國立臺灣科技大學
資訊管理系
碩士學位論文

學號：M9109102

三維模型之對齊

3D Model Alignment

研 究 生：毛紹嘉

指導教授：楊傳凱 博士

中華民國 九十三 年 七 月 二 日

中文摘要

愈來愈多製作三維模型的工具以及掃描裝置被開發出來，使得製作三維模型較以前容易，且因為網路的蓬勃發展，現今在網路上可以很方便、迅速地找到三維模型，且因為 3D 加速卡及 CPU 的速度越來越快，硬體效能的提升使得複雜的 3D 計算可以快速地完成，因此個人電腦在三維模型的應用上也越來越廣泛。本篇論文研究如何將一個經過平移、縮放、旋轉的三維模型還原成原本的三維模型，這方法可以比對就人視覺上是相同的，但大小、位置、角度不一樣的兩個三維模型，並且可以作為比對三維模型的最終校正，也可對抗用平移、縮放、旋轉來攻擊加了浮水印的三維模型的攻擊方式。

我們利用重心與模型間的關係來解決縮放與平移的問題。並且用找出模型的最長軸、次長軸及找出模型中周圍三角形面積和前四大的頂點兩種方式，來找出模型的特徵軸，將原模型與新模型調成一樣的方向，接著將新模型倒著做回原模型所經調正的動作就可將新三維模型對齊原本的三維模型。

目錄索引

圖表索引

第一章　序論

第一節　背景與動機

本篇論文在研究如何將一個經過平移、縮放、旋轉的三維模型還原成原本的三維模型(圖 1-1)。這方法可以比對就人視覺上是相同的，但大小、位置、角度不一樣的兩個三維模型，並且可以作為比對三維模型的最終校正，也可對抗用平移、縮放、旋轉來攻擊加了浮水印的三維模型的攻擊方式。

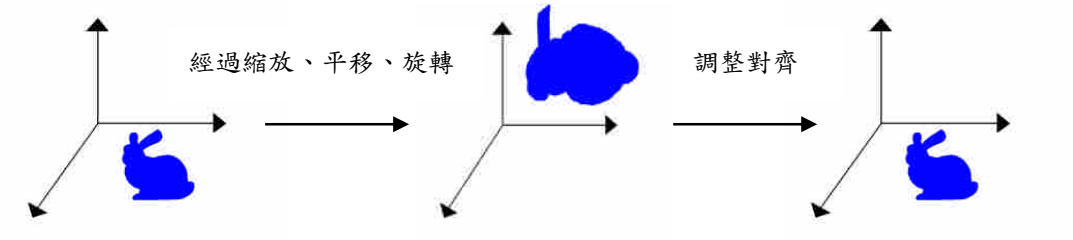


圖 1-1、三維模型對齊示意圖

由於製作三維模型的工具以及掃描裝置的進步，使得製作三維模型比較容易，並且由於網路的發展，現今在網路上可以很方便、迅速地找到三維模型，並且因為 3D 加速卡及 CPU 的速度越來越快，硬體效能的提升也使得複雜的 3D 計算可以快速地完成，因此個人電腦在三維模型的應用上也越來越廣泛。但由於多數的三維模型的格式主要的目的是供視覺化成像(rendering)，故只包含幾何上的座標位置及法向量等資訊，缺少明顯有用的資訊來做模型比對、對齊等動作，並且由於三維模型較平面影像多了一個維度，因此在搜尋方向角度、特徵比對上更加困難。

三維模型有許多特點和二維影像不同，舉例來說三維模型不需考慮光源、週遭物件(如鏡像)，一些在傳統二維影像上的問題也較不會在三維模型中出現。譬如我們可以預見的到，一個馬的三維模型就剛好有四支同樣大小的腳，但相反的二維影像中的馬有可能因為草原或其他景物擋住，使得馬匹的腿少於四支，或是因為影子的關係，使得影像中有多於四支腿的情形發生。

本篇論文在研究如何將一個經過平移、縮放、旋轉的三維模型還原成原本的三維模型。我們利用重心與模型間的關係來解決縮放與平移的問題，以找出模型之特徵軸的方式，將原模型與新模型調成一樣的方向，接著將新模型倒著做回原模型所經調正的動作就可將新三維模型對齊原本的三維模型。

第二節　論文架構

本論文第二章介紹與對齊模型有關的三維模型比對。第三章介紹我們經過平移、縮放、旋轉對齊新三維模型和原模型的方法。第四章介紹實驗的結果數據。第五章總結本論文並指出未來的可能研究方向。

