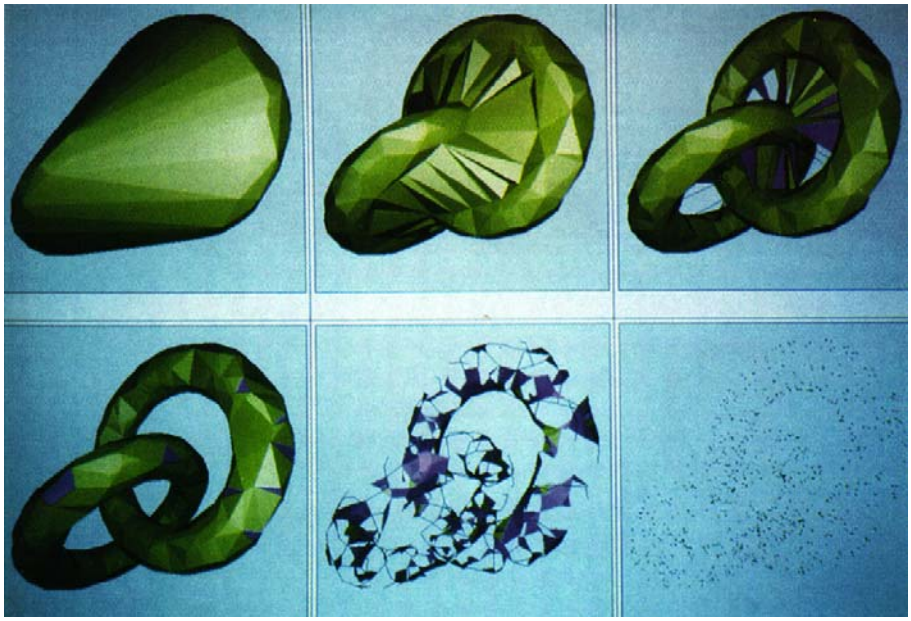

圖2.5　不同的alpha值將會有不一樣的形狀

(資料來源：H. Edelsbrunner and E. P. Mücke, “Three-Dimensional Alpha Shapes”, ACM Transactions on Graphics 1994)

三維的 Delaunay Triangulation D 是由一群分布於三維空間中的點集 S 建構出來的四面體，這些四面體的外接球內部不允許有其他屬於 S 的點。D 中存在著許

多的 simplex，在計算幾何的領域中，simplex 是三角形或四面體在任一維度下一種表示法，通常一個 n-simplex 表示一個 n 維度且為其 n+1 個頂點的凸包(convex hull)的多胞體(polytope)。例如：2-simplex 是三角形，3-simplex 是四面體，而 0-simplex 和 1-simplex 則分別是點和線段。

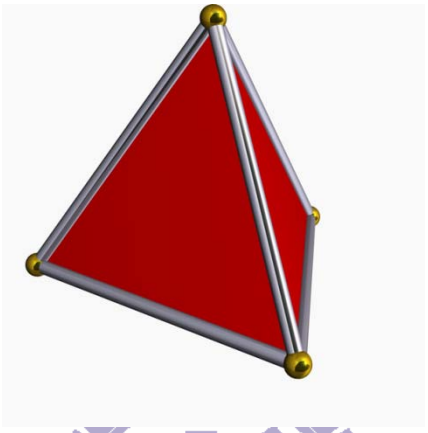

圖2.6　四面體(3-simplex)

(資料來源：Wikipedia,　http://en.wikipedia.org/wiki/Simplex)

在 Alpha Shapes 中，所有的幾何元件皆來自於 Delaunay Triangle 的 simplex，而這些 simplices 所構成的 α-complex 即為後來輸出的 Alpha shapes。 Simplicial complex 是一群 simplices 的集合，其必須符合下列兩個條件：

1. 任一個在 simplicial complex 中的 simplex 的幾何元件，仍然在此 complex。
2. 任兩個在 simplicial complex 中的 simplices 之交接處仍為此兩個 simplices 的幾何元件。

下圖(左)為一個合法的 simplicial 3-complex，圖(右)則因違反條件 2 而不為一個 simplicial complex。

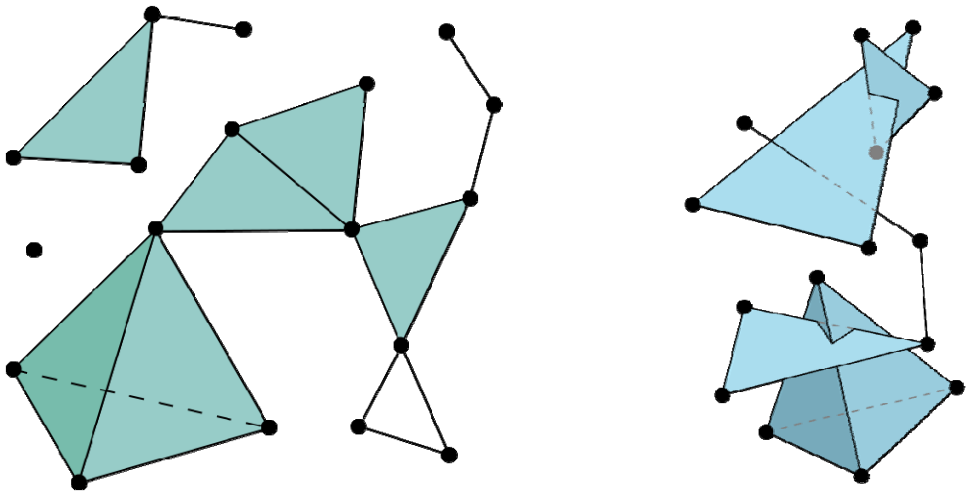

圖2.7　合法(左)與非法(右)的simplicial complex

右圖中，存在 simplices 交界處並非其幾何元件，故不為一個 simplicial complex

(資料來源：Wikipedia, http://en.wikipedia.org/wiki/Simplicial_complex)

Delaunay Triangle 中的任一 simplex 都擁有一區間，用來決定哪些 alpha 值會使得這些 simplex 成為 α-complex 的成員。建構三維 Alpha shapes 的步驟大致分為下面兩個步驟：

1. 利用 flip 演算法建構三維 Delaunay Triangulation

2. 計算 Simplex 的區間並將 Facet 分類

2.3.1. Flip Algorithm

Incremental-flip Algorithm 是建構 Delaunay Triangulation 常見的演算法，如下：

1. 將點集 S 依照某個方向排序，然後將排好的 S 裡面的點重新標籤使得{p_1, p_2,……, p_n}為排序過的序列
2. 用{p_1, p_2, p_3, p_4}初始 Delaunay Triangulation **D**
3. **for** i=5 **to** n **do** 加入點 p_i，並連接至 **D**，若不為一個 Delaunay，則持續 flip 直到變成 Delaunay 為止 **end for**

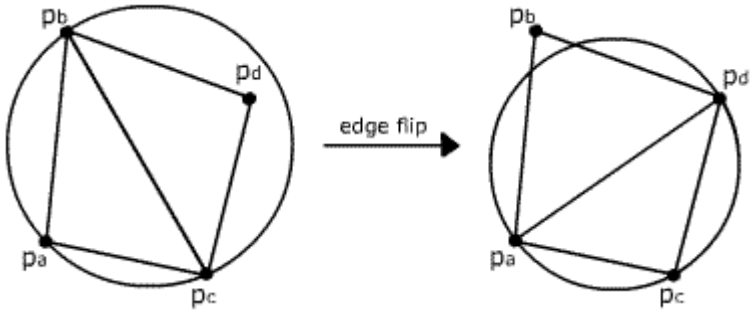

圖2.8 Delaunay Triangulation的flip

(資料來源：Constrained Delaunay Triangulation using Plane Subdivision, Maribor at al.)

