國立臺灣大學管理學院資訊管理學研究所

碩士論文

Department of Information Management
College of Management
National Taiwan University
Master Thesis

風格轉換及網格變形之三維內容製作

Style Transformation and Mesh Deformation for 3D Content Creation

鄭京恆

Ching-Heng Cheng

指導教授：陳炳宇博士

Advisor: Bing-Yu Chen, Ph.D.

99

中華民國 99 年 7 月

July, 2010

國立臺灣大學資訊管理學系
碩士論文
風格轉換及網格變形之三維內容製作
鄭京恆 撰
99 7

致謝

這篇論文能夠順利的完成，最感謝指導教授陳炳宇老師的幫助，謝謝在研究所的這段時間裡，老師細心的指點及包容。老師的教導讓我能夠進入一直希望學習的電腦圖學領域，老師對研究的認真與熱情激勵我更努力，老師對我的包容與提醒總是讓我感動，謝謝老師的照顧，讓我研究所的日子過的充實且有意義。

謝謝阿山總是幫我解答研究上的問題，在我苦悶的時候幫我打氣。謝謝貼心的裕美，每次都在我需要幫忙時義不容辭的伸出援手。謝謝研究所的學長鎧尹、jonash、DD、士強、holyo，一起奮鬥看了好幾次日出，最棒的聊天好友阿良，學弟妹大猩猩、克遠、郁婷、家榮、奕超，大家都給了我許多幫助，謝謝你們的付出，讓我能夠度過一個又一個的難關，能認識你們真是太棒了。

謝謝我的家人，在這段日子裡總是給予我無條件的支持，家人一直是我最大的動力，謝謝父母親的栽培與一路上的支持，謝謝總是非常照顧我的姐姐，謝謝總是陪伴我的 Jennifer，有你們的支持與鼓勵，我才能放心的往前大步邁進。

最後謝謝通訊與多媒體實驗的所有老師同學，在這個實驗室裡的日子真的非常開心，大家一起努力、一起玩樂、互相幫助、互相打氣，在這段時間裡不止學到了專業知識，也拓展了眼界，更讓我認識了許多好友，我絕對不會忘了這裡給我的一切，非常開心能成為這裡的一份子，謝謝你們讓我的研究所生活如此精采。

民國 99 年 7 月

鄭京恆 謹誌

摘要

三維模型(3D model)的製作以及其動作的生成，一直是電腦圖學中重要的研究，主要原因是良好的三維模型，在製作上相當困難以及耗費時間，在應用上總是帶來許多負擔。本論文中提出一套新的系統，讓使用者能藉由一個基本模型(base model)，利用此系統快速地製作出使用者指定之風格的良好三維模型，在有需要很多同質性、類似的三維模型時，能大大的縮短生產時間。

首先系統需要得到一個基本模型，以及使用者給定欲變形之形狀，並在這兩樣資訊上給定特徵點對應(features mapping)的限制，使系統得知重要的輪廓及變形的形狀。接著藉由風格轉換(style transformation)的技術，並配合模型變形(mesh deformation)的演算法，將基本三維模型轉換成符合使用者需求的新三維模型。

此系統在相似性高、重複性高的三維模型上效果顯著，例如需要數百隻的魚群、顛峰時段馬路的車輛、或是一支軍隊，利用此系統，則使用者只需要產生一個基本的三維模型，便可快速大量生產其類似卻具有不同風格之新三維模型，對於電腦圖學領域上的有相當大的助益。

關鍵字：特徵對應、風格轉換、模型變形。

Abstract

The generation of 3D models and their animations has been an important issue in computer graphics research. The main reason is the difficulty to create a exquisite 3D model. And in the process of creation, it usually costs very much and wastes long time. This paper will propose a new system that enables users to start from a base model with animation data, rapidly manufacture a well-designed 3D model. This is really effective and time-saving, especially when requiring many similar 3D models.

In the beginning, the system requires a basic model and also a shape for deformation. Based on the information mentioned above, users should provide constraints of features mapping, in order to inform the system of the main contours and the shape of deformation. According to the style transformation technique, accompanies with the algorithm of mesh deformation, the system can transfer a basic 3D model to a new 3D model as users wish. The system is even more effective when it comes to highly resemble and repeating models and it is really helpful for the computer graphics area.

Key words: Features mapping, style transformation, mesh deformation.

目錄

表次

圖次

Chapter 1

簡介

1.1 研究動機及背景

在近年的電腦圖形使用上，在3D動畫、商業廣告、電腦遊戲、視覺效果等，都有了非常顯著的增加，電腦圖學與生活已經有了密不可分的關係。藉著這樣大量的使用以及研究，電腦圖學中的各個領域都有了長足的進步，在現在的電影、電腦動畫、電腦遊戲裡的畫面即可明顯感受到，無論是三為模型的精細度、畫面特效的華麗，都跟幾年前的產品有著很大的差異。然而在三維模型的製作以及動作上，卻仍然是最耗費時間與成本的部分，因為這部分的工作必須由大量的工作人員利用手動的方式去製作模型，想要產生精緻的畫面，精細複雜的三維模型是必須的，也因此需要花費大量的成本。

在電腦圖學的領域中，三維模型是相當重要的研究，而在三維電腦動畫、三維電腦遊戲中，更是廣泛的使用。三維模型的基本元素包含了：三

維空間中的頂點座標、頂點向量、構成面、以及貼圖座標。越是精細、精美的三維模型，一定會擁有更多的頂點及面，才能讓生成出的影像更漂亮，但是這樣龐大且細緻的三維模型資料，製作起來難度也增加了許多，因此也產生了許多的問題及研究。

目前在三維模型製作上，已有許多開發工具來提供使用者使用，例如 Maya、3D Studio Max 等，使用者可以藉由這些軟體的輔助，製作三維模型的外觀、貼圖、及動作等。當然在各個公司裡使用的開發工具不盡相同，不過基本上的問題還是一樣的，正如這些製作動畫及電影遊戲的公司提供的資訊所揭露的，在製作過程中，花費在這些模型製作上的成本和時間是非常巨大的。而主要原因就是這些模型需要專業美工人員手動製作，每一個模型所需要的頂點、骨架、貼圖、動作，製作這些的步驟都非常耗時，也因此讓我們產生尋找能夠縮短製作三維模型時間的想法。

然而，要設計和製作一個全新的三維模型，基本上還是得靠使用者親自動手慢慢產生、調整。雖然在快速生成三維模型這個議題上，已經有非常多的研究，來開發系統增進速度或是使用者的便利性，但是若要能夠產生品質夠好的模型，還是只能靠製作者慢慢產生。因此，若使用者需要產生大量具備同樣性質的模型，是否能夠利用現有的技術做到調整、風格化的研究技術，來自動產生性質接近的模型提供使用者使用呢？例如在許多動畫或是遊戲中，需要軍隊、魚群、車潮，這類的情況，我們希望能夠找到一個方法來快速產生風格化的同質性三維模型，來節省製作三維模型的時間及困難度。

圖 1.1：同性質且數量龐大之三維模型。Traffic by Marlin studios。

1.2 研究目標

本文所提出的系統及作法，希望能解決的問題，就是減少製作所需數量龐大但特徵類似的三維模型。由前面介紹可以得知，無論是業界或學術界，在產生這些三維資料時，都必須耗費龐大的資源，這些資源都來自於三維資料產生的不易。我們希望能產生一套系統，讓製作人員或是任何使用者，他只需要一個基本的模型，就能快速產生一系列同特徵的，但風格不同的三維模型，大幅減少產生三維模型的時間及困難度，如此一來就能利用現有資源來快速達到生產三維模型的成果。

首先，系統將先讀入一個基本模型，而另外的一項資訊，則是讀入一張影像或是使用者自行繪畫的圖形，從影像或使用者給定的形狀和特徵點

資訊，利用風格轉換的技術，配合上三維與二維特徵點的比對，來快速產生不同形狀、不同風格、且具備該有之特徵的三維模型，來供使用者使用。

所謂三維與二維特徵點的比對，就是能夠讓使用者在三維模型上定出特徵點，而二維影像上可以定出其對應的特徵點，利用這兩個的對應關係以及空間中的位置，來將三維模型變形以符合其二維形狀。

利用此系統，使用者能夠大幅減少產生三維模型的時間，尤其是在重複性高、物體多的情況下，如熱帶軍隊、魚群、車潮、眾多建築物等，效果將更為突出。

1.3 論文架構

接下來的文章中，將會詳細介紹此系統，並且對於其演算法及功用有更詳細的介紹。首先會先介紹國內外在這領域上的相關研究，以及系統中使用的部份功能所參考之論文，這些皆會在第二章做明確的介紹。第三章開始則會將整個系統運作的流程做一步一步的介紹，一開始我們會介紹整個系統的流程圖，從輸入到輸出，經過了哪些關鍵的處理運作，接下來則會把系統流程中的每個步驟解釋，包含其中使用的演算法、運作的效能、以及達到的效果。最後，則是討論系統的功能，實際上的效用，以及未來的發展性。

Chapter 2

相關工作

本系統希望能夠利用二維影像的資訊，配合使用者定義其風格，來產生具有原始模型特徵且具備使用者風格之模型。因此主要研究議題包含兩個部分：一是風格轉換，另一則是模型形變，以下將介紹實作系統中所參考的研究。

2.1 風格轉換

風格轉換或是風格化都是電腦圖學中一個重要的應用，在二維影像上產生的效果，可以將影像或影片的性質轉換成不同的風格，例如原本是油畫，我們轉換成水墨畫，亦或是原本是擬真，我們轉換成卡通化。另外，在二維影像上，可以使目標物體的輪廓產生形變，同樣能達到風格化的目的。

(a) (b)

圖 2.1：(a) 漫畫風格化處理的影像[5]，(b) 二維影像之形變[6]。

而在三維模型上的應用，則偏向於模型外觀上的變化，也就是形狀變形。在[1]中，呈現了一個典型的風格轉換的動作，能夠將一個原本是正常形狀的魚的三維模型，經由風格轉換，產生出橢圓或是圓形的魚的模型。而這個變形的方法主要是考慮模型的輪廓，將三維模型投影到二維上，將這些輪廓邊與我們希望他變成的形狀來做比對，將這些輪廓邊盡量符合我們給的形狀，再逐步調整其三維模型上的頂點位置，以達到形狀變化以及風格轉換的效果。

圖 2.2：風格化之三維模型[1]。

由於在三維模型上的風格化，是定義在輪廓的形變，因此我們將風格化三維模型這個行為視為 shape modeling 的一部分，下面介紹了關於 shape modeling 的數種研究。

Example based shape modeling：一般來說，具有大量頂點的三維模型，都是由經驗豐富的美工人員或是三維掃描建模機器來製作，普通使用者是無法藉由簡單的操作來產生這種模型，因此在建模這個方面的研究就朝向了讓使用者能夠對於已經製作完成的龐大三維模型做操作，利用改變外形或是組合不同模型來生成各種新模型。較直觀的做法像 shape interpolation [17, 18]，還有[19]中藉由組合不同模型中的不同部分，來構成新模型，另外還有將一個模型的細節部分，經過形變之後接到另一個模型上，產生出更多樣化的模型，如[20, 21]中，把細節保留並轉移到了另一個模型上，形成更多種動作及外觀。