第二章　相關研究

第一節　模型搜尋比對

本論文可以比對相同的兩個三維模型，並且可以用於比對搜尋三維模型方法的最後調正工作。因此我們在這邊先介紹幾種模型搜尋比對的方式。

隨著網路上三維模型的逐漸增加，故有需要研究出搜尋三維模型的功能，讓使用者能方便快速地找到三維模型，而搜尋比對三維模型的方法有許多是去比對兩個三維模型的特徵值來判斷兩者的相似度(圖 2-1)。

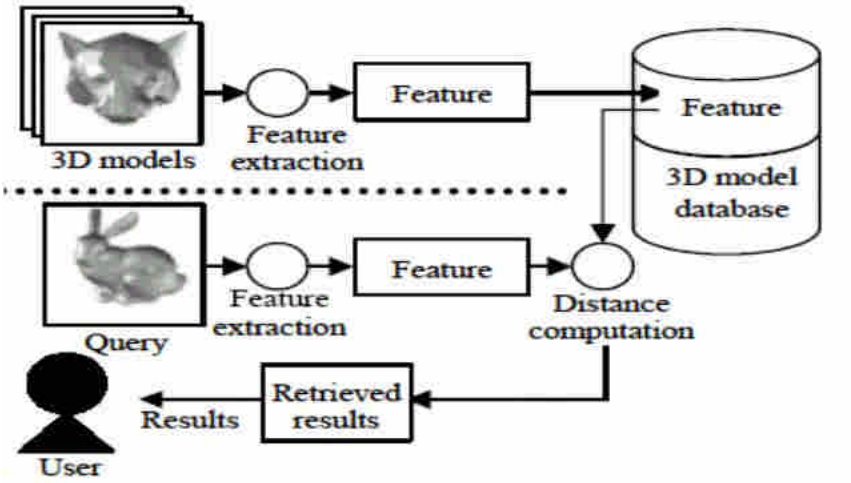


圖 2-1、搜尋比對三維模型示意圖

有許多論文是依三維模型之幾何分布的特性來搜尋三維模型，依幾何分佈的方法主要可分為依形狀搜尋和依拓樸搜尋。依形狀搜尋的方法主要是用頂點和多邊形的分佈來判斷三維模型的相似度[1] [2] [3] [4] [5] [6]，此類方法的難處是，如何決定有效的特徵來代表一個三維模型，並且這特徵必須具有以下特性:唯一不可取代的(unique)、 敏銳的(sensitive)(當模型的頂點等改變時，可以有效反應出模型的變動)、強健的(robust)(可以抵擋各式攻擊)、還有能在各種三維模型上運作。依拓樸搜尋的方法是利用三維模型的整體拓僕結構來分辨三維模型[7] 。此類方法的難處是找出一個方法可以從所有三維模型中取出拓樸結構，還有需要

辨別不同種類模型的拓樸結構。兩種方式都各有其優缺點：依拓樸搜尋的方式可以有效辨別出方向不同但本質上相同的兩個三維模型，但是依形狀搜尋的方式則對此比較難分辨；依形狀搜尋的方式可以有效分辨出兩個本質相同的三維模型，但若模型的部份連結不一樣，則依拓樸搜尋就一定不能分辨出來，例如兔子模型的耳朵跟腳都是兔子模型的一部分，若依拓樸搜尋的方式則無法分辨出兔子模型的腳有沒有連結到兔子模型的身體，但是依形狀搜尋的方式則可以分辨出。

第二節　Shape Distributions

很多比對搜尋三維模型的先前工作是在定義適當的特徵值之下來進行比對的動作[1]~[12]，Osada[2]發表了一種經過特殊設計的 shape function 來計算代表整個模型特性的分佈曲線(probability density functions, pdfs)，接著比對兩個模型曲線的差異以辨別兩個模型的異同。

Shape Distribution 是利用數種 shape function 來代表三維模型，並計算出三維模型在該 shape function 下的分佈曲線(probability density functions，pdfs 或 cumulative distribution functions, cdfs)，再利用數學方式比較兩個分佈曲線的分佈差異來判斷兩個三維模型是否相似(圖 2-2)。