如圖 3.4，左側 △(p_a, p_b ,p_c)的外接圓內側還包含一點 p_d，違反 Delaunay 之規定，故須將中間的邊做翻轉，由原本的 edge(p_b, p_c)轉成 edge(p_a, p_d)，翻轉過後△(p_a, p_b ,p_c)的外接圓內側則沒有任何其他的點。

2.3.2. Intervals and Face Classification

Alpha shapes 中的幾何元件被分為三類：singular、regular 和 interior。其關係是如下：

$$\begin{cases} interior & if\ \sigma_T \notin \partial\mathcal{S}_\alpha \\ regular & if\ \sigma_T \in \partial\mathcal{S}_\alpha，且屬於\mathcal{K}_\alpha 中更高維度之\ simplex\ 的幾何元件 \\ singular & if\ \sigma_T \in \partial\mathcal{S}_\alpha，且不屬於\mathcal{K}_\alpha\ 中更高維度之\ simplex\ 的幾何元件 \end{cases}$$

其中，σ_T表示 Delaunay Triangle 中，以所有點集 S 的子集 T 所構成的 simplex，$\partial\mathcal{S}_\alpha$為 Alpha shapes 之表面，$\mathcal{K}_\alpha$則是 α-complex。然而，在 Delaunay Triangle 中的 simplex 又可再分成：

$$\begin{cases} attached & if\ |T| = 2,3\ \text{且}\ b_T \cap \mathrm{S} \neq \emptyset \\ unattached & otherwise \end{cases}$$

其中，T 為構成σ_T的點集合，b_T為 T 的外接球面，S 則是所有點的集合。

接下來，alpha 值的區間與 simplex 的分類可以歸納為下表：

表2.1　simplex 區間值分類

		singular	regular	interior
四面體				$(\rho_T, \infty]$
邊 或 三角形	$\notin \partial$conv(S), unattached	$(\rho_T, \underline{\mu}_T)$	$(\underline{\mu}_T, \overline{\mu}_T)$	$(\overline{\mu}_T, \infty]$
	$\notin \partial$conv(S), attached		$(\underline{\mu}_T, \overline{\mu}_T)$	$(\overline{\mu}_T, \infty]$
	$\in \partial$conv(S), unattached	$(\rho_T, \underline{\mu}_T)$	$(\underline{\mu}_T, \infty]$	
	$\in \partial$conv(S), attached		$(\underline{\mu}_T, \infty]$	
頂點	$\notin \partial$conv(S)	$[0, \underline{\mu}_T)$	$(\underline{\mu}_T, \overline{\mu}_T)$	$(\overline{\mu}_T, \infty]$
	$\in \partial$conv(S)	$[0, \underline{\mu}_T)$	$(\underline{\mu}_T, \infty]$	

其中，∂conv(S)表示點集 S 的凸包之表面，ρ_T為 T 的外接球半徑，而

$\overline{\mu}_T$、$\underline{\mu}_T$ 則是 simplex 從 singular 轉成 regular，從 regular 轉成 interior 的分界值，其計算方式如下：

若 σ_T 為四面體，則 $\overline{\mu}_T = \underline{\mu}_T = \rho_T$，否則：

$$\begin{cases} \underline{\mu}_T = min\{\varrho_{T'} | \sigma_{T'} \in \mathrm{up}(\sigma_T), \mathrm{unattached}\} \\ \overline{\mu}_T = max\{\varrho_{T'} | \sigma_{T'} \in \mathrm{up}(\sigma_T)\} \end{cases}$$

其中，$\mathrm{up}(\sigma_T)$表示在 Delaunay Triangle 中，所有包含σ_T且$|T| \leq 3$的 simplices 之集合。換句話說，$\underline{\mu}_T$ 須從大自己一個維度且包含自己的 simplices 中，找外接球半徑最小者；$\overline{\mu}_T$ 則是從大自己一個維度且包含自己的 simplices 中，找外接球半徑最大者。

全部的 alpha 區間值算好且將幾何元件都分類後，儲存在合適的資料結構裡頭，當使用者選擇不一樣的 alpha 值及幾何元件種類時，將會依照適當的幾何元件輸出，這些幾何元件的集合即為 Alpha shapes。

第三章　效果

3.1 Alpha-Blending

在多重體積顯像系統中，同時會存在多個體積影像，倘若每個體積影像都呈現不透明的狀態，我們將很難去觀察它們分布的位置。我們採用 Alpha Blending 這種在 3D 影像處理中常見的技術，以下為其公式：

$$\begin{cases} out_A = src_A + dst_A(1 - src_A) \\ out_{RGB} = \left(src_{RGB} \times src_A + dst_{RGB} \times dst_A(1 - src_A)\right) \div out_A \\ out_A = 0 \rightarrow out_{RGB} = 0 \end{cases}$$

其中out_A，out_{RGB}，src_A，src_{RGB}，dst_A，dst_{RGB}分別為最終輸出影像、前景集背景影像的透明度(alpha)與色彩值，透明度 alpha 界於 0.0～1.0 之間，值越小表示越趨透明。 下列圖 4.1 為本系統實做 Alpha blending 之實驗效果：

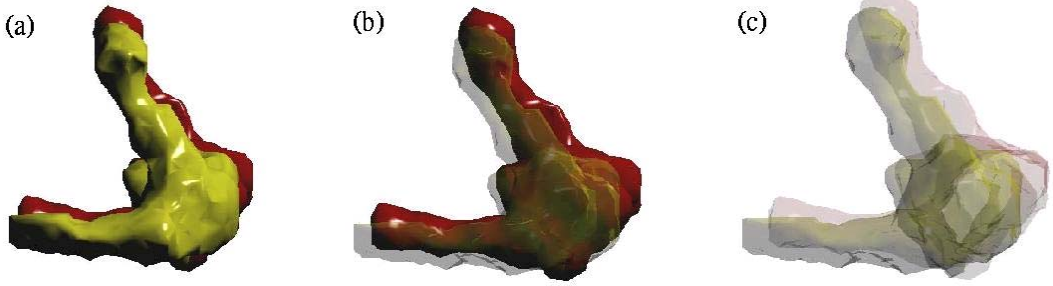

圖3.1　Alpha blending 效果比較

(a)src_A 、dst_A皆等於 1　(b) src_A小於 1，dst_A等於 1　(c) src_A 、dst_A皆小於 1

由圖 4.1 可發現，當前景 alpha 值皆為 1 時，後景會完全被前景擋住，而前景 alpha 值小於 1，後景 alpha 值等於 1 時，我們可以透視前景來看到後景，若前後景 alpha 值皆小於 1，則二者皆可被透視。

3.2 光照效果

為了讓幾何模型看起來更加趨近於真實物體，圖學中加入了許多模擬現實世界光照效果的模型，又以 Phong lighting model 最被廣為使用，以下為其公式：

$$I = k_a l_a + k_d l_d(\vec{l} \cdot \vec{n}) + k_s l_s(\vec{h} \cdot \vec{n})^n$$

其中 $\vec{n}$ 為物體表面向量，$\vec{l}$ 為光的方向，$\vec{h}$ 為光與視角夾角一半的向量，l_a, l_d, l_s 分別為光源的 ambient，diffuse，specular 之係數，k_a, k_d, k_s 分別為物體的 ambient，diffuse，specular 之係數，n 為 shininess 係數。

除了光照模型外，shading 的模式也對產生出來的幾何物件有非常大的影響，常見的有 Flat shading， Gouraud shading 以及 Phong shading 模型，以下為使用此三種 shading 模型所顯現的結果：

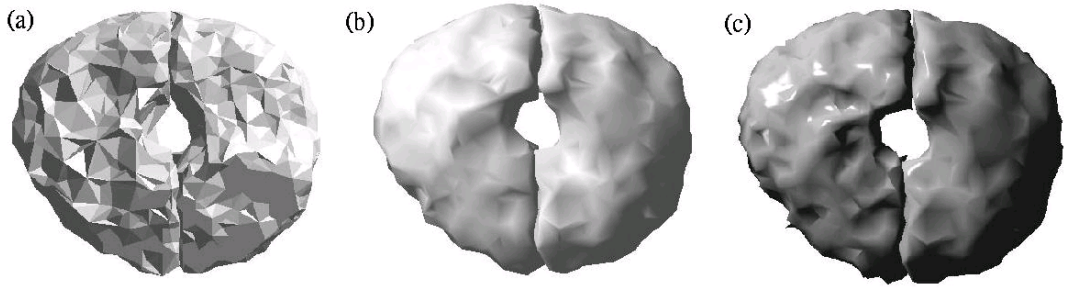

圖3.2　(a)Flat shading　(b)Gouraud shading　(c)Phong shading

由於 Flat shading 是將幾何物件中的每個元件分配一個法向量，故可明顯看出幾何元件的組成，而 Gouraud shading 則是算出每個頂點法向量後，根據頂點的顏色去內插出幾何元件的顏色，所以亮部區域會顯得太大，Phong shading 則是用頂點法向量去內插出幾何元件上每個點的法向量，使得最後呈現的色彩為物件上每個點的法向量與光線運算的結果，最為接近真實世界。