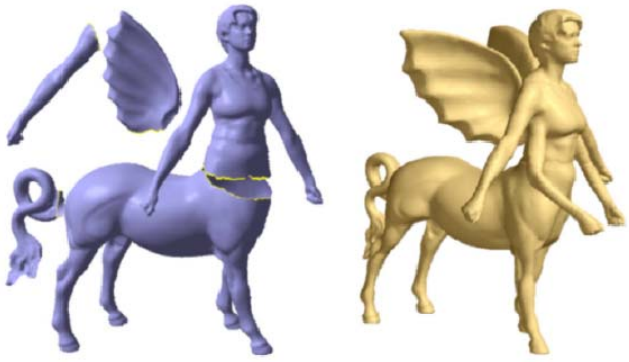


圖 2.3：藉由形變後的模型再加以組合成新模型[20]。

除了由三維模型與三維模型的組合建模方式之外，同樣的也有從二維上取得使用者給予資訊，利用這些資訊來編輯三維模型，如提供二維影像產生出輪廓。在[14]中，利用卡通中物體的二維輪廓資訊，來讓三維模型的同一部位形變，使得三維模型的動作跟二維影像符合。

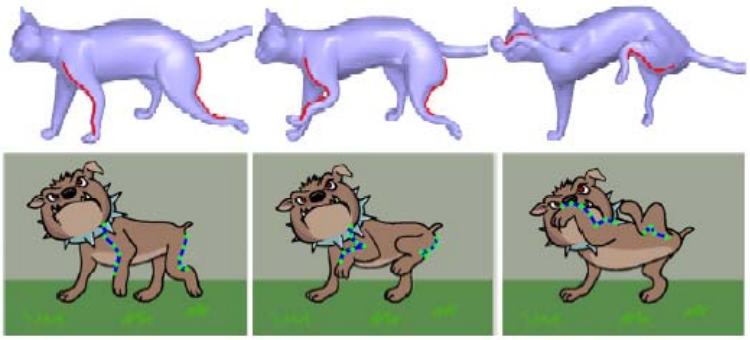


圖 2.4：藉由二維影像的資訊來對三維模型做形變[14]。

Sketch based shape modeling：以目前常見的三維模型製造軟體來說，對於一般使用者操作是相當困難的，主要是給專業美工人員使用。因此要提供簡單的介面讓使用者能夠編輯、控制模型，就變成相當重要的研究。[22]中提出了一套簡單操作的介面，讓使用者能夠利用滑鼠定義有興趣的區域，並且控制此區域做出想要的形變。由於使用者在看三維模型時，想控制或是注意的區域通常是邊界跟輪廓，因此在許多研究中也提供了使用者快速定義出這些特徵線或點的介面，並且利用滑鼠畫出的途徑來做為形變的輪廓依據，如[22, 23]中的介面。

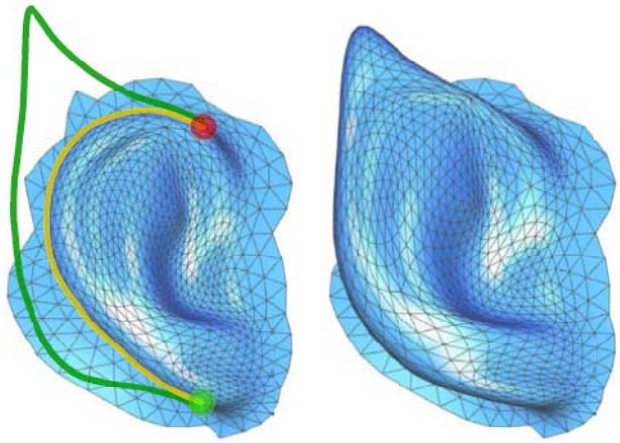


圖 2.5：利用介面快速定義和編輯輪廓[22]。

2.2 模型形變

三維模型形變的研究，就是針對模型調整其頂點位置，讓其能夠有新的動作或是形狀的變化。在變化過程中，必須要保持模型本身的完整性，包含了模型外觀細部特徵的保存、模型整體外觀的平滑、形狀的連續性，為了能夠保存模型表面的細節，並且達到各種不同的形變方式，產生了許多技術來產生良好的形變。

Multi-resolution techniques[21, 25, 26, 27]：此方法將三維模型分解成許多層級的 frequency bands，也就是將模型從原始的漸漸轉變為簡化過、頂點和面較少的模型。而在形變時，則是控制簡單、粗糙的模型去產生變化，形變完成之後再把細節慢慢加入，讓模型變回具有原本數量的點與面之外觀。由於形變與加回頂點的過程是兩個獨立的步驟，因此在形變幅度

較大時，容易產生誤差。

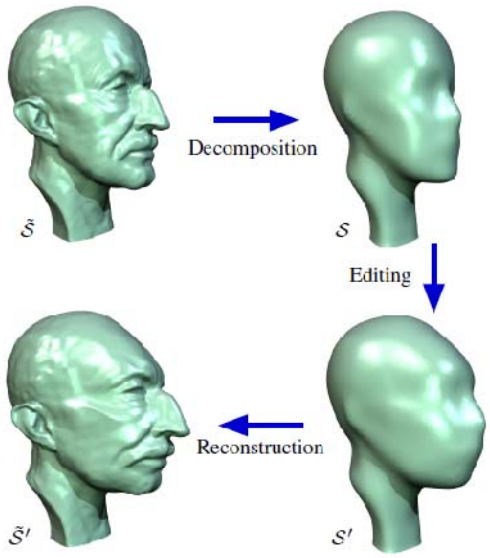


圖 2.6：Multi-resolution 之形變過程[27]。

Gradient domain techniques[2, 12, 13, 17, 18]：這個方法是目前三維模型形變最常使用的方法，重點是將模型形變的問題轉化為最小誤差(energy minimization)問題。此方法將形變時，保存表面細節以及使用者控制的頂點位置的資訊，轉化成數學的限制式，進而利用這些限制找出一個最佳化的形變，使得其誤差最小。

然而在此過程中，為了將表面細節構造保存，必須對頂點計算相對座標，也就是把頂點與週遭頂點的關係紀錄下來，而紀錄的方式許多研究是使用微分來處理，這會使得限制式變成非線性而必須使用高斯牛頓法(Gauss-Newton iteration)不斷逼近求最佳解，但會非常耗費時間。為了計算快速，已有許多技術，藉由不同的估計方法來將非線性問題轉化為線性

問題，如 local linearization[12]、interpolation from handles[14]。

其中在[12]中，提出的 laplacian coordinates 是最為廣泛使用的方法，利用矩陣儲存了每個頂點與週遭頂點的差異值，在形變過程中仍然盡量維持此能量，便可以達到保存細部特徵的形變。

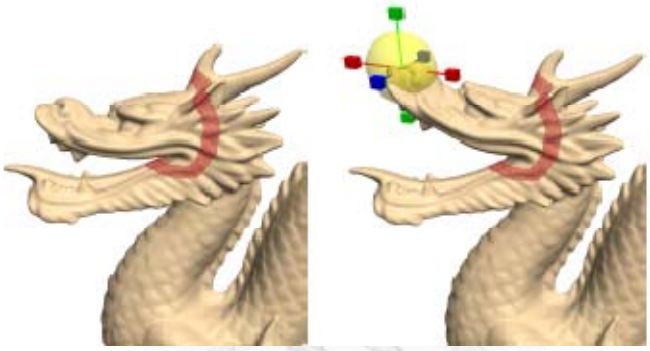


圖 2.7：藉由 laplacian coordinates 產生的模型形變[12]。

除了保存表面特徵，模型形變還需要一些額外的限制來達到各種不同需求的變形，在 2006 年由 J. Huang，K. Zhou 等人提出的[13]，提出了非常多種常用的形變限制式以及工具。

產生這些限制的需求是由於使用者在對模型做編輯和形變時，想控制的變化是非常抽象且難以控制的，因此衍生了許多工具和限制方式來輔助這些編輯的過程，例如在 free-form deformation (FFD)[28]中的控制格、[29]中的控制曲線、[30]中的控制點。除了工具，另外還有一些常用的限制方式：如骨架限制，藉由控制骨架的變形來產生因應的表面形變，使得模型的動作更符合使用者需求；體積限制，在形變過程中同時保持固定的體積；投射限制，把使用者欲控制的區域依使用者需求移動到新的三維空間中位

置。在[13]中介紹了相當多的技術轉化這些限制及工具成為數學限制函式，配合之前提到的 Gradient domain techniques，就能快速的產生適當的形變結果。

圖 2.8：包含體積限制之形變[13]。

Chapter 3

系統流程及概述

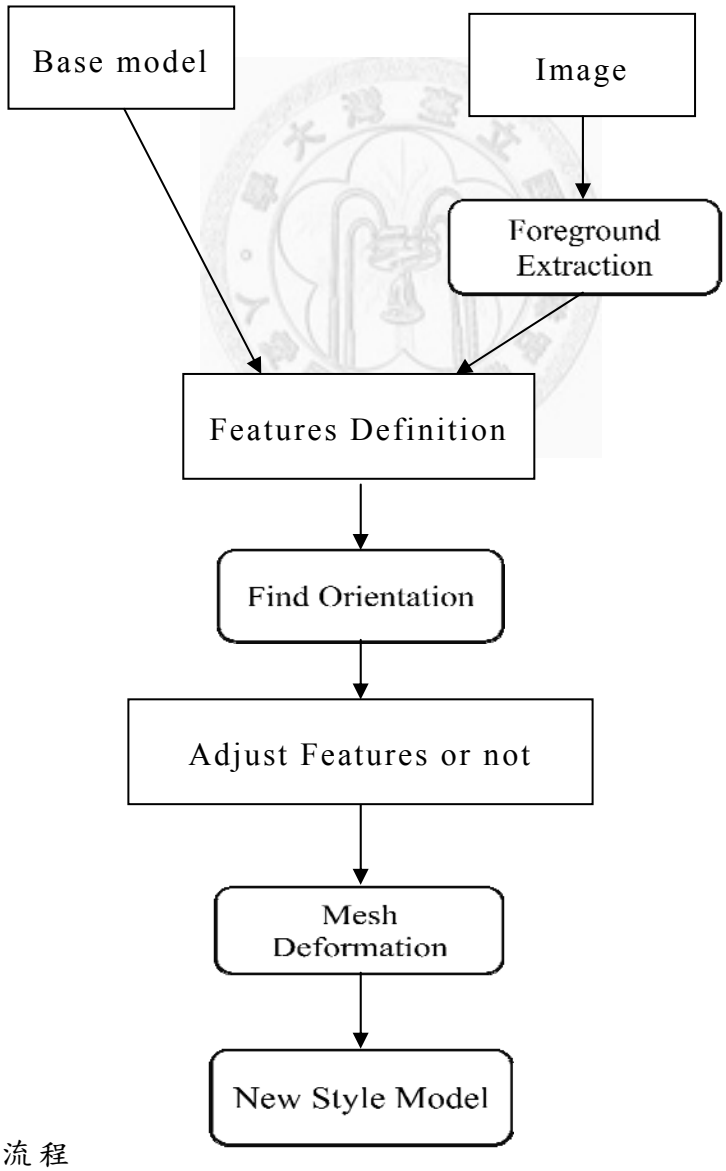


圖 3.1：系統流程

我們先對整個系統的流程做概述，在第四章會對每一個步驟的演算法做深入的介紹，而圖中圓形框的步驟代表著系統自動處理，而方框則是需要使用者操作給予資訊。第一步，載入三維模型以及二維的圖片：使用者欲載入的三維模型就是系統做形變的基礎模型，之後的形變都是由這個基礎模型去改變；二維圖片則是使用者希望給予形變的限制依據，在讀入影像之後會自動處理，將前景萃取出來。第二步，定義特徵點：系統提供介面，讓使用者能夠同時在三維空間及二維圖片上定義特徵點，每一組特徵點都會互相對應且記錄起來。第三步，分析方向：利用使用者定義之特徵點的空間位置，計算出最佳的方向，使得三維模型之特徵點的投射成像會最接近二維圖片上特徵點的位置。第四步，產生形變：在使用者確定方向正確之後，可以在繼續增加對應的特徵點或直接開始形變，系統會利用現有的特徵點以及圖片的二維輪廓資訊，加入投射限制、laplacian 限制，做出最佳化的形變結果，以產生使用者想要的模型。