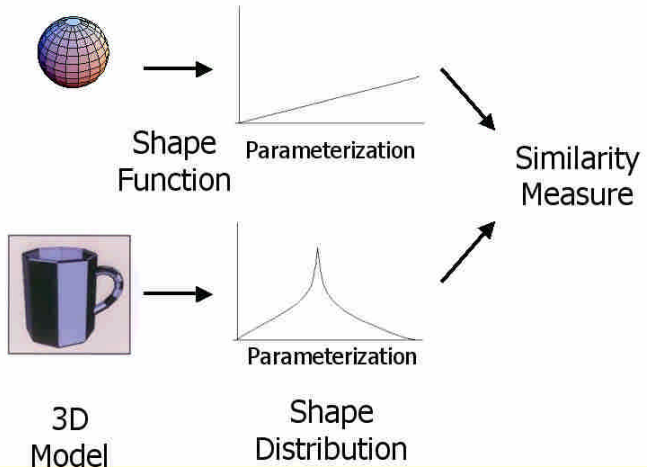


圖 2-2、Shape Distribution 示意圖

這方法的好處是簡單而且可以對抗很多對模型的攻擊，並且不須更動到模型本身就可完成。但是據蘇龍祥的論文[16]指出，shape distribution 雖然可以找出相

似度高的模型，但在分辨比較不相似的模型的情況下所算出來的 score 較無法作為相似程度的參考(也就是說假設 A 與 B 都不相似於 C，而且算出 $SCORE_A$、$SCORE_B$，但是即使 $SCORE_A > SCORE_B$，我們還是很難保證說 A 比 B 更像 C)。

Shape Distribution 的方法如下:

1、先選擇要使用的 shape function，shape function 的分類如下(圖 2-3):

A3: 模型上隨機選出的三個點所夾的角度。

D1: 模型上一固定點和模型上數個隨機選出的點的距離。

D2: 模型上隨機選擇的點之間的距離。

D3: 模型上隨機選擇三個點的面積。

D4: 模型上隨機選擇四個點的體積。

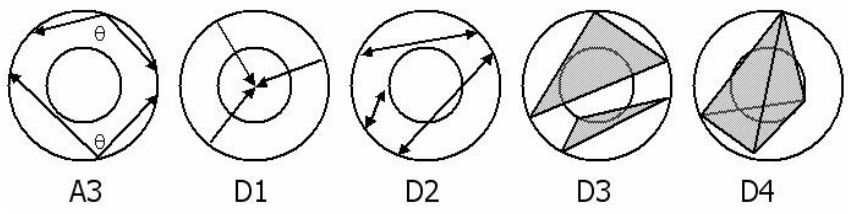


圖 2-3、Shape Function 示意圖

2、兩個模型各隨機選出相同各數的點，假設選出來的隨機點計算出 n 組函數值，分別將兩個模型的 n 組函數值作成直方圖分佈(histogram)(設函數值分佈在 0–100，分隔成相同的區間，計算落在各區間的函數值各數有幾個，例如，分佈在 0–10 有 K_1 個，10–20 有 K_2 個．．．90–100 有 K_{10} 個，$K_1 + K_2 + ... + K_{10} = n$)。將直方圖的頂端連接成曲線圖，再將此曲線以函數型態表示，這個函數就是模型在此 shape function 得出的結果(如圖 2-4、2-5)。