第四章　實作與成果

本系統為 VC++搭配 QT、CGAL 及 OpenGL 開發而成，主要分為果蠅腦神經元件(neuropil)與神經(neuron)兩部分來操作， 為了讓使用者能夠更清楚的分辨各神經元件，本系統提供對各神經元件改變顏色與透明度之功能，對於載入後，暫時不需要使用到的神經元件，也提供隱藏功能，由於表面立體資料顯影欲做透明效果必須將幾何元件之所有多邊形依距離視點之遠近來排序，在對幾何元件做旋轉時將會非常耗時，因此，本系統在對物件做旋轉時暫停排序功能，等轉到使用者有興趣的角度時，才會對所有多邊形做排序。神經的部分，本系統同樣也能對各神經群、神經組及單一條神經分別指派其特定的顏色，以利於使用者分辨，另外還有 Connect 功能，用來顯現神經與其連接的神經元件；Search 功能，用來顯示使用者當前所選取的神經；以及用線條來表示，以加快操作速度。

4.1 系統使用者介面

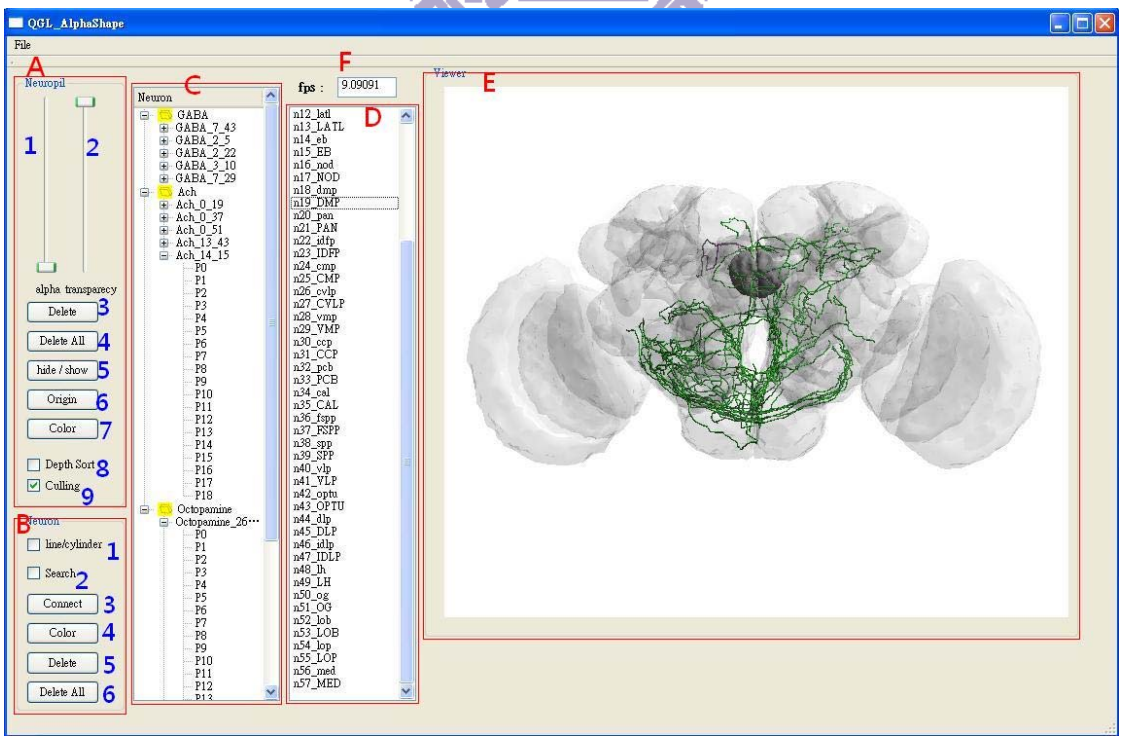

圖4.1　　系統使用者介面

本系統的介面大致上可分為 A ~ F 六個區域 (如圖 4.1 所示)：

A 區域：神經元件的操作，分別為：

1. Alpha shapes 的 alpha 值
2. 透明度
3. 刪除被選取的神經元件
4. 刪除全部的神經元件
5. 顯示/隱藏被選去的神經元件
6. 回到一開始的視點中心
7. 對被選取的神經元件配置顏色
8. 對神經元件的幾何元件作深度排序
9. 將背面的幾何元件去除

B 區域：神經的操作，分別為：

1. 線狀神經/管狀神經的切換
2. 神經搜尋的開關
3. 對被選取的神經做連結功能
4. 對被選取的神經配置顏色
5. 刪除被選取的神經
6. 刪除全部的神經

C 區域：神經列表

D 區域：神經元件列表

E 區域：模型顯影區

F 區域：每秒影像的張數

4.2 初始狀態

圖4.2　　果蠅腦外殼

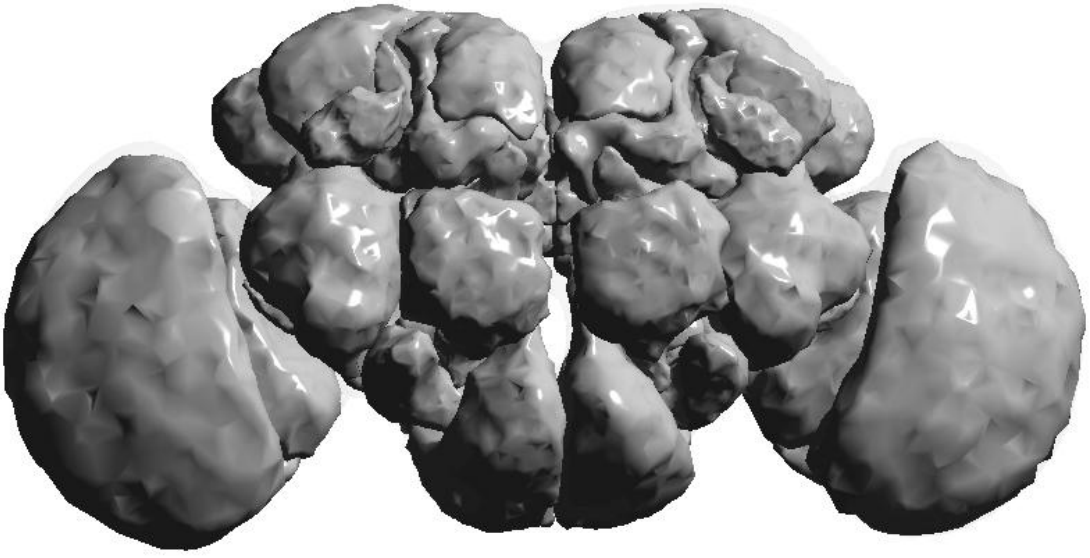

圖4.3　　果蠅腦神經元件(正面)

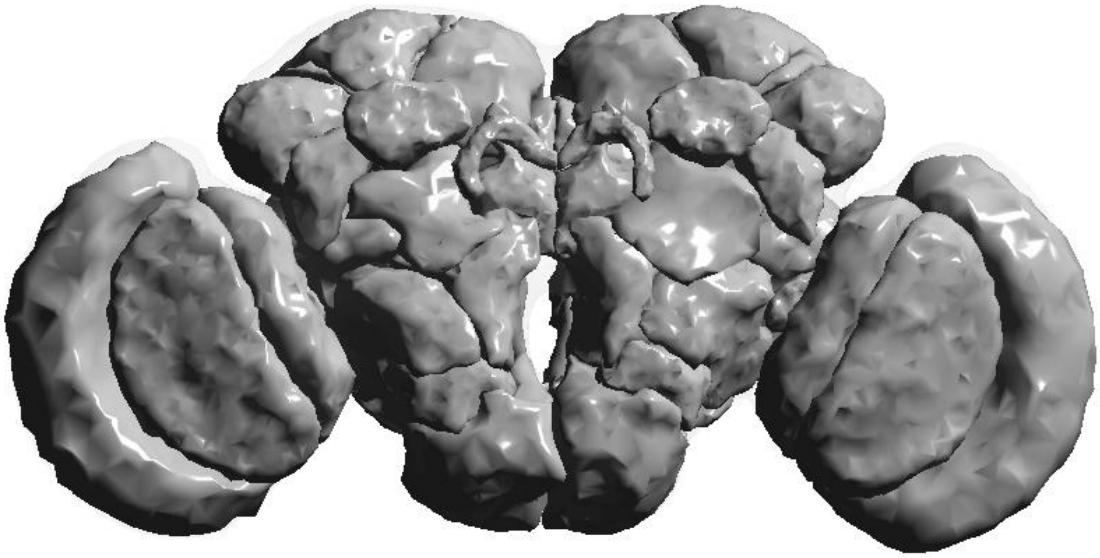

圖4.4　果蠅腦神經元件(背面)