Chapter 4

演算法

4.1 特徵點定義

4.1.1 影像分割與前景萃取

影像分割，就是把一張二維影像做分離，取出使用者想要的部分，如某個物體、前景、背景等。通常使用的方法是利用影像中畫格顏色的長條圖分佈，來決定某個畫格所歸屬的群組，其中包含了統計以及分群組的計算。目前已有許多的工具及演算法來達到影像分割的效果，以下將稍微介紹。

魔術筆：此方法是[7]中，由使用者來指定要分離的點，會計算此點周圍連結的其他畫格之顏色是否低於容忍值，也就是能夠跟指定的點歸類於同一群組，利用這個方法一直擴張下去直到所有能夠連結的區域標示完畢。

Intelligent Scissors：[15]中所提出的方法，藉由使用者利用滑鼠，大概點出物體的周圍邊界，而系統會算出一個最小成本路徑，在使用者邊移

動滑鼠時邊計算最佳的路徑，然而在貼圖複雜變動的部分，可能的路徑就有很多種，必須由使用者非常仔細的去確認哪個路徑才是希望的結果。

Graph cut：由[10]中提出的，利用了前景背景的長條圖分佈，以及建構一個演算法用的 graph，藉由計算最小路徑法來找出最好的分割，使用者可以快速的指定前景背景的位置，就可以得到不錯的結果。

GrabCut：此方法是將 graph cut 的演算法做了改進後的結果，在 2004 年由[11]中提出，使用者只要指定一塊區域作為前景的區域，便可快速產生影像分割。演算法中將原本的灰階分佈改成 Gaussian Mixture Model (GMM)，並且利用 iterative 的方法來最小化路徑的成本，藉此可以達到更精確的分割。

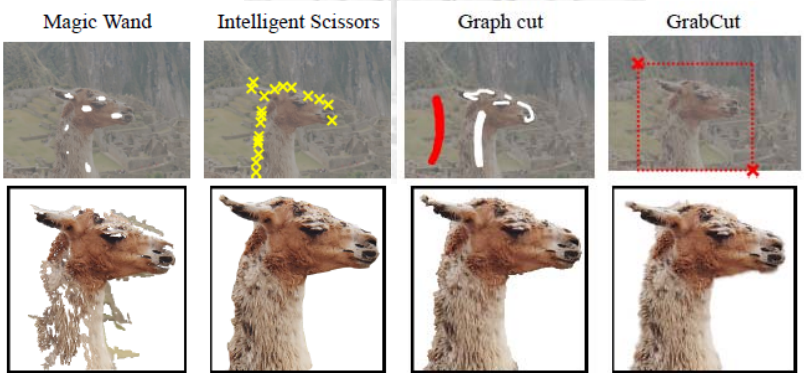


圖 4.1：各種影像分離的工具及效果[11]。

我們在二維影像上的處理，必須能夠得到我們希望的物體輪廓，以得到風格轉換所需要的資訊，因此我們要找出能夠分離二維影像前景背景的方法。如之前所述，二維影像的前景萃取已有非常多的研究方法，其中在[10]，BoyKov, Y.等人提出的利用 graph cut 來達到影像分割的方法，能夠

快速且有效的取出前景背景，不過在許多邊緣模糊:如毛髮，前景背景顏色混淆，都會產生一些不好的結果，而我們需要的輪廓資訊希望越準確越好，因此我們找了關於 graph cut 的延伸研究，最後採用了[11]中 GrabCut 這個方法，將原本的 graph cut 做了部分的修改，將影像模型(image model)改成 Gaussian Mixture Model(GMM)，並且加入了迭代預測(iterative estimation)以及不完全標記(incomplete labeling)，經過 GrabCut 可以讓使用者快速的操作，並且精確的取出前景輪廓資訊。

圖 4.2：前景萃取[11]。

在系統中，讀入影像之後的第一步，就是要先分離前景背景，讓使用者能清楚辨別目標物輪廓的形狀與位置，方便尋找特徵點。而此二維影像中目標物的輪廓資訊，是系統之後形變步驟的重要參考，因此在輪廓的處理必須精細且準確。在[10]中，Boykov 等人提出之使用 graph cut 做影像分割的演算法，其實就能快速的做前景背景的分離，但是在比較複雜的狀況下，常常會造成前景的輪廓有剝落跟不完整的情況，這是我們強烈希望能避免的。因此我們選擇了在 2004 年 Carsten, R.等人提出的『GrabCut』

[11]，來產生更完整、精細的前景輪廓資訊。GrabCut 是由 graph cut 的改進而來，所以會先重點介紹 graph cut 的演算法。

4.1.2 Graph cut

在 2001 年 Boykov 提出的這篇論文[10]裡，藉由 graph cut 來分離黑白影像前景背景的方法，主要可分為兩個步驟，首先第一步為使用者必須給定 trimap T，標記出影像的前景 T_F 與背景 T_B，接著採用 gray-level distribution 來表示前景背景的分佈。令陣列 $\mathbf{z} = (z_1,\ldots,z_n,\ldots,z_N)$來儲存整張影像，而 n 為其索引。而分割出的前景背景影像則存成另一個陣列 $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_N)$，而存的值是以 opacity 來表示，一般來說 $0 \le \alpha_n \le 1$，但在系統需要的 hard segmentation 中，則是 $\alpha_n \in \{0,1\}$，其中 0 代表了背景，而 1 代表著前景。此外，令一個參數 θ 來描述影像的前景和背景之 gray-level distribution，包含了這些數值的長條圖分佈，則 θ 代表式如下：

$$\theta = \{ h(z; \alpha), \alpha = 0,1 \}$$

θ_0 代表了背景的分佈，而 θ_1 代表前景的分佈，這兩個長條圖分佈的資訊來源就是使用者一開始給的 T_F 與 T_B。因此我們有了已知的影像 $\mathbf{z}$，前景背景的分佈 θ_0、θ_1，而未知的 $\boldsymbol{\alpha}$ 則是我們希望得到的數值，找出最好的 $\boldsymbol{\alpha}$ 就能良好的分離前景背景，因此要利用接下來的最小化能量函式：

$$\mathbf{E}(\alpha, \theta, \mathbf{z}) = U(\alpha, \theta, \mathbf{z}) + V(\alpha, \mathbf{z})$$

式子中的 U 的定義是一個估計的數值，用來估計加入 $\boldsymbol{\alpha}$ 至分佈之後與 $\mathbf{z}$ 的分佈 θ 的符合程度，定義這樣的 U 的理由，是因為我們希望找的 $\mathbf{E}$ 能夠做出一個很好的分割，也就是原本藉由使用者給予的 T 之前景背景的長條圖分佈跟加入 $\boldsymbol{\alpha}$ 的分佈應該要一致，為了達到這樣的效果，U 的計算如下：

$$U(\alpha, \theta, \mathbf{z}) = \sum_{n} -\log h(z_n; \alpha_n)$$

而 V 則是代表著影像中每一個畫格與其周圍鄰居的關係，定義成式子中的 smoothness term，表示如下：

$$V(\alpha, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} dis(m,n)^{-1}[\alpha_n \neq \alpha_m] \exp -\beta(z_m - z_n)^2$$

算式中的 $dis(\cdot)$代表的是 m 跟 n 的尤拉距離，$\mathbf{C}$ 為所有畫格與鄰居的配對關係之集合，$[\phi]$則是一個判斷函式，只看 ϕ 的結果給定 0 或是 1，m 與 n 相同就給定 0，不用計算其數值。γ 與 β 皆為系統給定的常數，γ 決定了整個式子對能量的影響程度，而 β 則代表考量 smoothness 的重要與否，當 $\beta = 0$ 則代表不論在何處 smoothness 都非常重要，而在[1]中有討論 γ、β 的給定造成的影響。

如此一來整個能量最小化的函式就定義完畢，我們希望得到之良好的影像分割就是求取一個全域最小值的：

$$\hat{\alpha} = \arg\min \mathbf{E}(\alpha, \theta)$$

這部分最小化的計算則使用了標準最短路徑法[9][10]來解，我們會替所有畫格以及前景背景建構一個 graph，如下圖所示：

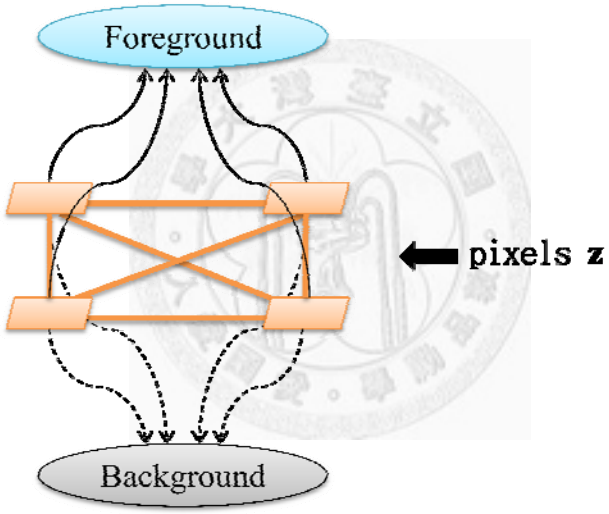


圖 4.3：Graph cut 之建構圖。

其中 U 定義的就是圖 4.2 中黑線及黑虛線部份，分別代表歸屬到前景或是背景的能量，若加入前景對前景本身的分佈來說很適當，則連結到前景的能量越小，反之越大；而 V 則代表橘線部份，也就是畫格之間由於顏色連續與否所定義的能量，顏色越接近則能量越大，代表可能畫格與鄰居可能為同一物體之顏色，反之則越小。將整個影像建構好圖形之後，找出最短

路徑來分割整個圖成前景跟背景，就可以產生合適的分割。

4.1.3 GrabCut

而 GrabCut 則是對於原本 Boykov[10]提出的方法提出了改進，首先是 gray-level distribution 的變更，他們將這個分佈模型改成了能適用更多影像的 Gaussian Mixture Model(GMM)，因此對於原本[10]中提出之能量函式 **E** 的 *U* 項，必須做些修改。此外，原本的函式只做一次的全域最小計算，而 GrabCut 將此計算改為一個梯次求解的過程，藉由學習上一個解出的最佳解，來找出是否有更好的解答。

首先是 GMM 的定義，我們對於前景跟背景的分佈會各給予一個 GMM，每個 GMM 由 *K* 個模型組成，一般來說給定 *K*=5，為了將 GMM 加入最小化能量函式，就必須額外增加一個向量 $\mathbf{k} = \{k_1,\ldots, k_n,\ldots, k_N\}$，每一個 $k_n \in \{1,\ldots K\}$，代表每一個畫格所指定的為前景或背景中第幾個模型。則我們的 **E** 會改變成如下：

$$\mathbf{E}(\alpha, \mathbf{k}, \theta, \mathbf{z}) = U(\alpha, \mathbf{k}, \theta, \mathbf{z}) + V(\alpha, \mathbf{z})$$