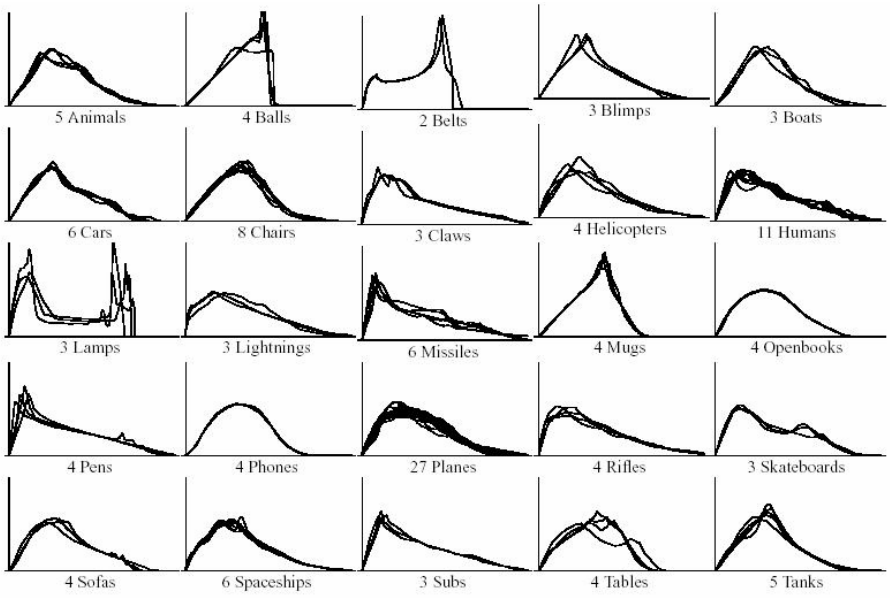


圖 2-4、25 類共 133 個模型在 D2 shape function 下的 shape distribution

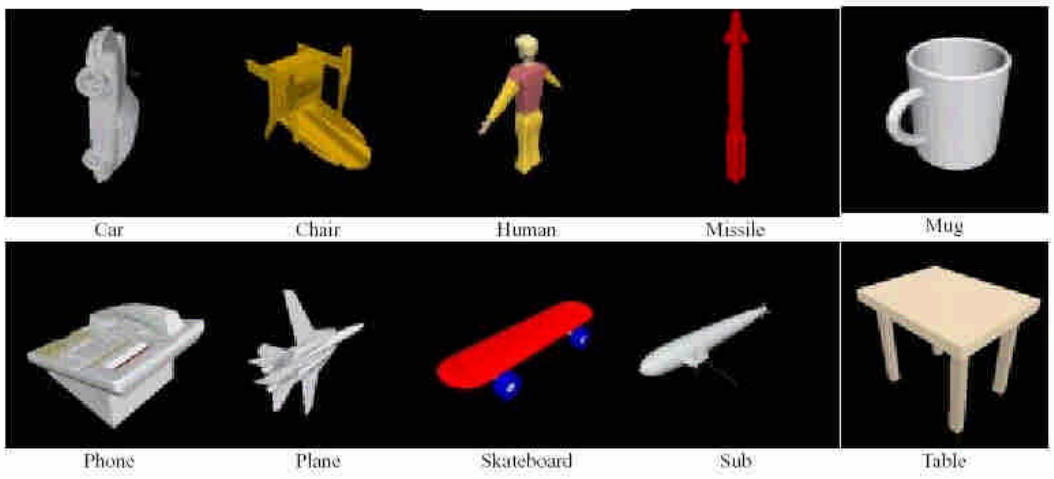


圖 2-5、各種類模型的例子

3、算出兩個模型 distribution 的差異，Osada 提出 Minkowski L_N norm 的方式來判斷差異:

PDF L_N : Minkowski L_N norm of the pdf:

$$D(f,g)=(\int|f-g|^N)^{1/N}$$

CDF L_N : Minkowski L_N norm of the cdf:

$$D(f,g)=(\int|\hat{f}-\hat{g}|^N)^{1/N}$$

式中 f 與 g 代表兩個模型的 Probability Density Function(PDF)，$\hat{f}$ 、 $\hat{g}$ 代

表 Cumulative Density Function(CDF)，例如 $\hat{f}(x) = \int_{-\infty}^{x} f$ 。利用以上公式可比對兩個模型 distribution 的差異值。

第三節　以視覺相似度對齊三維模型

台灣大學資訊工程學研究所的歐陽明教授與陳鼎匀博士提出了以視覺相似度為基礎的三維模型對齊與搜尋系統。他們的方法是根據三維模型在視覺上的相似度來決定兩個三維模型的相似度。在解決旋轉問題方面，他們從各個角度投射三維模型產生二維的影像，再一一比對從哪個角度會得到最相近的結果，並用三維模型對齊的技術來比對三維模型的相似度，以達到搜尋的功能。

他們對齊三維模型的過程如下:

$TS \rightarrow R_C \rightarrow TS \rightarrow R_r \rightarrow TS$

TS 代表解決平移(Translation)和縮放(Scaling)問題，R_C 代表粗略(coarser)的對齊旋轉方向，R_r 代表比較精細(refined)的對齊旋轉方向，圖 2-6 解釋整個過程。

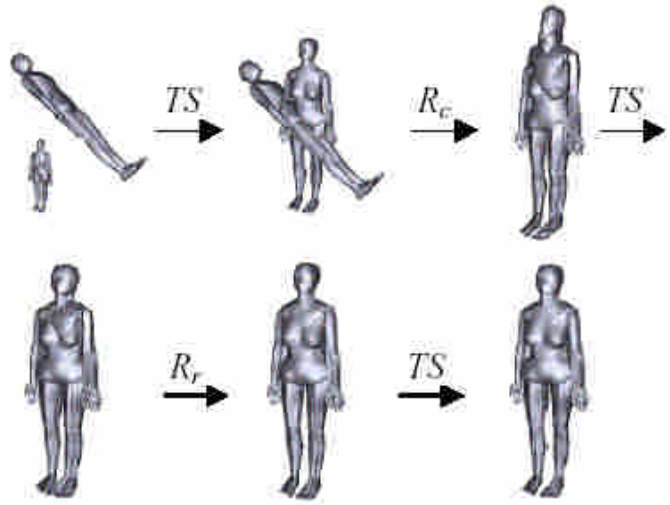


圖 2-6、三維模型對齊過程

在解決平移方面，他們是用三維模型的中心來當原點，公式如下:

$T = (T_x, T_y, T_z)$

$$T_i = \frac{MaxCoor_i + MinCoor_i}{2}, i = x, y, z$$

$MaxCoor_i$、$MinCoor_i$ 分別代表在 i 軸上有最大值和最小值的頂點座標。

在解決縮放方面，他們用三維模型 x、y、z 軸上的最大值來作正規化的動作，公式如下:

$$S = \frac{1}{\max_{i=x,y,z}(MaxCoor_i - MinCoor_i)}$$

$MaxCoor_i$、$MinCoor_i$ 所代表的意義同上。

在解決旋轉方面，他們是從一固定角度去看其中一個三維模型，得出一組影像，接下來在另一個三維模型上找出一個觀察角度，使得其所得出的影像與之前固定角度下誤差最小，如此即可從兩個觀察角度中得出該如何對齊旋轉(圖 2-7 說明整個過程)。

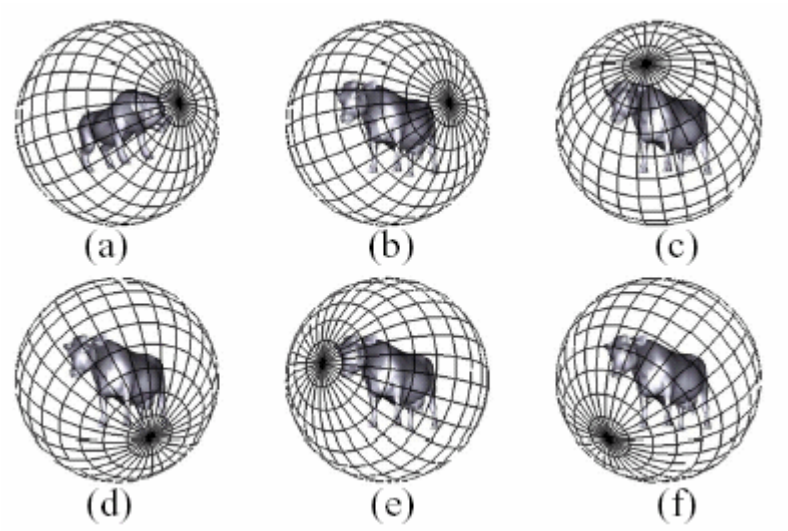


圖 2-7、以視覺相似度解決旋轉問題

圖 2-7 中有兩個模型，分別是豬的模型(a)與牛的模型(b)，兩者在位置、大小上都概略地接近，但其旋轉方向不同。要將牛的模型對齊豬的模型，可先從一個角度觀察豬的模型，得到一組影像，再由各個角度觀察牛的模型，並比對此角度得出的影像與豬的模型的影像，我們可以發現從(e)角度觀察可以得出最小誤差，再由(a)與(e)中間算出觀察角度的旋轉關係，如此即可將兩者對齊。

前兩個 *TS* 是讓兩個模型能概略地在位置和大小上相似，並讓接下來的 R_C、R_r 較易算出，當最後旋轉方面完全對齊，就可以得出正確的平移位置及縮放大

小。

最後以公式流程介紹此篇論文如何將一個模型(B)對齊另一個模型(A)：

(1) 先個別調整位置與大小，分別得到 A' 與 B'，使得模型 A' 與模型 B' 在位置和大小上概略相似。

$$A' = A \cdot T_A \cdot S_A$$

$$B' = B \cdot T_B \cdot S_B$$

(2) 計算 B' 粗略(coarser)對齊 A' 旋轉方向的旋轉矩陣。

$$R_C = RotateCoarse(B', A')$$

(3) 將 B' 粗略旋轉對齊 A'，再調整位置與大小得到 B''，使得 B'' 更相似於 A'。

$$B'' = B' \cdot R_C \cdot T_B \cdot S_B$$

(4) 計算 B'' 精細(refined)對齊 A' 旋轉方向的旋轉矩陣。

$$R_r = Rotate\,\mathrm{Re}\,fine(B'', A')$$

(5) 將模型 B'' 精細旋轉對齊 A'，再調整位置與大小，如此模型 B'' 即對齊 A'。

$$A' \sim B'' \cdot R_r \cdot T_B \cdot S_B$$

(6) 將 A' 與 B'' 展開。

$$A \cdot T_A \cdot S_A \sim B' \cdot R_C \cdot T_B \cdot S_B \cdot R_r \cdot T_B \cdot S_B$$