4.3 腦神經元件色彩客製

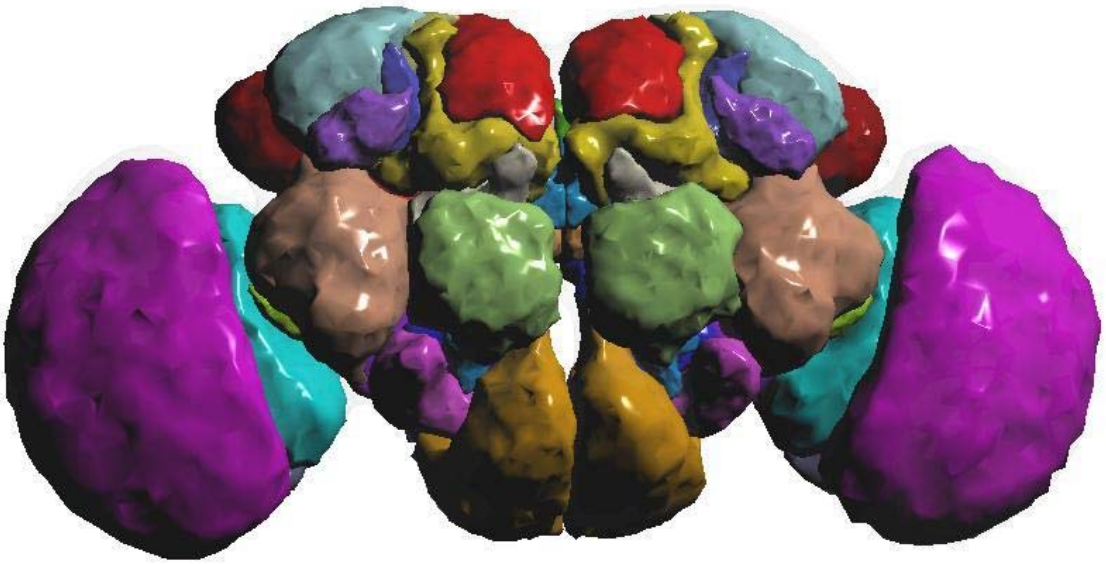

圖4.5　腦神經元件色彩客製(正面)

將左右相對位置上的神經元件配置同一種顏色

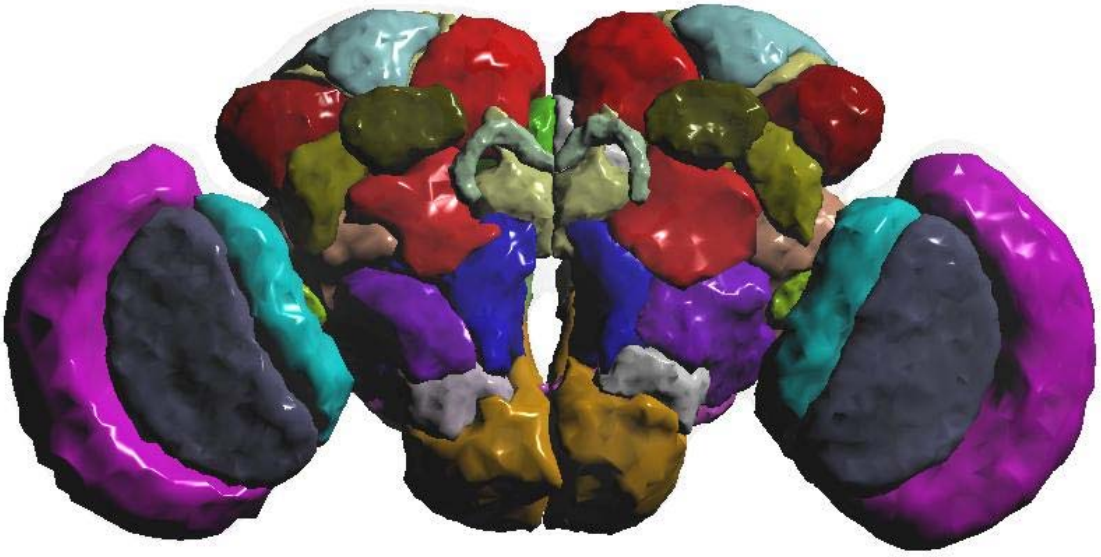

圖4.6　腦神經元件色彩客製(背面)