由於 GMM 具有 k 個模型，*U* 也必須重新定義成：

$$U(\alpha, \mathbf{k}, \theta, \mathbf{z}) = \sum_n D(\alpha_n, k_n, \theta, z_n)$$

其中$D(\alpha_n, k_n, \theta, z_n) = -\log p(z_n|\alpha_n, k_n, \theta) - \log \pi(\alpha_n, k_n)$，$p(\cdot)$為 Gaussian probability distribution，而$\pi(\cdot)$則代表 GMM 中的 mixture weighting coefficients。而因為部分為常數，我們將 D 重寫成下列式子：

$$D(\alpha_n, k_n, \theta, z_n)$$

$$= -\log\pi(\alpha_n, k_n) + \frac{1}{2}\log\det\Sigma(\alpha_n, k_n)$$

$$+ \frac{1}{2}[z_n - \mu(\alpha_n, k_n)]^T\Sigma^{-1}[z_n - \mu(\alpha_n, k_n)]$$

其中Σ代表相關變數，μ代表平均值，π為其 weighting。而 V 則維持原來的定義即可。

接著 GrabCut 需要用 iterative 的方法來求取最佳的圖像分割，整個演算法可分為下列步驟：

1. 使用者給定 Trimap T，形狀為長方形，將長方形內的 pixels 都設為前景，外的設為背景。
2. 藉著 1.得到的分割，分別做前景與背景的 GMM，這邊系統採用[5]中的分叢方法。
3. 將所有前景中的 pixels 指定到一個最接近的 GMM component，同樣的，背景中的每個 pixels 也指定到背景的 GMM 中最接近的 component。
4. 將舊的 GMM 丟掉，利用現在 pixels 與 components 的關係，創造一個新的 GMM。

5. 建立 graph，將所有連結的 penalty 算出，利用 graph cut 算出最佳解，將結果的前景背景紀錄下來。

6. 重複 3.~5.，直到收斂為止。

藉由此過程，可以讓使用者只要簡單的操作，即可產生精確的前景背景分離。

圖 4.4：GrabCut 做出的前景背景分離。

4.1.4 位置選擇

接著在系統中，使用者必須點擊滑鼠，在二維影像上定義其特徵點，不過使用者可能無法非常精確點到特徵點的畫格位置，因此這部分系統會自動計算最佳的位置來做為特徵點。在經過了 GrabCut 處理後，已經有了影像的輪廓資訊，因此只要將使用者點擊的位置，在一定範圍內，梯度最大的點做為特徵點即可，也就是若範圍內有凸出、尖角，就是適合的特徵點。不過在一些情況下，特徵點會定義在較平滑的位置，所以若範圍內的點梯度都很接近，系統則會選擇最輪廓上最接近的位置。

(a) (b)

圖 4.5：(a) 判定特徵點位置在梯度最大的位置。(b) 判定特徵點週遭為平滑面，因此找尋最近的點。

4.2 最佳化方向

此部分介紹的是能夠自動尋找最佳方向，使得三維模型特徵點在這個方向上能夠最符合二為特徵點的位置，就可以減少使用者需要自己調整三維模型方向的時間以及誤差。首先是找出三維特徵點經過投射後與二維特徵點的關係，如下列所示：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

令 **p**(X, Y, Z)為三維空間中的一點的座標，則(u, v)就代表了二維投射面上 **p** 投射的位置，而矩陣則就是表示其從三維投射至二維的投影矩陣 **M**，而這個投影矩陣包含了許多個重要的元素。第一，相機參數矩陣，也就是我們看出去的方向以及起點；第二，投射矩陣，其中含有投影成像的平面位置，以及定出三維空間裡要投影的範圍。第三，仿射(affine)矩陣，代表了三維空間內物體做的移動、旋轉。由此得知，在實做上，假設目前藉由使用者提供的一組特徵點(f_m, f_i)，其中f_m = (x, y, z)為三維空間中的特徵點位置，而f_i = (u, v)為二維空間上的特徵點座標，而其中轉換的矩陣為 **A**，我們所希望的最佳方向的計算函式就會如下：

$$\mathbf{\Pi} = \min \sum_{(f_m, f_i) \in \boldsymbol{F}} \mathbf{A} f_m - f_i$$

其中**Π** 代表了我們所希望得到的最佳方向矩陣，而 ***F*** 則是各組特徵點的集合。在實際計算最佳方向的投射矩陣中，我們增加了加速的步驟在其中，由於前面提到的，投影矩陣包含了相機參數、投射、仿射這三個部分，而其中的相機參數、投射矩陣在系統中是固定的，使用者在操作介面時就是用這些已經訂好的常數在控制增加特徵點，因此未知項只有仿射矩陣的部分，也就是物體的大小、平移、旋轉，藉由這部分的資訊，可以將系統計算簡化，只要最佳化仿射矩陣即可。

下面是最佳化方向後的結果：

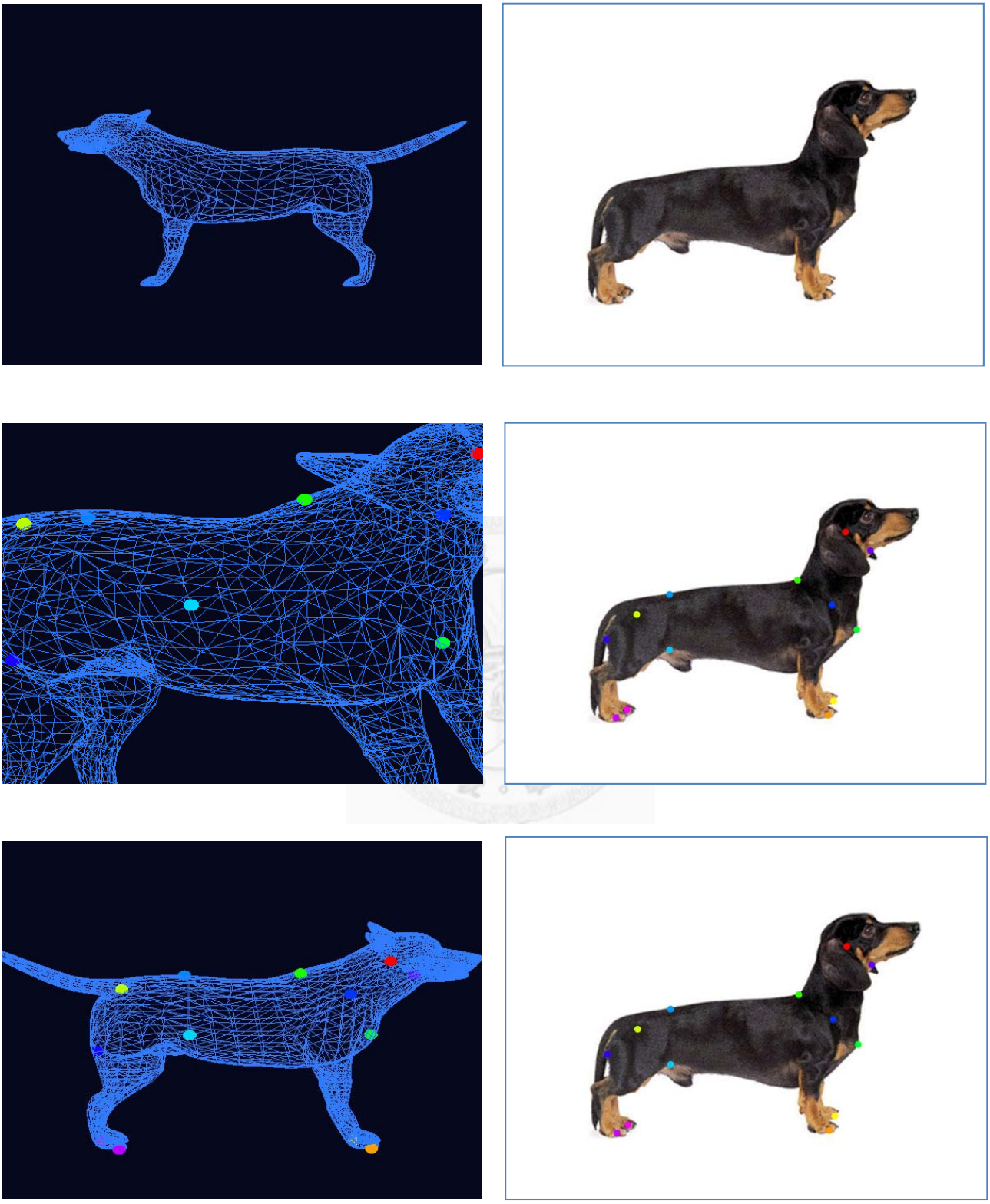


圖 4.6：藉由特徵點自動計算方向。

由於這部分只是在計算最佳的方向，因此使用者再定義特徵點時可以定義較少、並且兩個視窗中原本位置不會相差太大的頂點作為特徵點，一來可以避免因為差異過大的特徵點影響整個最佳化，另外也可以提升精確度。

4.3 模型形變

三維模型的變形，要考慮的因素很多，必須要考慮頂點位置的變化、相連的邊的長短調整、整體模型的完整性及平滑，以及模型本身的細節保存，在電腦圖學的模型領域也是一個非常重要的問題。在電腦圖學領域裡，此主題已有非常多且深入的研究。其中保持模型細節的方法，在[12]裡，定義了三維模型的 laplacian coordinates，是之後的形變研究參考的重要論文，利用 laplacian coordinates 可以在形變過程中保持許多模型的外觀細節，然而在大幅度的形變上，還是很容易使得三維模型產生錯誤。

而在近幾年的形變研究中，就以[13] [14]由微軟亞洲研究院提出的論文最為出眾，在這之中提出了許多不同的限制式來幫助形變，有骨架限制、體積限制、投影限制，另外也提出了非線性的最小化解法，除此之外還做了非常多的加速方式，不但將大幅度形變的效果變更好，保存細節更精細，還加快了對複雜模型的運算速度，是形變領域中一大突破，而其中限制式的定義對於我們系統的形變非常有用。

在前幾個步驟中，系統得到了使用者指定的特徵點資訊，二維影像的輪廓，三維模型最佳的方向。為了產生新的風格化模型，藉由使用者給予欲變形的特徵點位置，將三維模型形變之後，讓其盡量滿足使用者指定的特徵點位置。而形變的限制分為兩部分來達到這樣的效果，最佳化的形變函式如下：

$$\mathbf{V} = \min(\mathbf{P} + \mathbf{L})$$

V 代表著新的三維模型之頂點集合，**P** 為投射限制的函式，而 **L** 則代表 laplacian 限制的函式。我們希望得到的結果，是能夠符合使用者給定的特徵位置，同時保存模型本身細節的模型，因此將兩個限制式的項目加入，並且求取一個總合的最佳解。

4.3.1 投射限制

投射限制的目的，就是原本三維模型上的某個頂點，藉由移動到一個新的三維空間位置，而投射到使用者指定的投射平面位置上。典型的形變系統通常都會利用二維圖形介面來讓使用者直接控制三維模型上的點，拖曳這些點讓它們的投射位置符合使用者的要求。

我們系統提供了不同的做法，在使用者定義了特徵點的位置之後，我們直接利用這些特徵點的投射來控制模型的形變。以往的做法通常是使用者利用介面，直接由使用者對一個或多個頂點做拖移或旋轉，使的目標頂