(7) 將(6)右式乘上 $S_A^{-1} \cdot T_A^{-1}$，如此即可得知將 B 對齊 A 的公式

$$A \sim B \cdot T_B \cdot S_B \cdot R_C \cdot T_B \cdot S_B \cdot R_r \cdot T_B \cdot S_B \cdot S_A^{-1} \cdot T_A^{-1}$$

他們在 Pentium III 800MHz 上測試，在平均 5274.4 個頂點，10233.8 個三角型的測試模型中，粗略(coarser)對齊平均執行時間為 14.5 秒，精細(refined)對齊平均執行時間為 23.5 秒。

第三章　三維模型對齊

第一節　三維模型對齊方法大綱

在此章，將介紹我們如何對齊本質是一樣，但可能大小、位置、旋轉方向角度不一樣的兩個三維模型(如圖 3-1)。我們先調整新模型的位置，使新模型移回原點，接著調整模型的大小，讓新模型大小跟原模型一樣，最後調整旋轉角度(如圖 3-2)。以下將細部說明這三個動作。

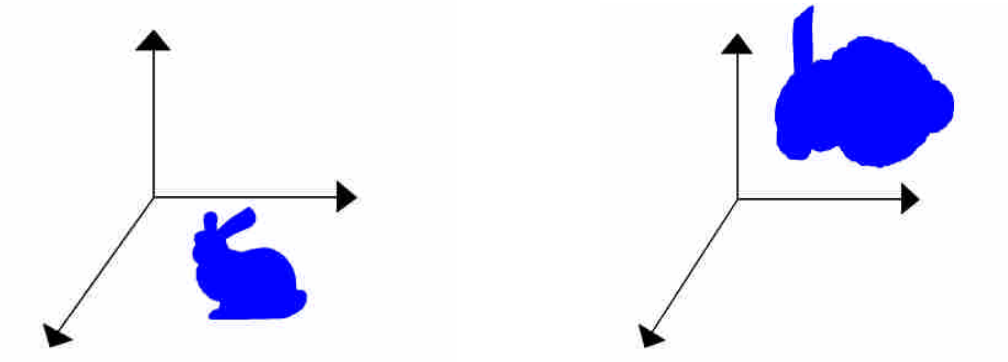


圖 3-1、兩個本質一樣，大小、位置、旋轉方向角度不一樣的三維模型

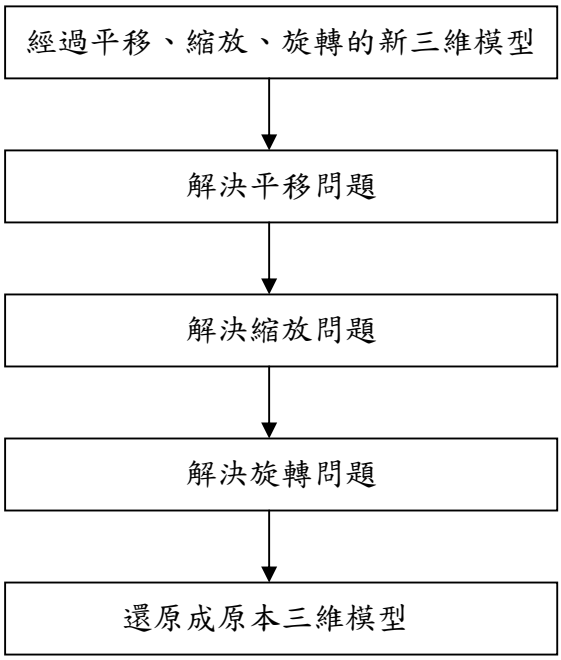


圖 3-2、對齊三維模型流程

第二節　調整位置

雖然新模型的每一點經過平移後都可能與原來不同，但我們可以先找出原模型與新模型的重心，將新模型的重心移回到原模型的重心上(如圖 3-3)，並且新模型的每一點都移回同樣的距離，如此就可將新模型調整回原模型的位置。