4.4 腦神經元件透明效果

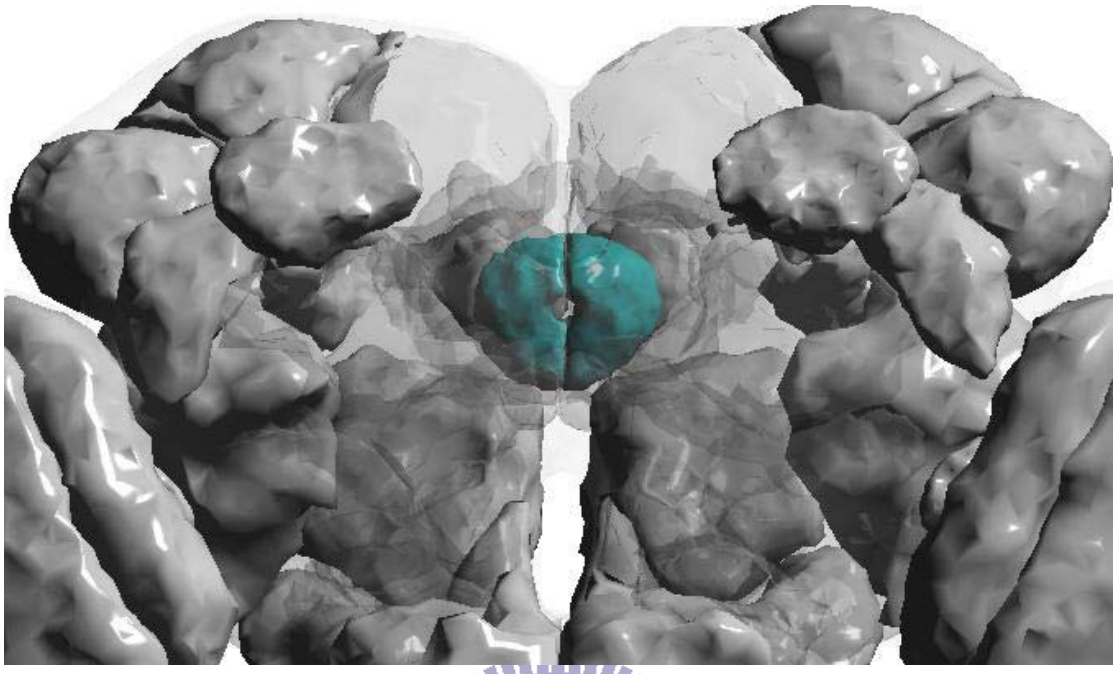

圖4.7　神經元件的透明化 I

將神經元件 EB(藍色)前的神經元件之 alpha 值調低，EB 因此可以顯露出來

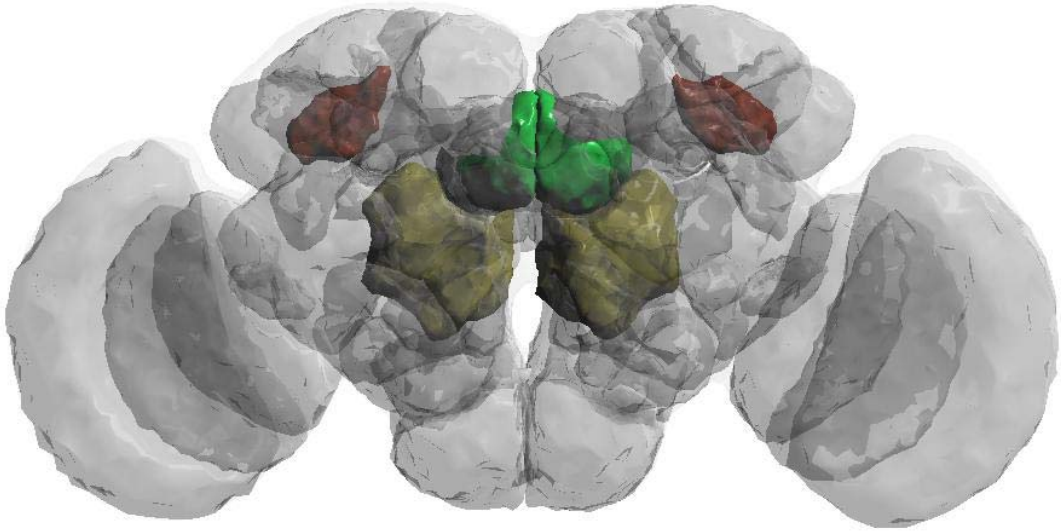

圖4.8　神經元件透明化 II

紅色，綠色，黃色分別為神經元件 OPTU，FB，AL

4.5 管狀神經 V.S. 線狀神經

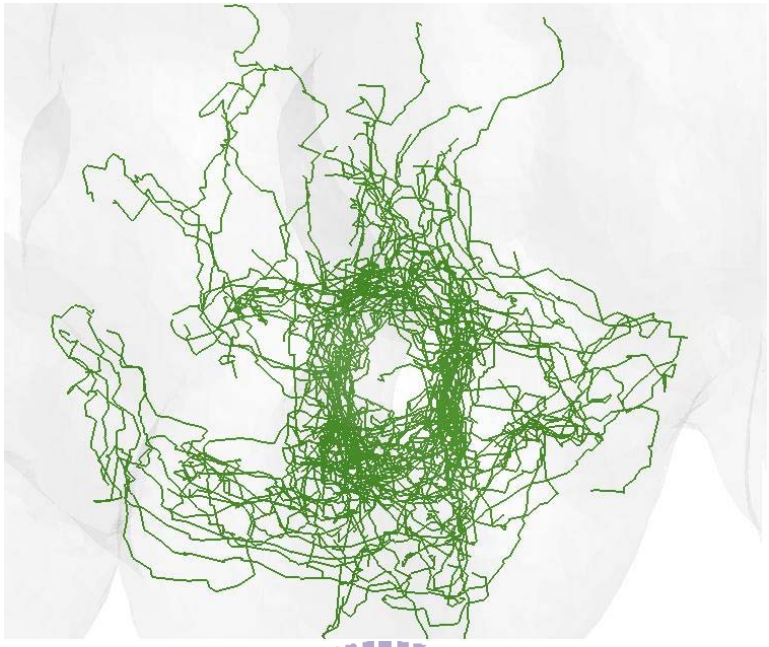

圖4.9　線狀神經

線狀神經繪製的速度較快，但因為沒有表面法向量資訊，以致看不出其前後關係