點移動到使用者欲形變的位置。然而，在多數的情況下，這樣的操作雖然方便，卻是複雜且精細的，不僅花費時間龐大，同時難以一次調整許多位置，在過程中容易出錯。我們的系統則省略了這部份的人為控制，一來在移動的過程中，動作要很仔細且精準，若是要變成使用者想要的一個完整模型，要花費許多的時間及專注力。藉由我們的系統簡化了使用者需要親自操作的部分，減少了花費的時間以及精神，利用直接指定多組特徵點的方式，來達到以往使用者必須手動調整多個頂點的效果。在三維模型投射上的基本情況如下：

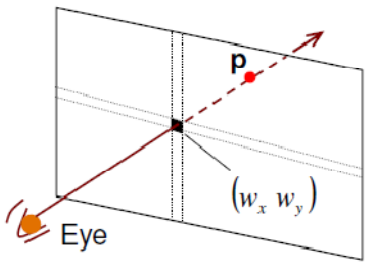


圖 4.7：投射限制之圖示[13]。

點 **p** 經過投射之後成像在(w_x, w_y)的位置，而投射限制，就是要讓任一點 **p**，移動到一個三維空間中的位置，使得此 **p** 在成像之後能夠準確的落在使用者指定的位置上。為了定義這樣的限制，我們令 $\mathbf{p} = Q_pF$，F 代表某個三維模型上特徵點之三維空間位置，而 Q_p 則是代表著我們的投射限制矩陣。此外，令 M 為原始將三維空間物體轉換至成像平面的投射矩陣，且令 f 為成像平面的焦距。定義好之後就可以將 **p** 成像至平面的計算寫成：

$$\left(f\ \frac{M_x^r \mathbf{p} + M_x^t}{M_z^r \mathbf{p} + M_z^t}, f\ \frac{M_y^r \mathbf{p} + M_y^t}{M_z^r \mathbf{p} + M_z^t} \right)$$

式子中的M^r表示著 M 中的旋轉量，而M^t則代表著 M 中的平移量，接著我們將使用者的目標位置設為(w_x, w_y)，就可以寫出下列式子：

$$(f\ M_x^r - \ w_x M_z^r)\ \mathrm{Q_p F} = \ -f\ M_x^t + w_x M_z^t$$

$$\left(f\ M_y^r - \ w_y M_z^r\right)\ \mathrm{Q_p F} = \ -f\ M_y^t + w_y M_z^t$$

解出適當的 $\mathrm{Q_p}$ 之後，我們就得到了形變所需的資訊，也就是三維模型上之特徵點要移動到的新位置，使得特徵點能符合使用者指定的位置。最後，將我們的投射限制寫成下列式子：

$$\mathbf{P} = \ \mathrm{QF} - \mathrm{W}$$

Q 為我們求出來的投射限制矩陣，而 W 則是使用者指定的頂點集合，在我們系統中就代表著二維特徵點的位置。將這個式子加入最後要最佳化的函式，讓我們的形變過程中會盡可能的符合使用者給予的特徵點。

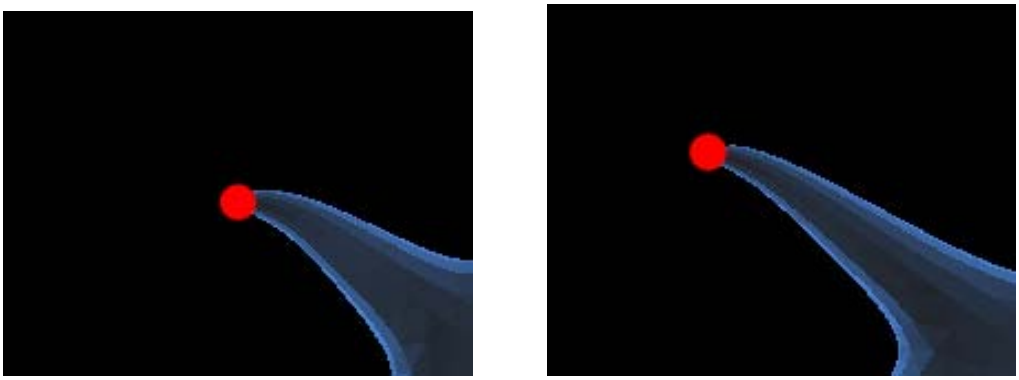


圖 4.8：利用投射限制拖移模型頂點。

4.3.2 Laplacian 限制

在形變過程中，為了保持三維模型的細節，我們加入了 laplacian 限制式，laplacian coordinates 中儲存了模型中每一個頂點與週遭頂點的差異值，也就是把與週遭頂點的關係做量化，利用盡量使這個數值不要太大的變動，進而保持模型上的細節特徵。

這個方法主要參考了[12]的內容，首先我們要定義模型的 laplacian：

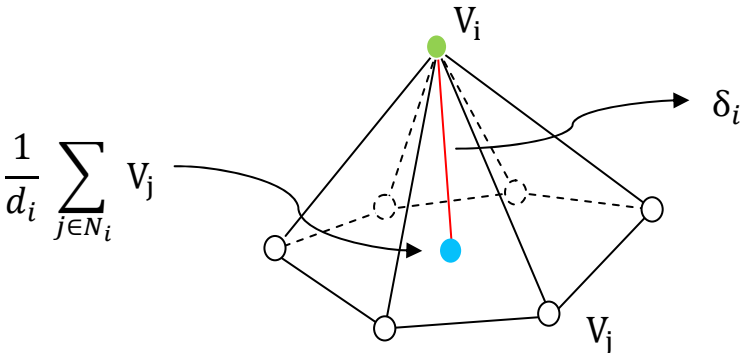


圖 4.9：Laplacian 之圖示。

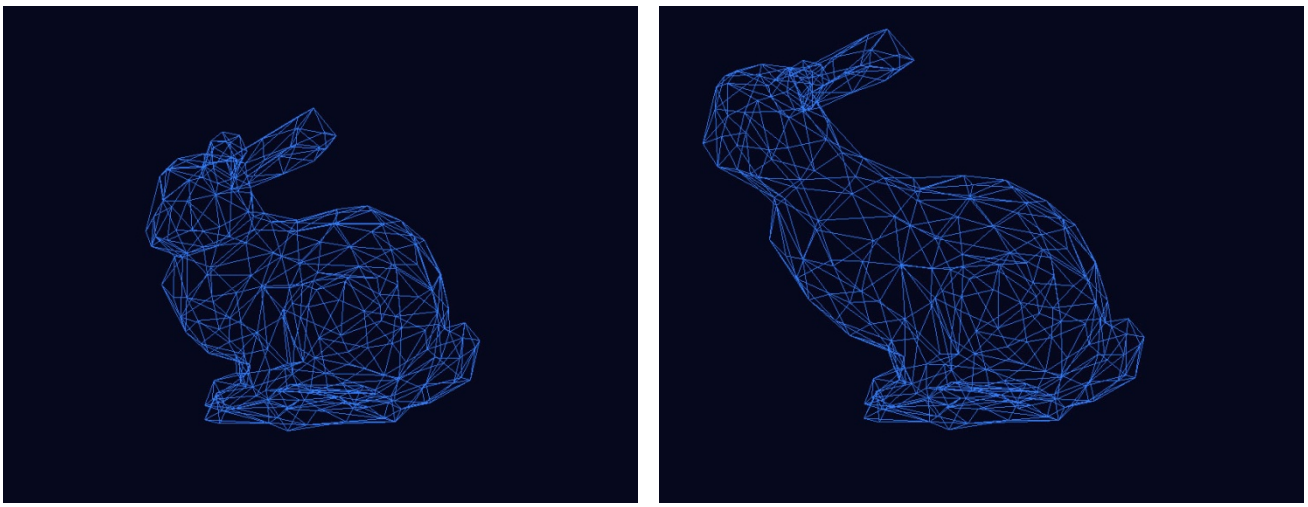
令 laplacian coordinates 表示為δ_i，其計算方式如下：

$$\delta_i = V_i - \frac{1}{d_i}\sum_{j \in N_i} V_j$$

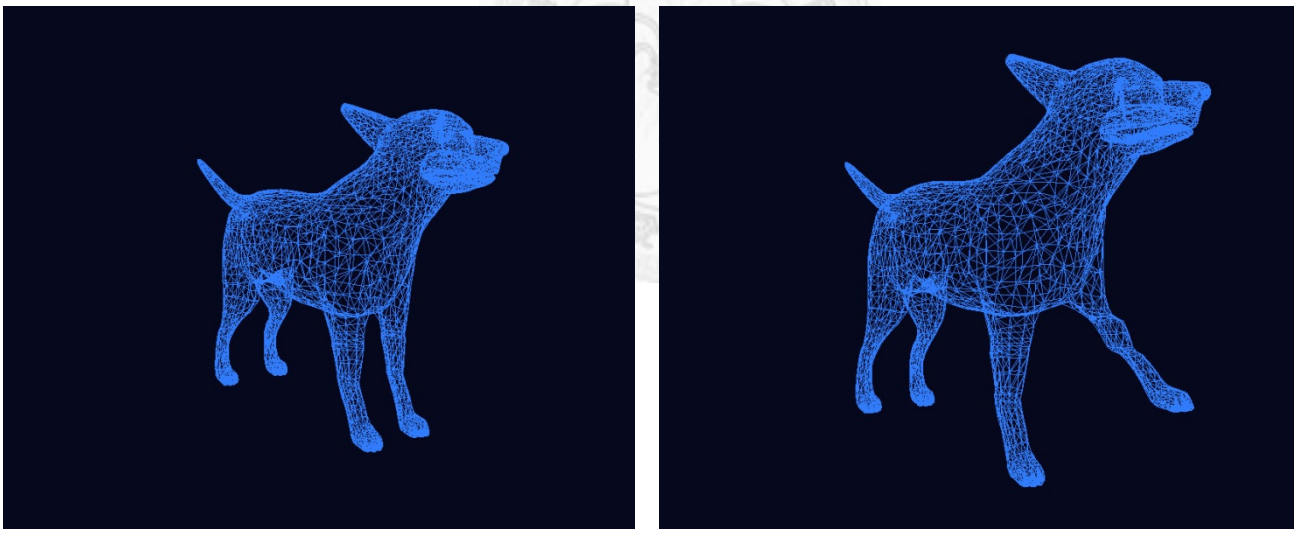
Laplacian coordinates 中儲存了點 V_i 與週遭頂點之平均值 V_j 的差異，代表的就是在頂點 V_i 上的細節資訊。將 laplacian 限制加入的原因，是為了在拉動三維模型時，除了輪廓的改變之外，另外能保留模型本身的細部特徵，而 laplacian 則是在做三維模型形變的領域中最常使用的方法之一。利用計算出來的 laplacian coordinates，我們定義的限制如下：

$$\mathbf{L} = \| \mathrm{L}(\mathbf{v}) - \mathrm{L}(\mathbf{v}') \|^2$$

在算式中，**v** 即是形變後三維模型的頂點集合，$\mathbf{v}'$則表示變形前的模型頂點，L(・)則是 laplacian 運算式，也就是計算出來的 laplacian coordinates，把這個加入我們的最佳化函式，盡量使得變形前後的 laplacian 差距越小越好，以維持模型的細節不變。最後同樣的把 laplacian 的限制式加入整個形變的函式中，以求得整個函式的最佳解，也就是最佳的形變結果。



(a) (b)



(c) (d)

圖 4.10：利用 laplacian 限制，使模型形變後能保持表面細節特徵。(a) 原始兔子的模型；(b) 拉動兔子頭的部分；(c) 原始狗的模型；(d) 拉動狗左前腿的部分。

4.3.3 形變結果最佳化

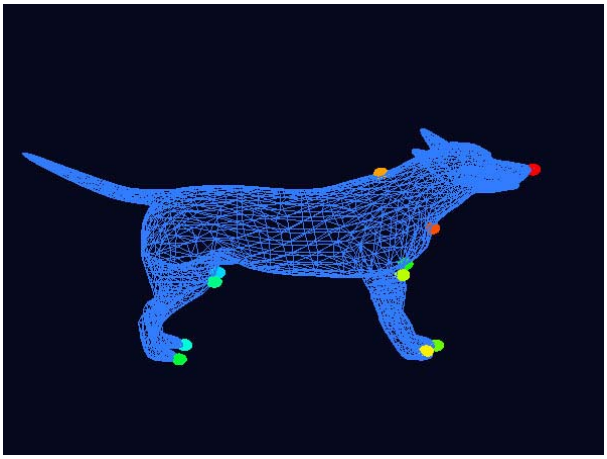
在系統實作中，模型形變主要的限制式包含了投射限制 **P** 及 laplacian 限制 **L**，我們希望能夠盡量的符合使用者指定的投射限制，但一方面又不能破壞模型的完整性與細節，因此必須計算出最小化的 **P+L**，也就是最佳的形變後模型。因此我們將整個最佳化計算過程寫入矩陣，並且利用 taucs library[16]來幫助運算。

首先系統在讀入模型之後，就會開始建構模型形變所需的 laplacian coordinates，並存入矩陣，這部分是形變過程中花費最久的時間，模型越大越複雜，建構時間會越長。而在使用者定義特徵點完畢，並且自動最佳化方向後，系統就會利用最佳化方向之結果來產生投射限制的矩陣內容，假設某一組三維空間與二維空間之特徵點 (f_m, f_i)，利用二維特徵點位置 f_i，計算出該特徵點在三維空間對應的新位置 f_m'，而此新位置就是原本三維特徵點 f_m所對應的投射限制，接著把此限制式加入我們的最佳化矩陣，依序把所有組特徵點都加入之後，就完成了我們所希望求得之最佳化矩陣，計算式如下：

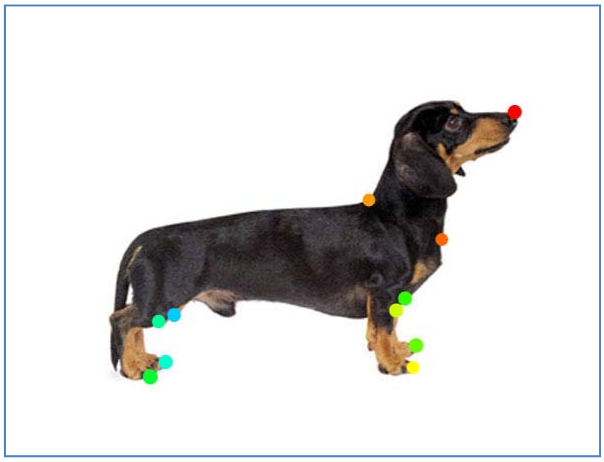
$$\mathbf{MV = X}$$

算式中的矩陣 **M**，存入的就是包含 laplacian 限制式以及使用者給定特徵點之投射限制式，**X** 是原始模型中算出的 laplacian coordinates 以及特徵點新位置 f_i' 的座標。而 **V** 就是最佳化後的頂點位置集合，藉由此最佳化可以得

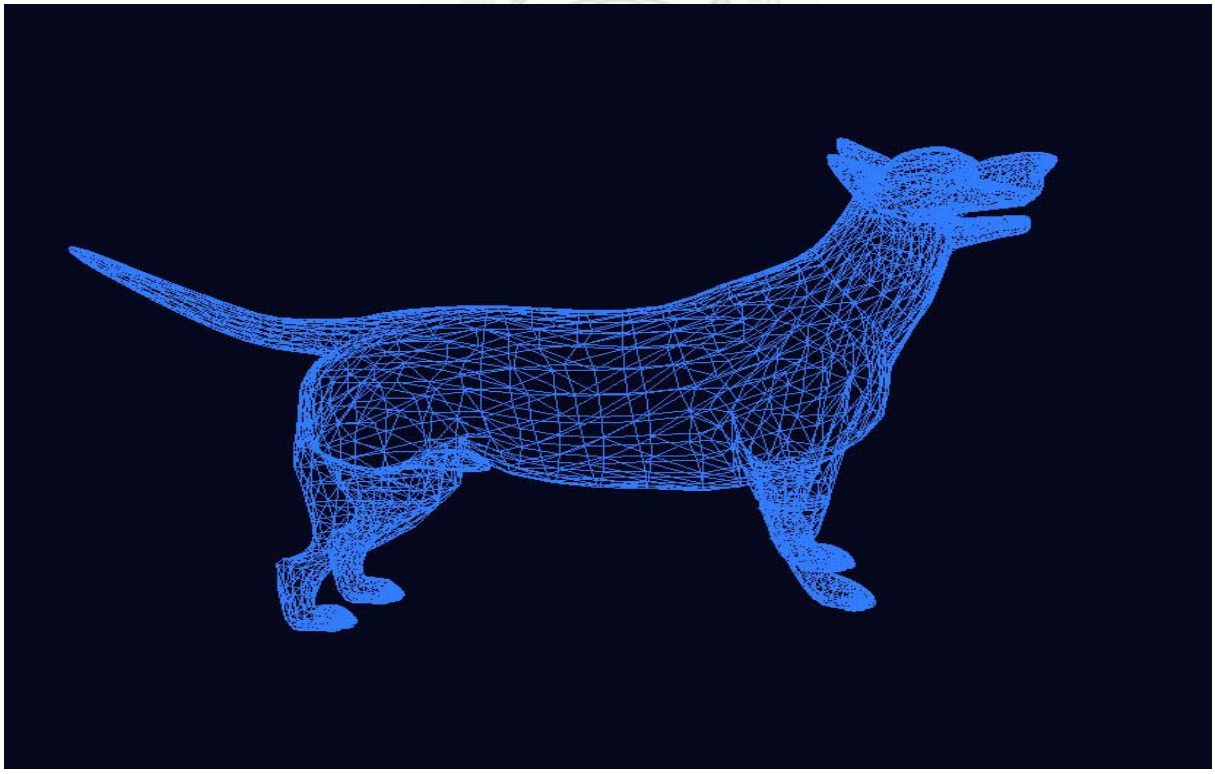
到符合使用者定義之特徵點，且具有本身模型特徵細節之風格化模型。



(a)



(b)



(c)

圖 4.11：(a),(b) 定義特徵點配對；(c) 符合使用者風格化之模型。

Chapter 5

系統成果

我們設計了一個半自動的系統，用來快速產生具有使用者指定風格的三維模型。利用這個系統，使用者便不需要花費龐大時間在重新製作類似的模型，而可以簡單的定義數個特徵點，即可讓系統生成保有原始三維模型細節的新模型。

系統實作部份，使用 C++語言，另外包含了 Fltk、OpenGL、Taucs、OpenCV，其中 Taucs 是專門設計來解決 sparse linear system 問題的函式庫。由於系統分成數個步驟，而中途會需要使用者操作，因此分段來提供系統花費時間及效能。

由於影像在讀入之後，系統會先裁切或縮放至固定大小符合介面的視窗(系統設定為 600*450)，再進行 GrabCut，另外在計算方向的參數是使用者指定的特徵點，因此數目不定，同時使用者需要時間也不固定，系統測試結果採用了六組結果。下表為硬體資訊：

表 5.1：系統資訊

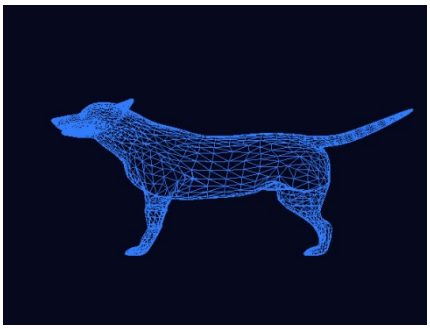
CPU：Intel Core 2　E4600 @ 2.40 GHz
RAM：3.5 GB
Video card：GeForce 9800 GTX

在形變的部分，系統需要建構大型的矩陣來儲存 laplacian 與投射的限制式，因此模型的複雜度會影響此步驟耗費時間，而處理完矩陣後的最佳化過程基本上都是即時進行的，下表為建構矩陣以及各個步驟耗費的時間，另外也包含特徵點數目和模型資訊：

表 5.2：模型資訊及系統操作時間

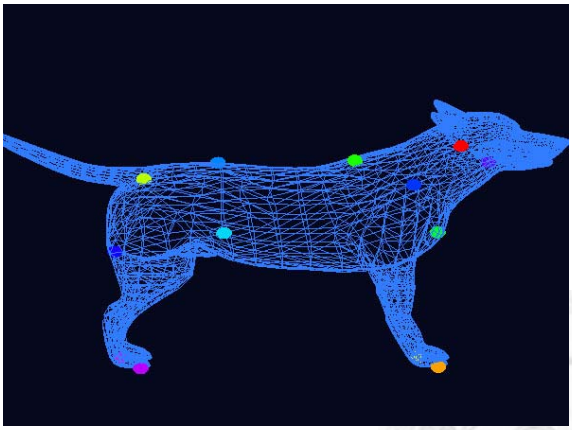
Model name	**Number of vertex/ faces**	**Features for alignment / deformation**	**Orientation alignment (sec)**	**Build laplacian coordinates (sec)**	**Mesh deformation (sec)**
Dog(1)	4,070/8,135	10 / 11(10-2+3)	0.328	0.941	0.439
Dog(2)		10 / 16(10-1+7)	0.347		0.463
Starfish(1)	1,890/3,775	7 / 10(7-2+5)	0.271	0.392	0.296
Starfish(2)		7 / 10(7-2+5)	0.248		0.276
Horse(1)	502/999	10 / 25(10-3+18)	0.313	0.136	0.116
Horse(2)		10 / 25(10-3+18)	0.361		0.092

最後是結果呈現：



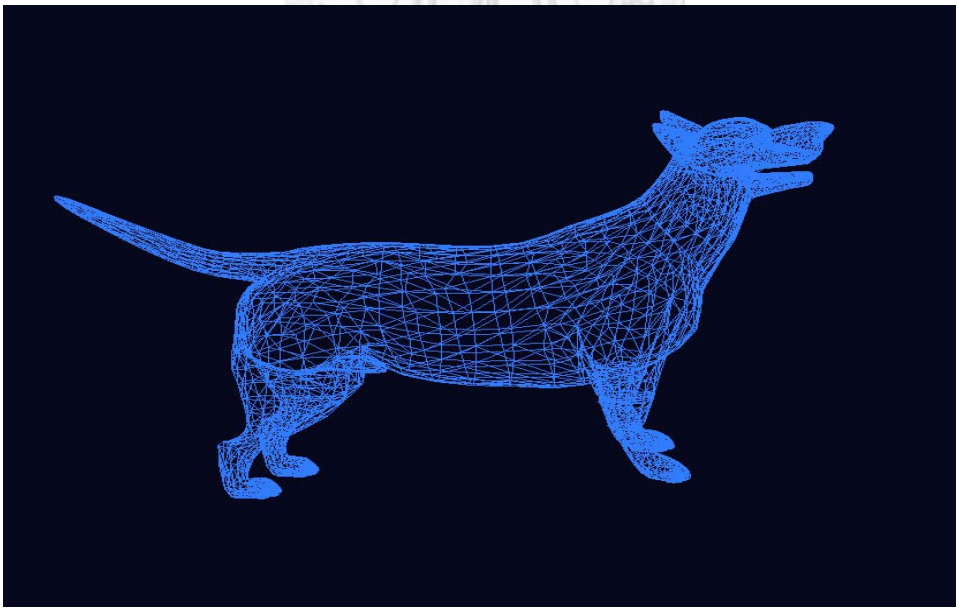
(a)