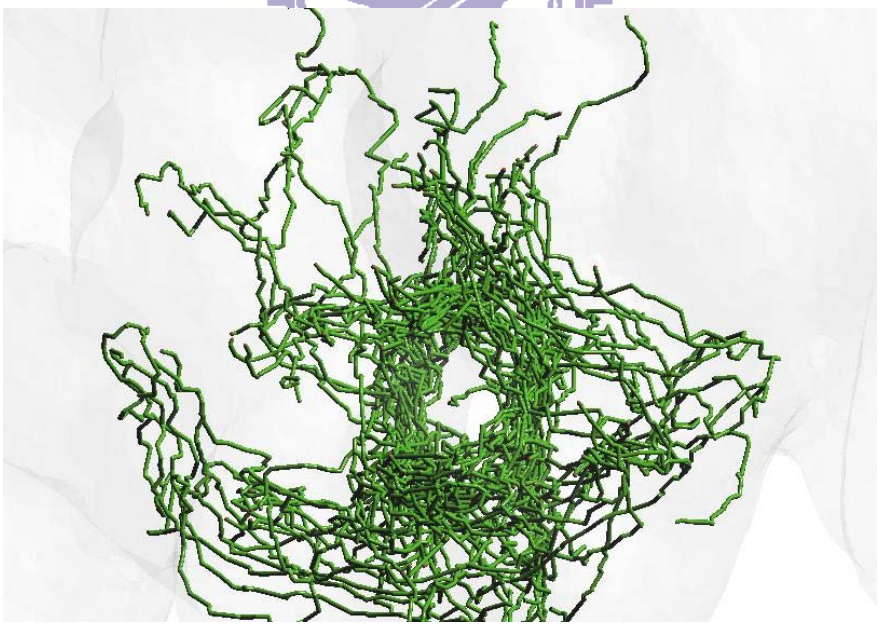

圖4.10　管狀神經

管狀神經繪製的速度比線狀神經慢許多，但可明顯的看出其前後關係

4.6 神經的色彩客製

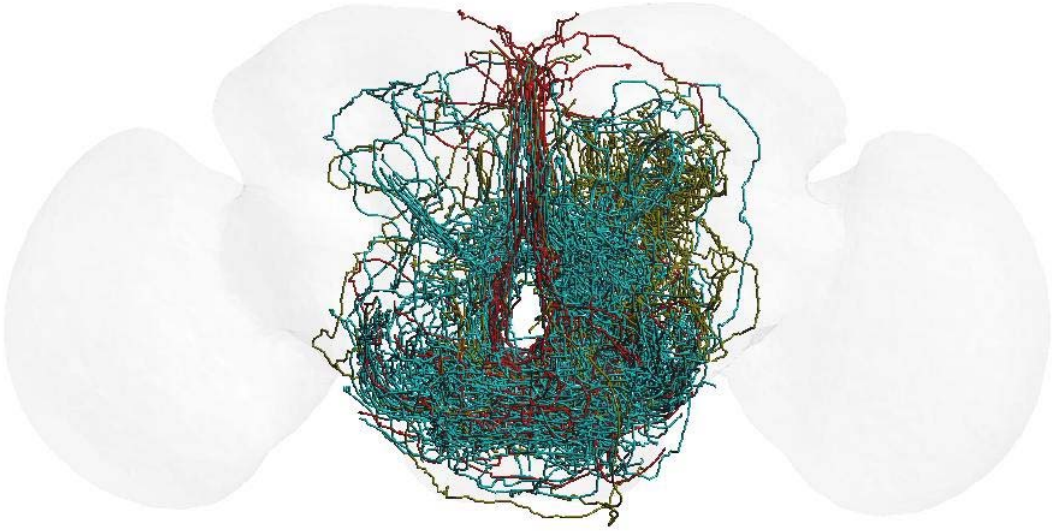

圖4.11　對神經群分別配置一種顏色

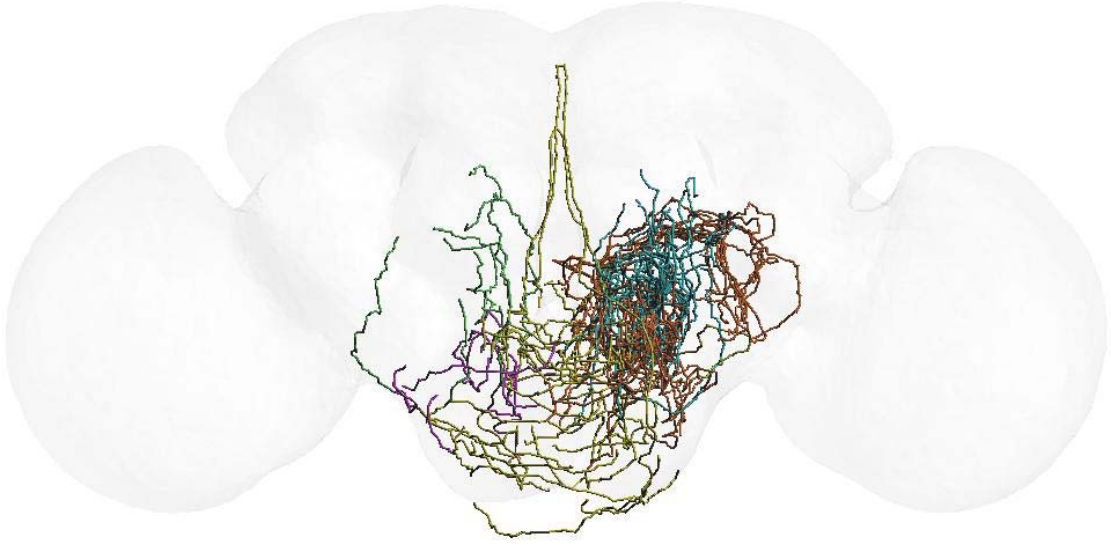

圖4.12　對神經組分別配置一種顏色

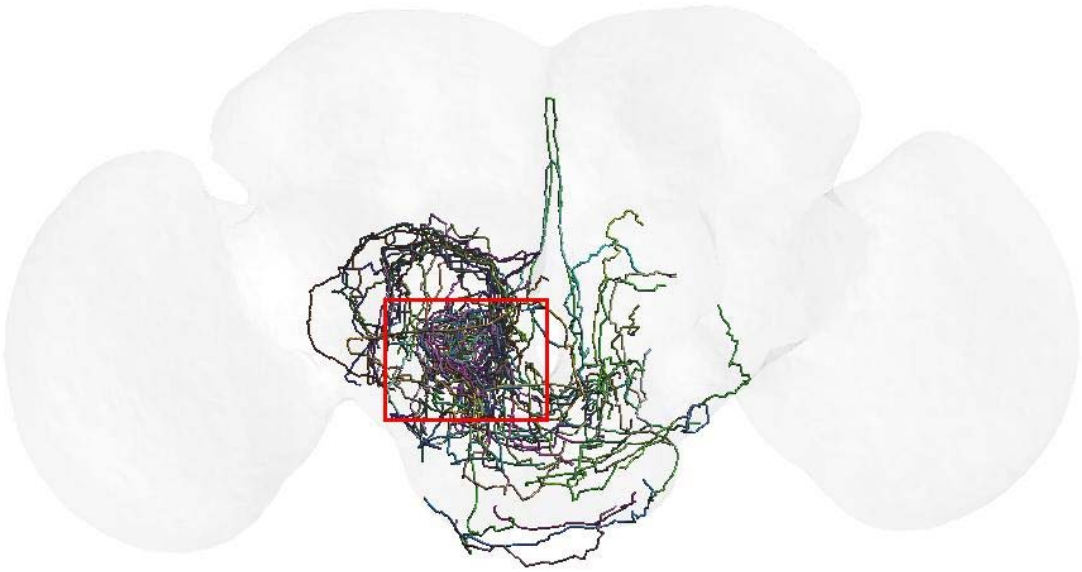

圖4.13　對每條神經分別配置一種顏色

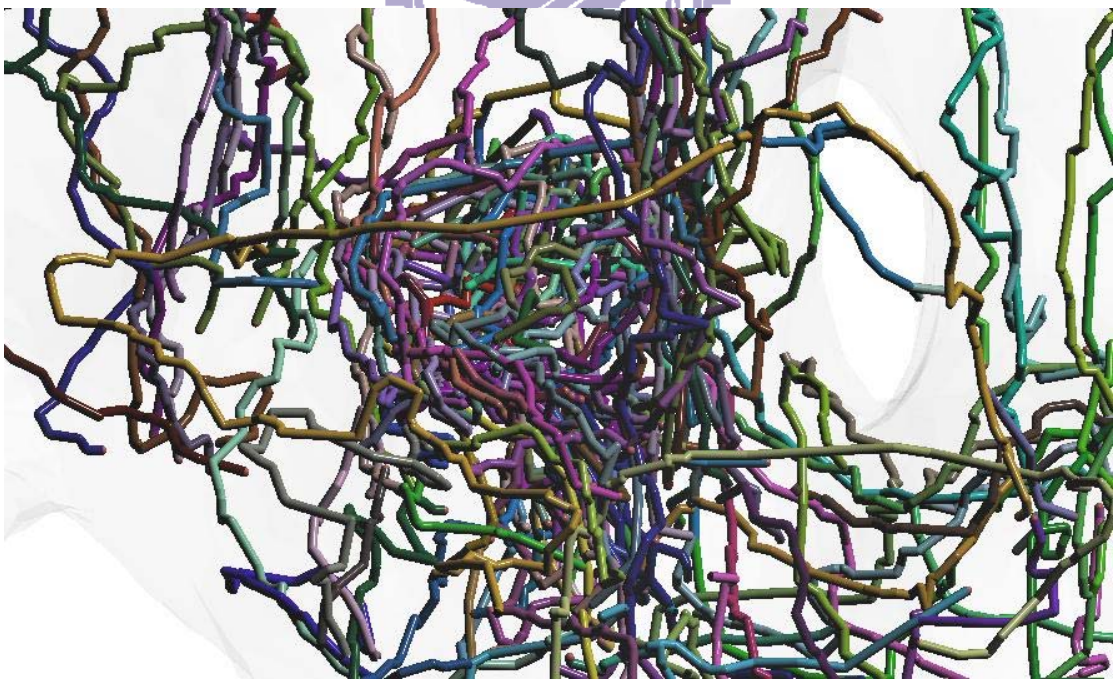

圖4.14　圖 4.13 中紅色方框區域放大圖