(b)



(c)

(d)



(e)

圖 5.1：(a)~(c)為經過前景萃取之圖形及原始三維模型，並經由使用者定義特徵點；(e)為最後結果，具有短腿長脖子之風格化模型。

圖 5.1 中包含了所有步驟中的結果，接著就直接呈現形變後的風格化模型，包含改變形狀或是動作：

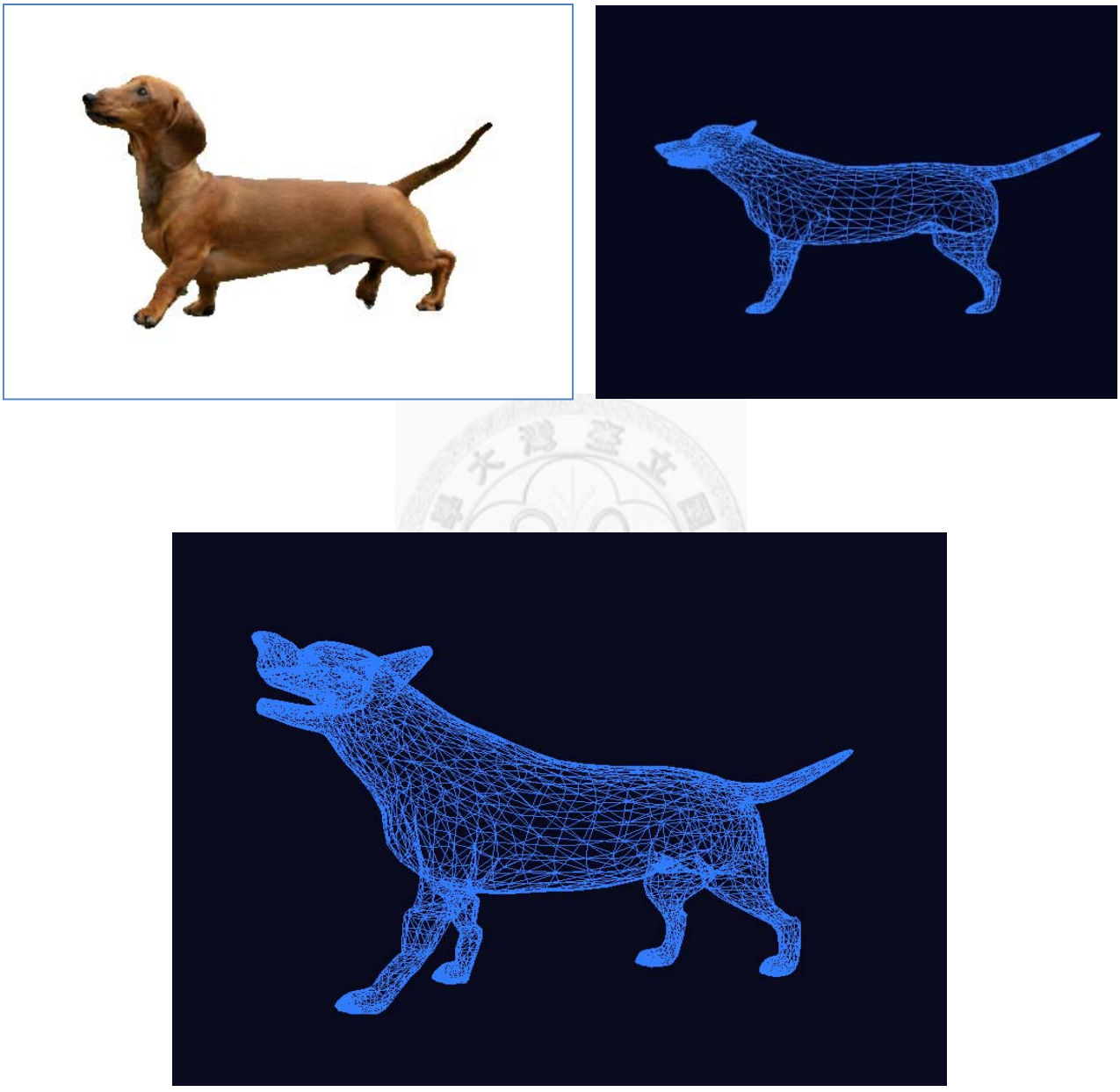


圖 5.2：藉由二維影像風格化的結果。

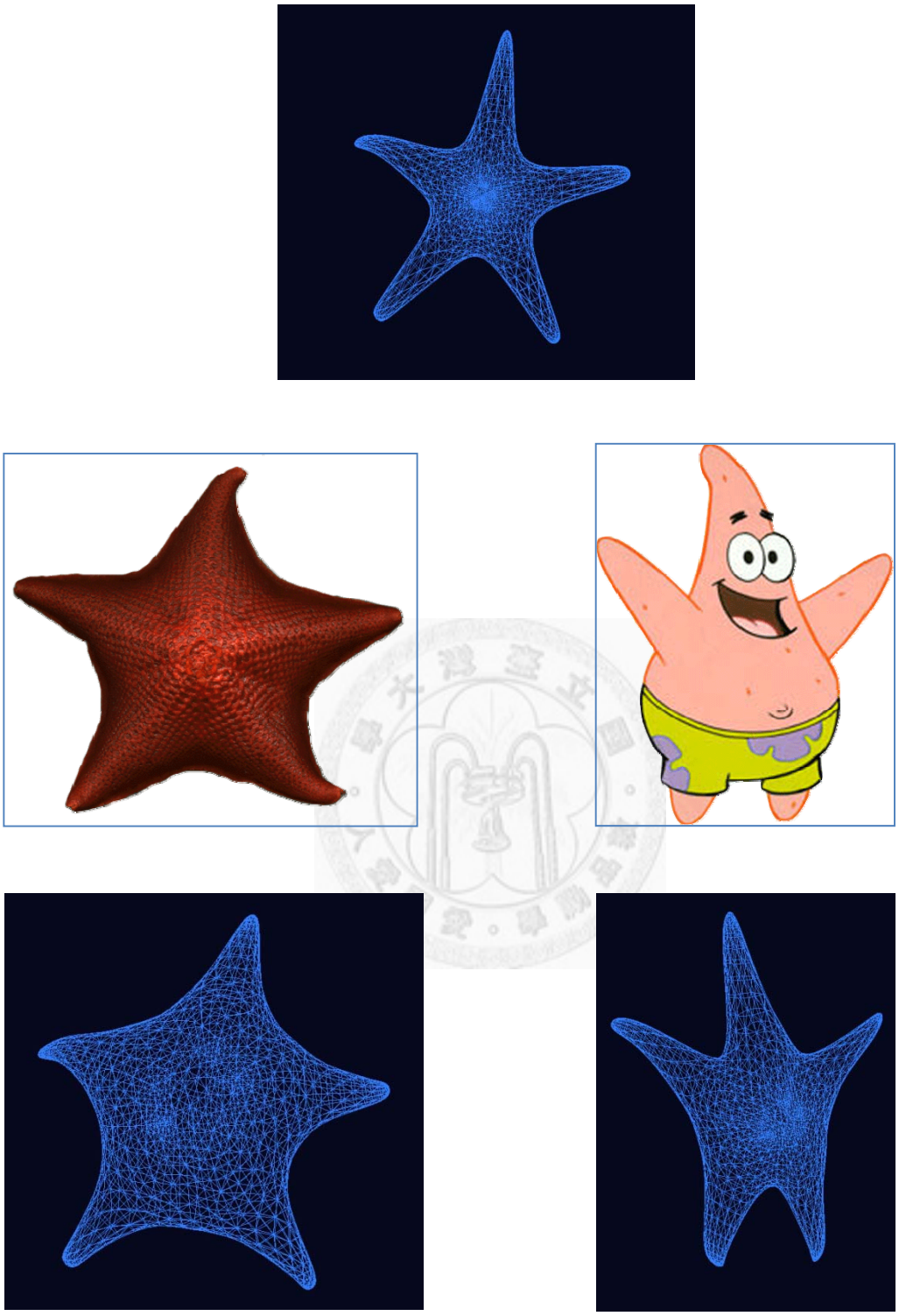


圖 5.3：最上方是輸入的原始模型之海星，中間列是讀入的不同風格二維影像，最下列是經過系統風格轉換後的結果。

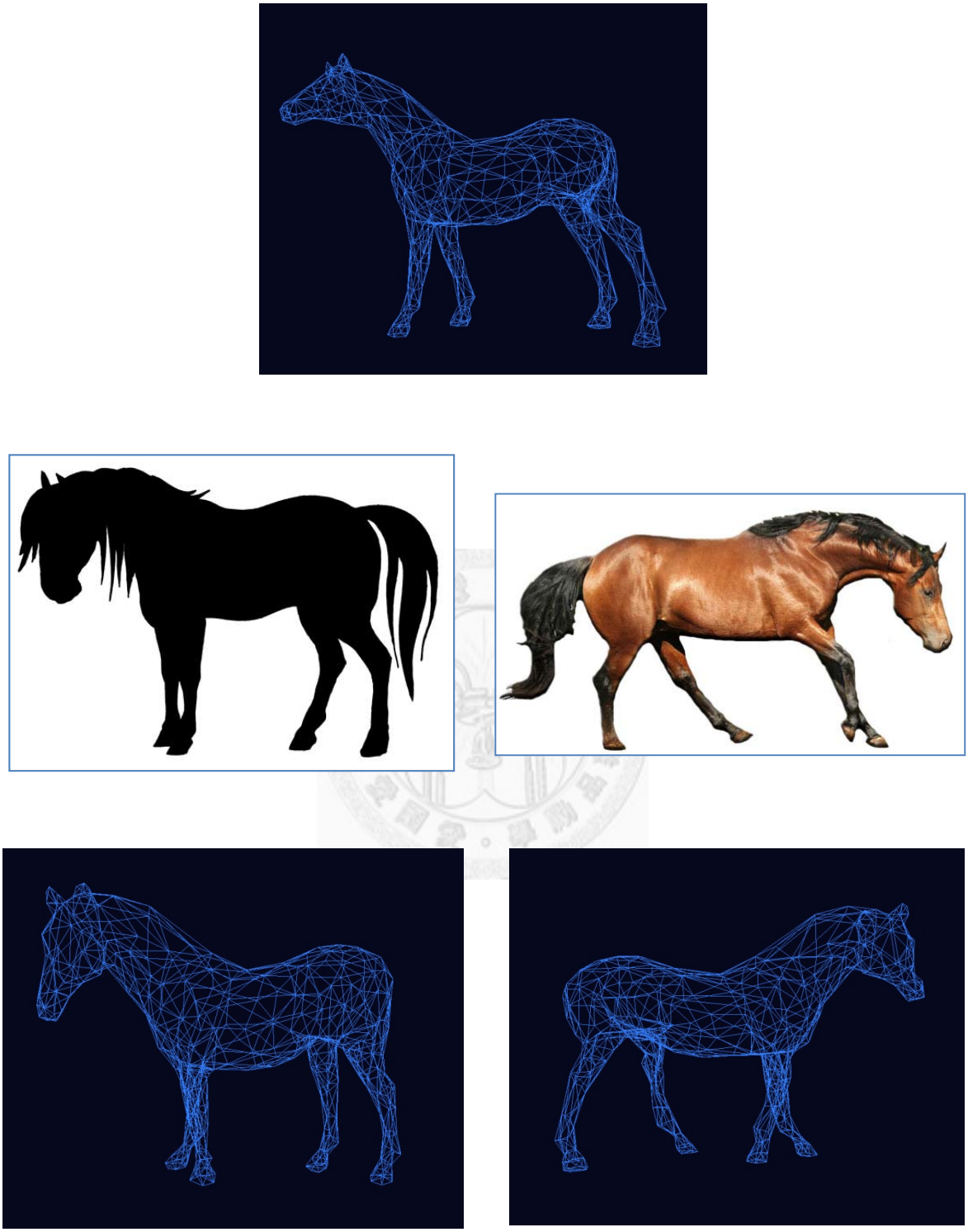
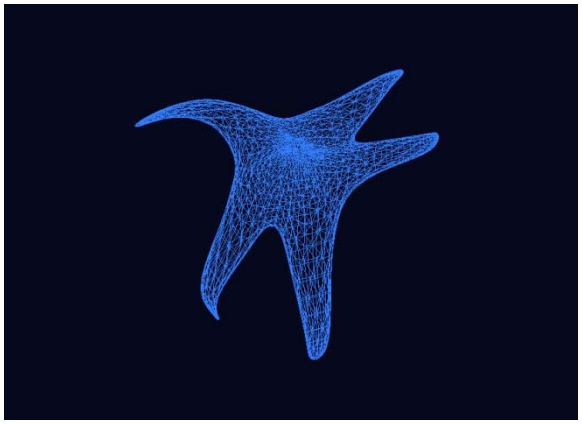


圖 5.4：最上方式輸入的原始之馬模型，中間列是讀入的不同風格二維影像，最下列是經過系統風格轉換，產生動作形變的結果。

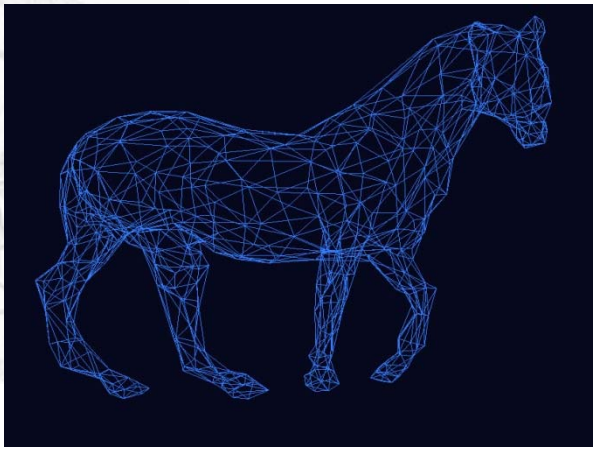
另外也呈現一些形變失敗的結果：

(a)



(b)

(c)



(d)

圖 5.5：風格化失敗的輸出模型：(a), (b)為利用圖 5.3 的海星原始模型做的風格化結果；(c), (d)為利用圖 5.4 的馬原始模型產生的結果。

從失敗的結果中，在圖 5.5(b)可以發現，在原始模型與風格化之影像的差異過大時，由於形變需要改變的資訊太多，在最小化誤差能量的過程中，就會影響到最後的結果，使得模型的形變不完全，且有些位置根本無

法位移到正確位置上，也就是在模型形變中，光靠 laplacian 限制與投射限制，在過於複雜的變化時容易產生錯誤的結果；而圖 5.5(d)中，馬的腳變動的幅度也相當大，除此之外其實馬的腳在模型中應該是要有骨架的，藉由骨架才能夠表示馬腳的動作，因此在形變時不僅腳的形狀不能符合二維影像中給予的限制，甚至在馬腳的整體表面都產生了凸起或不平滑的結果，若加入骨架以及鋼體限制，馬腳的形變效果應該能夠大幅提升。

Chapter 6

結論與未來工作

本文結合了關於影像處理以及模型形變，設計了一套半自動的系統，藉由前景萃取，二維與三維的對應關係，加上三維的限制做出形變，讓使用者能簡單的操作，就可快速產生新的風格化模型，大大的降低生產三維模型的消耗時間與成本。利用此系統，就能夠大量產生保有原本模型細節構造，並且符合使用者給予之外形的模型，在關於電腦圖學的產業上都有相當大的發展空間。

本系統最大的優點包含兩個方面：第一個是操作簡易且具有對應資訊的使用者介面，使用者可以輕易的從三維模型與二維影像中點出對應關係的特徵點，藉由系統輔助處理，能夠自動判斷出該位置最佳的特徵點，另外還有創新的自動計算方向，讓使用者能夠減少調整方向的時間以及錯誤。第二個則是自動化的形變，以往形變都需要使用者做相當複雜且精細的操作，許多情況下甚至很難控制模型變化成想要的形狀，藉由指定特徵點的

方式，本系統能夠最佳化且快速的產生符合特徵點要求的模型，這部分對於使用者的幫助是非常巨大的。

然而，雖然系統目前擁有這兩項優點，但在處理部份形變時仍然會遇到困難。例如形變中的旋轉，就沒辦法在系統中利用三維二維特徵點的資訊來達到；另外的問題是，當形變幅度過大，光利用 laplacian 限制式會沒辦法保存模型的外觀，容易產生失敗的模型，如圖 5.5。不過目前在圖學領域的研究，也有相當多的方法可以做之後延伸的方向，像是藉由使用者介面提供額外的旋轉資訊，以及更多進階的限制式，如剛體限制、骨架限制、較複雜的 laplacian 限制，或許就能夠解決上述的問題。

另外未來的主要工作，希望能夠藉由自動嵌入骨架的系統，還有貼圖系統，讓本系統能夠功能更加完備、使用更加方便。我們期望配合自動嵌入骨架，以及利用骨架動作資訊，不只能夠像系統現在快速產生三維模型，還能夠讓新產生的風格化模型套用原始模型的動作，使得新模型不需要再重新製作一次動作的資訊。而貼圖系統，則是希望能夠快速的將影像轉移到新產生的模型上，如此一來就能達到真正快速的產生一個新的、包含動作及表面貼圖的三維模型。

參考文獻

[1] Guanghua Tan, Wei Chen and Ligang Liu.: **Image Driven Shape Deformation With Styles.** *In Poster of Pacific Graphics* 2008.

[2] Tsung-Yu Tsai 2008: **Constraints-based 3D Model Deformation.** *Master thesis, National Taiwan University.*

[3]Shih-Chaing Dai 2008: **Animating Characters in Pictures.** *Master thesis, National Taiwan University.*

[4] Orchard, M. T., and Bouman, C. A. 1991. : **Color Quantization of Images**. *IEEE Transactions on Signal Processing 39*, 12, p. 2677–2690.

[5] Catherine S., Vincent B.. **Comics Stylization form Photographs**. *International Symposium on Visual Computing 2008*.

[6] Scott S., Travis M., Joe W.. **Image Deformation Using Moving Least Squares.** *In Proceedings of ACM SIGGRAPH 2006* p. 533-540

[7] Adobe Systems Incorp. 2002. **Adobe Photoshop User Guide.**

[8] Igarashi T., Matusoka S., Tanaka H. **Teddy: A sketching interface for 3D freeform design**. *In Proceedings of ACM SIGGRAPH* 1999 p. 409–416.

[9] Kolmogorov, V., and Zabih, R. 2002. **What energy functions can be minimized via graph cuts?** *In Proc. European Conference on Computer Vision 2002.* p.65-81.

[10] Boykov, Y., and Jolly, M. 2001. **Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images.** *In Proc. IEEE Int. Conf. on Computer Vision.*

[11] Carsten R., Vladimir K., Andrew B. **"GrabCut" – Interactive Foreground Extraction using Iterated Graph Cuts.** *In Proceedings of SIGGRAPH 2004.*

[12] Sorkine, O., Cohen-Or, D., Lipman, Y., Alex, M., Rossl, C., and Seidel, H. 2004 : **Laplacian surface editing**. *In Proceedings of the symposium on Geometry processing,* p.175–184.

[13] Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.-Y., Teng, S.-H., Bao, H., Guo, B., and Shum, H.-Y.. **Subspace gradient domain mesh deformation.** *ACM Transactions on Graphics (TOG),* 25 (3):p.1126–1134, 2006.

[14] Zhou K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B. and Shum, H.-Y.. **Large mesh deformation using the volumetric graph laplacian**. *In Proceedings of SIGGRAPH 2005 / ACM Transactions on Graphics (TOG)*, 24 (3):p.496-503, 2005.

[15] Mortensen, E., and Barrett, W. 1995. **Intelligent scissors for image composition.** In *Proceedings of ACM SIGGRAPH 1995*, p.191-198.

[16] Toledo, S.. **Taucs: A Library of Sparse Linear Solvers, version 2.2.** Tel-Aviv University, Available online at http://www.tau.ac.il/stoledo/taucs/。

[17] Alexa, M., 2003. **Differential coordinates for local mesh morphing and deformation.** *Visual Computer*, 19(2):p.105-114.

[18] Sheffer, A., Kraevoy, V., 2004. **Pyramid Coordinates for Morphing and Deformation.** In *Proceedings 2nd International Symposium on 3D Data Processing, Visualization, and Transmission,* p.68-75.

[19] Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D., 2004. **Modeling by example**. *ACM Transactions Graph.*, **23**(3):p.652-663.

[20] Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y., 2004. **Mesh editing with Poisson-based gradient field manipulation.** *ACM Transactions Graph.*, **23**(3):p.644-651.

[21] Biermann, H., Martin, I., Bernardini, F., Zorin, D., 2002. **Cut-and-Paste Editing of Multiresolution Surfaces.** *In Proceedings 29th Annual Conference on Computer Graphics and Interactive Techniques,* p.312-321.

[22] Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D., 2005. **A sketch-based interface for detail-preserving mesh editing.** *ACM Trans. Graph.*, **24**(3):p.1142-1147.

[23] Zimmermann, J., Nealen, A., Alexa, M., 2007. **SilSketch: Automated Sketch-Based Editing of Surface Meshes.** *In Proceedings 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling,* p.496-503.

[24] Zorin, D., Schroder, P., and Sweldens, W. 1997. **Interactive multi-resolution mesh editing.** *In Proceedings of SIGGRAPH 1997,* p.259–268.

[25] Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P., 1998. **Interactive Multi-Resolution Modeling on Arbitrary Meshes.** *In Proceedings 25th Annual Conference on Computer Graphics and Interactive Techniques,* p.105-114.

[26] Guskov, I., Sweldens, W., Schroder, P., 1999. **Multiresolution Signal Processing for Meshes.** *In Proceedings 26th Annual Conference on Computer Graphics and Interactive Techniques,* p.325-334.

[27] Bostch, M., and Kobelt, L. 2003. **Multiresolution surface representation based on displacement volumes.** *Computer Graphics Forum 22*, 3, p.483–492.

[28] Sederberg, T. W., and Parry, S. R. 1986. **Free-form deformation of solid geometric models.** *In Proceedings of SIGGRAPH 1986*, p. 151–160.

[29] Singh, K., and Fiume, E. 1998. **Wires: a geometric deformation technique.** *In Proceedings of SIGGRAPH 1998*, p. 405–414.

[30] Hsu, W. M., Hughes, J. F., and Kaufman, H. 1992. **Direct manipulation of free-form deformations.** *In Proceedings of SIGGRAPH 1992,* p. 177–184.