4.7 神經群與神經元件的連結

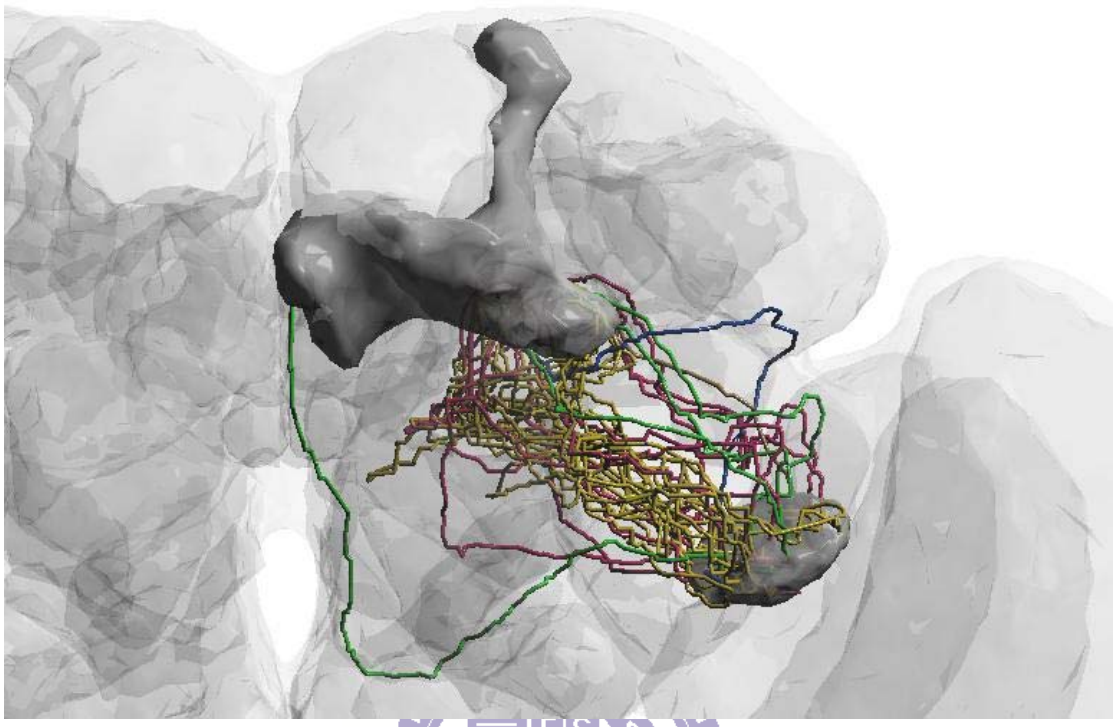

圖4.15　四個神經群與其所連結之神經元件

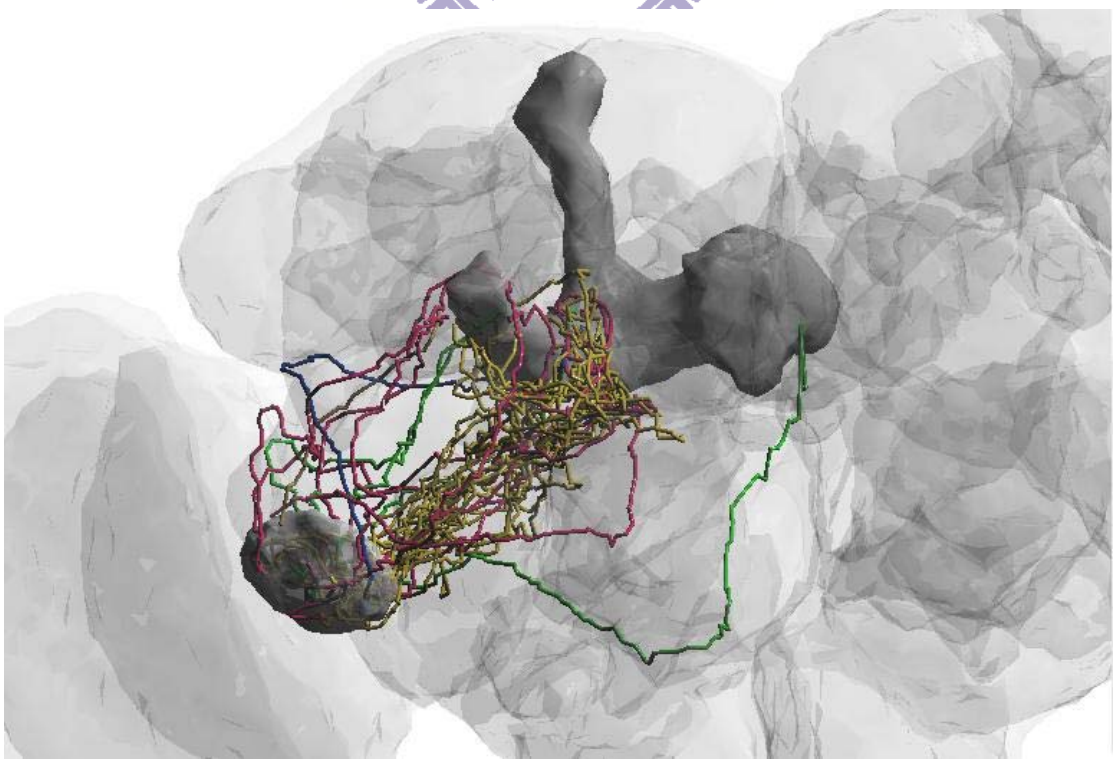

圖4.16　圖 4.15 的另一角度

4.8 神經組與神經元件的連結

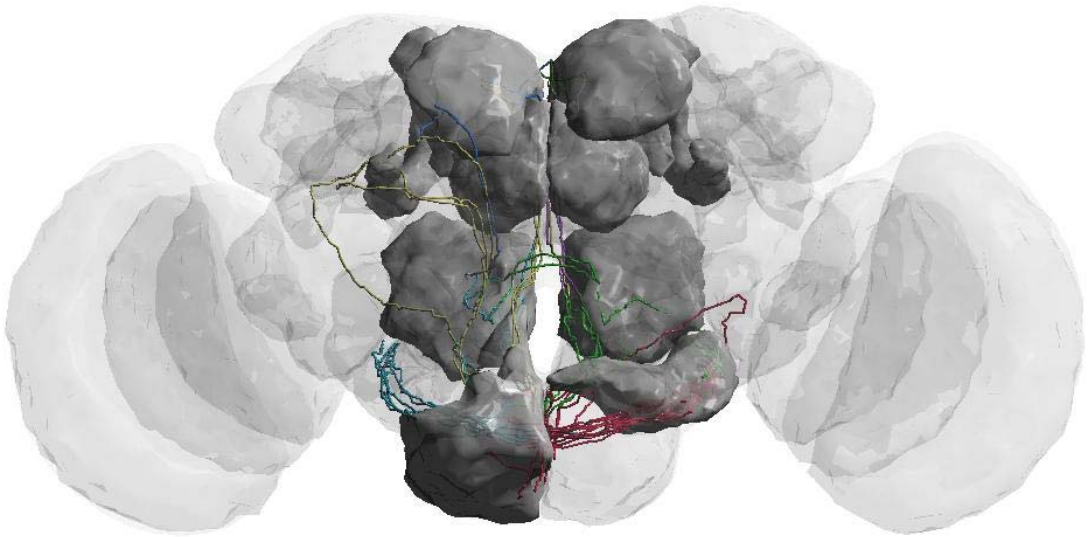

圖4.17　神經組連結神經元件 I

同一個神經群中的九個神經組所連結之神經元件，不同顏色表示不同神經組

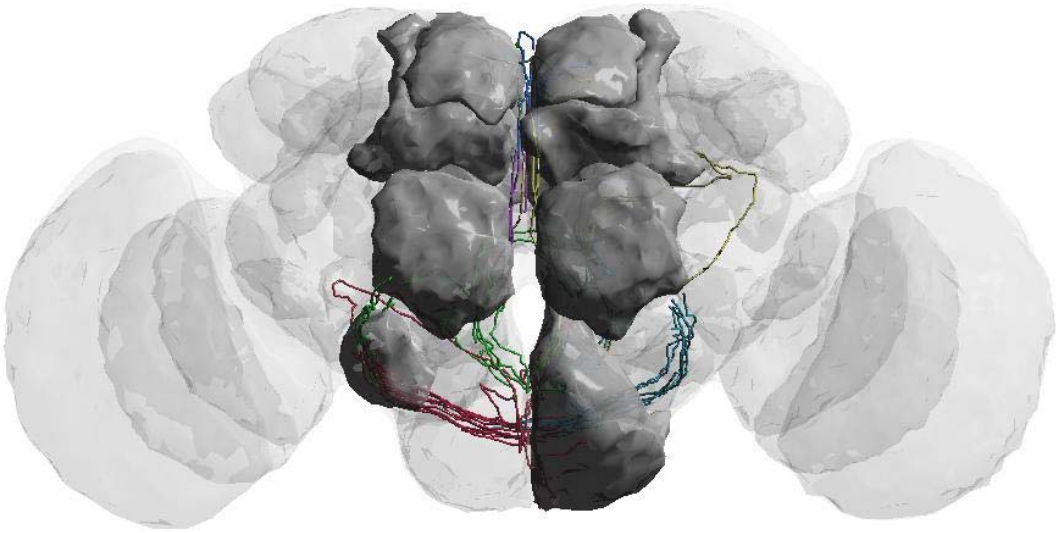

圖4.18　神經組連結神經元件 II (圖 4.17 背面)

4.9 單一神經與神經元件的連結

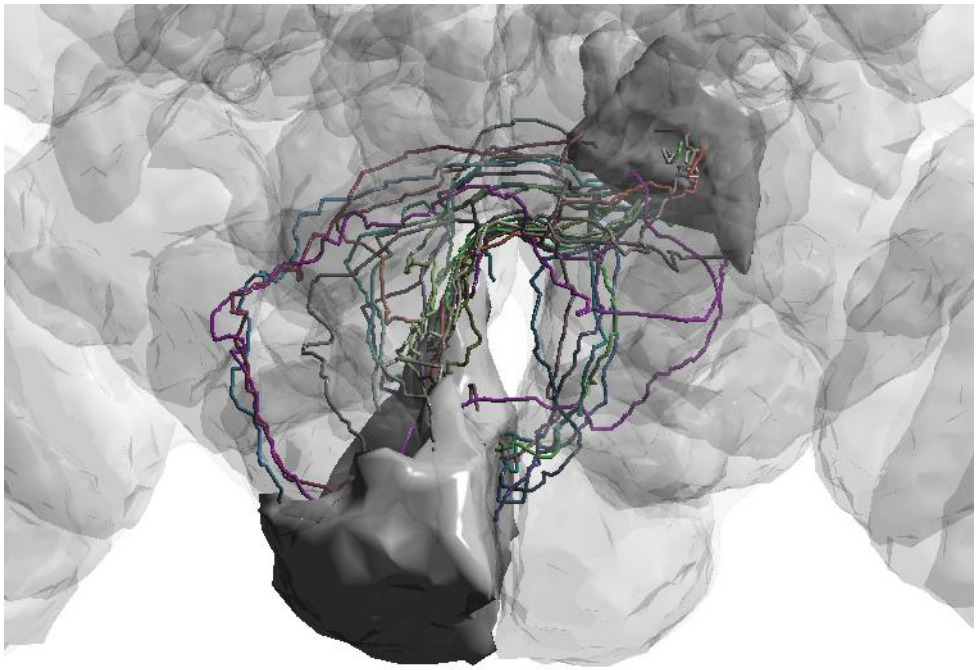

圖4.19　單一神經連結神經元件 I

同一神經組中二十八條神經連結至神經元件，每個顏色為個別的一條神經

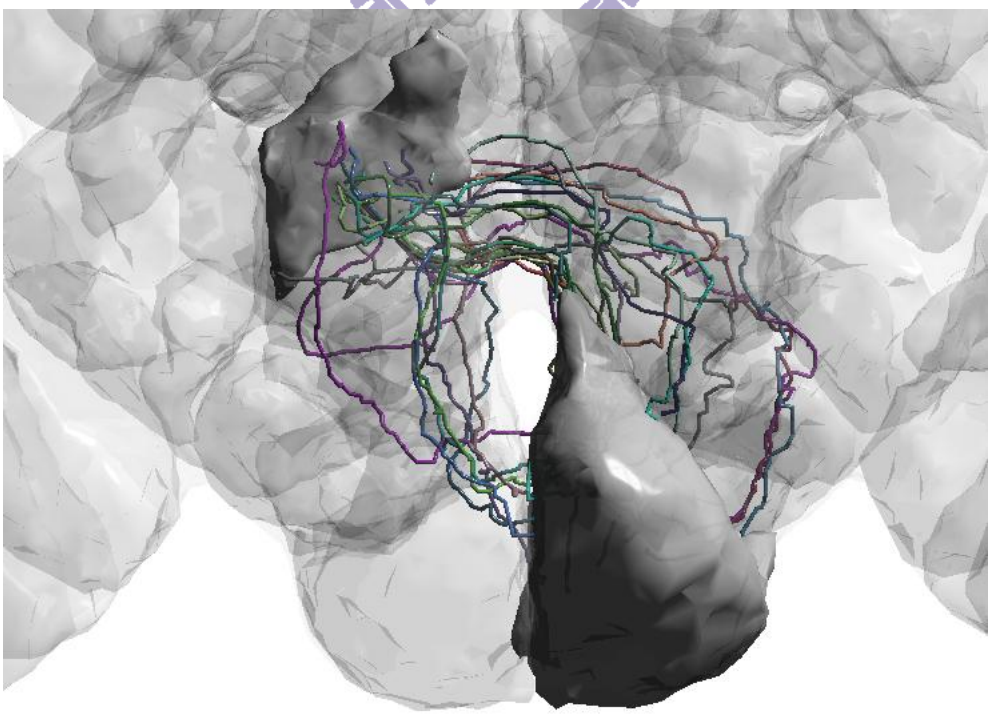

圖4.20　單一神經連結神經元件 II (圖 4.19 背面)

4.10 神經搜尋

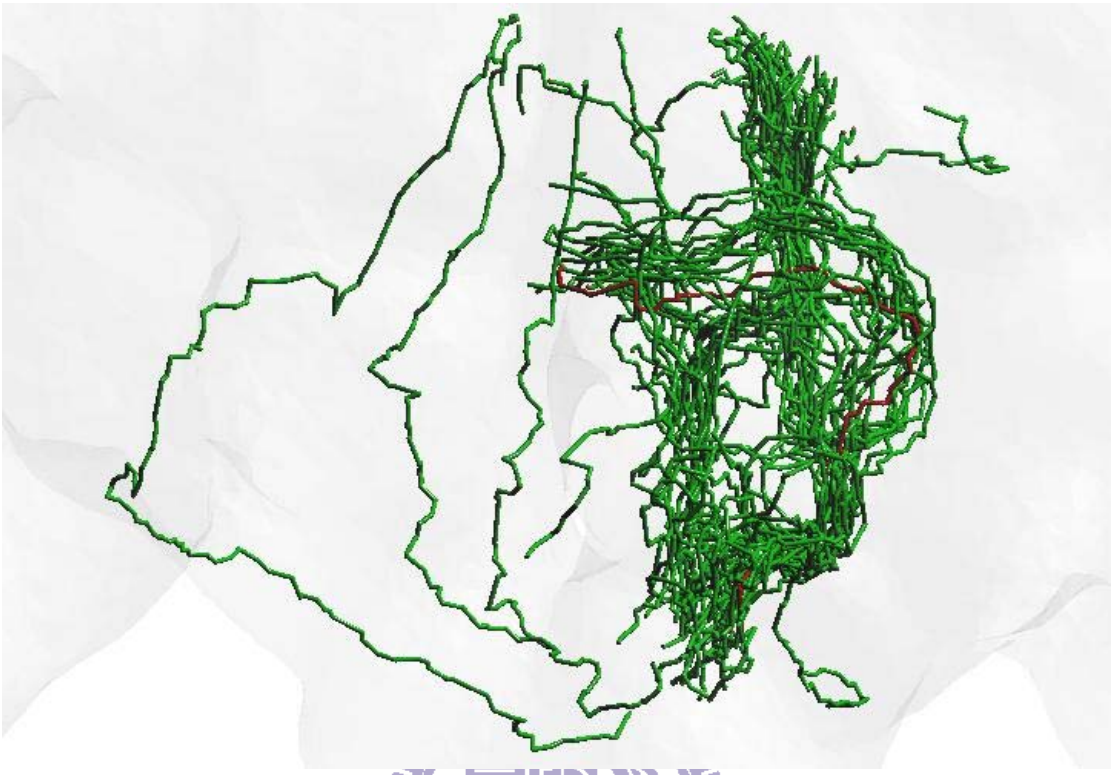

圖4.21 神經搜尋

在八十條神經中搜尋某一條神經，當下指定的神經會呈現紅色，其餘的皆為綠色

4.11 系統效能

表面立體資料顯影的效能與多邊形的數目有很直接的關係，多邊形的數目越少，繪圖的速度越快，反之，則越慢，但當多邊形數目越少時，輸出的模型將會變得越粗糙。 下列表格為本系統在 INTEL E8400 Core2Duo，NVIDIA 9800GTX，3.25G RAM，WIN XP 32-bit 環境下實測之結果：

表4.1 系統效能表

	無排序	排序
腦外殼 (18000△)	640 fps	35 fps
58 個神經元件(89806△)	22.5 fps	5.25 fps
腦外殼+58 個神經元件 (107806△)	22 fps	4.4 fps

第五章　結論與未來展望

本系統旨在讓使用者能夠清楚的觀察神經及其所連接之神經元件的相對位置關係，為此，系統內實作了多個立體資料顯影常用的技術，例如 alpha blending 及光照明暗的效果，另外，為了能夠在一大群神經中找到想要觀察的神經，系統也添加了搜尋(Search)的功能，而想要知道神經連接至哪些神經元件，只要將想觀察的神經選取，並使用連結(Connect)功能即可。

由於本系統是建構於表面立體資料顯影技術，所以在使用 alpha blending 技術的同時須將所有的多變形依照距離視點的遠近來排序，但如此一來將會使得系統效能嚴重落後，因此在多邊形的深度排序上，要如何才能更有效率是未來的研究方向之一。 此外，目前電腦 CPU 已往多核心發展，一個程序同時在多個不同的 CPU 核心上處理稱之平行化處理。平行化處理暨能夠達到物盡其用的理念，又能提高不少的系統效能，無疑地是未來的一大趨勢，如何把系統中比較浪費時間的演算法轉換成能夠平行處理的演算法是一個很重要的方向。

現今社會在平板電腦及智慧型手機等攜帶式的行動電子設備上瀏覽視訊或圖片等影像已是一種潮流，這種將大量運算放在遙遠的伺服器上，把運算完的結果傳至手持裝置的架構稱為 client/server，對於此，可以將本系統切開成兩部分，使用者介面放在 client 端，運算的部分則放在 server 上，最後輸出的影像再透過網路傳至遠方的手持裝置上，實現手持裝置的立體資料顯影系統。

參考文獻

[1] Bui-Tuong, Phong, “Illumination for Computer-Generated Pictures”, Communications of the ACM. Vol. 18, no. 6, June, 1975, pp.311-317.

[2] William E. Lorensen, Harvey E. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, Proc. ACM SIGGRAPH Computer Graphics, Vol. 21 Issue 4, July 1987.

[3] Herbert Edelsbrunner, Ernst P. Mücke, “Three Dimensional Alpha Shapes”, ACM Trans. Graphics, Vol. 13, no. 1, Jan. 1994, page 43-72.

[4] Gregory M. Nielson, Bernd Hamann, “The asymptotic decider: resolving the ambiguity in marching cubes”, Proc. IEEE Visualization Conf. (VIS 91), 1991.

[5] R. J. Rost, OpenGL Shading Landuage, 3rd edition, Addison Wesley, July 2009.

[6] E. Angel, Interactive Computer Graphics, 5th edition, Addison Wesley, 2009