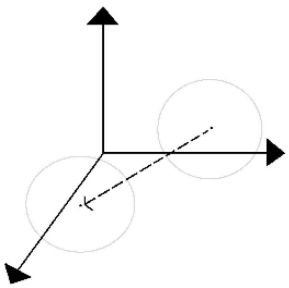


圖 3-3、調整位置示意圖

步驟為先將原模型與平移後的新模型的重心算出來，計算出兩者在三個維度的差，將新模型的每個點都減去這個差值，計算方式如下:

計算原模型的重心: $C_x = \frac{\sum_{i=1}^{n} X_i}{n}$　$C_y = \frac{\sum_{i=1}^{n} Y_i}{n}$　$C_z = \frac{\sum_{i=1}^{n} Z_i}{n}$

計算新模型的重心: $C_x' = \frac{\sum_{i=1}^{n} X_i'}{n}$　$C_y' = \frac{\sum_{i=1}^{n} Y_i'}{n}$　$C_z' = \frac{\sum_{i=1}^{n} Z_i'}{n}$

計算原模型與新模型重心差: $D_x = C_x' - C_x$　$D_y = C_y' - C_y$　$D_z = C_z' - C_z$

將模型各點減去差值: $P_x' = P_x - D_x$　$P_y' = P_y - D_y$　$P_z' = P_z - D_z$

C_x、C_y、C_z 分別是原模型重心 x、y、z 三個維度的值，C_x'、C_y'、C_z' 分別是新模型重心 x、y、z 三個維度的值，n 為模型頂點個數，X_i、Y_i、Z_i 分別是原模型各頂點 x、y、z 三個維度的值，X_i'、Y_i'、Z_i' 分別是新模型各頂點 x、y、z 三個維度的值，D_x、D_y、D_z 分別是原模型與新模型重心差 x、y、z 三個維度的值，P_x、P_y、P_z 分別是原模型各頂點 x、y、z 三個維度的值，P_x'、P_y'、P_z' 分別是新模型各頂點 x、y、z 三個維度的值。

第三節　調整大小

概念是找出模型被縮放的大小比例值，將被縮放的新模型每一點都除以此比例值即可解決。步驟為分別算出新模型重心到所有點的距離和、原模型重心到所有點的距離和(如圖 3-4)，將兩者相除得到一比例值，把新模型的每個點都除以此比例值，即可將新模型還原成原模型的大小。

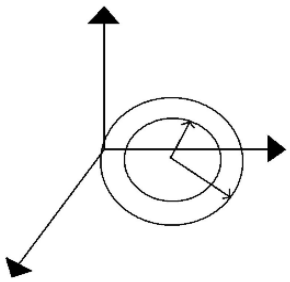


圖 3-4、調整大小示意圖

計算方式如下:

計算原模型重心到所有點距離和:

$$S=\sum_{i=1}^{n}\sqrt{(X_i-C_x)^2+(Y_i-C_y)^2+(Z_i-C_z)^2}$$

計算新模型重心到所有點距離和:

$$S'=\sum_{i=1}^{n}\sqrt{(X_i'-C_x')^2+(Y_i'-C_y')^2+(Z_i'-C_z')^2}$$

X_i、Y_i、Z_i為原模型各點 x、y、z 三個維度座標值，C_x、C_y、C_z為原模型重心三個維度座標值，X_i'、Y_i'、Z_i'為新模型各點 x、y、z 三個維度座標值，C_x'、C_y'、C_z'為新模型重心三個維度的座標值。

第四節　調整旋轉角度

在調整旋轉角度方面我們的構想是，將旋轉後的原模型與新模型都旋轉成一樣的方向，再將新三維模型倒著做回原模型所經調成此方向的動作就可將新三維

模型還原、對齊為原本的三維模型，此流程如下(如圖 3-5)。

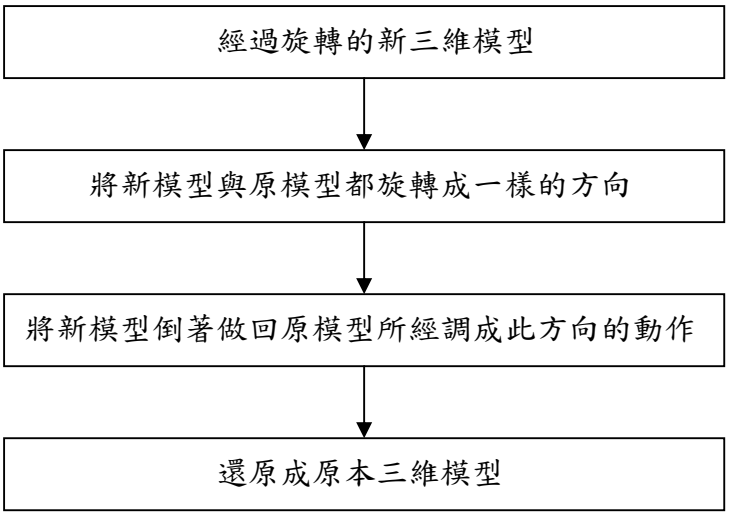


圖 3-5、調整旋轉問題流程

將原模型與新模型都調正成一固定方向可以用以下公式來表示:

$$R_y R_z R_x OriginModel = R_y^{'} R_z^{'} R_x^{'} NewModel$$

我們的目的是要讓兩邊的模型乘上這些旋轉矩陣後得到一樣的方向，也就是變成一樣的模型，R_x 、R_y 、R_z 為將原模型繞著 x、y、z 軸方向旋轉某一角度的旋轉矩陣， $R_x^{'}$ 、$R_y^{'}$ 、$R_z^{'}$ 為將新模型繞著 x、y、z 軸方向旋轉某一角度的旋轉矩陣。

接著將新三維模型倒著做回原模型所經調正的動作就可將新三維模型對齊原本的三維模型:

$$OriginModel = R_x^{-1} R_z^{-1} R_y^{-1} R_y^{'} R_z^{'} R_x^{'} NewModel$$

R_x^{-1} 、R_y^{-1} 、R_z^{-1} 分別為 R_x 、R_y 、R_z 等旋轉矩陣之反矩陣，$R_x^{'}$ 、$R_y^{'}$ 、$R_z^{'}$ 為將新模型繞著 x、y、z 軸方向旋轉某一角度的旋轉矩陣。

要將原模型與新模型轉成同樣的方向，我們可以利用兩者在幾何(geometry)上的相同特性來達成，雖然新的三維模型因為旋轉後點的座標與原本的三維模型不同，但有些幾何上的特性是不變的，譬如雖然經過旋轉，但新的三維模型上的

最長兩點距離與原本的三維模型上的最長兩點距離是一樣的，所構成的點也是一樣的(如圖 3-6)；同理，次長距離也是一樣的，而在原本三維模型上周圍三角形面積和最大的點在旋轉後的新三維模型上也仍然是周圍三角形面積和最大的點。我們可以利用這些特性在原模型與新模型上找出相同的線來解決旋轉問題。

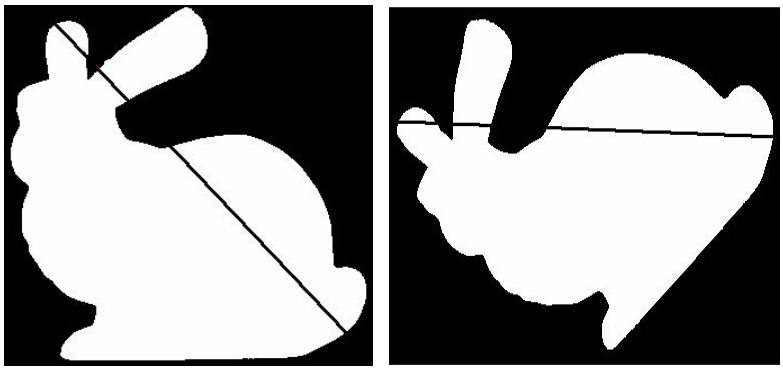


圖 3-6、最長兩點距離不受旋轉影響

我們在調整旋轉角度上試驗了兩種方法，第一種是分別找出在原模型與新模型上的最長軸與次長軸，再依此進行調正的動作。第二種是分別找出原模型與新模型上周圍三角形面積和前四大的四個頂點，取第一大與第二大兩個點形成一個軸，第三大與第四大兩個點形成另一軸，再去進行調正的動作。以下我們分三項進行論述，第一項介紹找出最長軸次長軸的方法，第二項介紹用點的周圍三角形面積和來型成兩個軸的方法，第三項介紹調正的動作。

第一項　找出最長軸、次長軸

我們先用逐點比對的方法來找出三維模型中的最長軸與次長軸，譬如，假如現在有 n 個點，我們逐點去計算兩點距離，計算點 1 跟點 2、點 1 跟點 3・・・到點 1 跟點 n，然後計算點 2 跟點 3、點 2 跟點 4・・・以此類推一直計算到點 n 跟點 n，事實上找出最長軸、次長軸有更快的方法，我們將在第五章討論，而用最長軸、次長軸來對齊模型的數據結果將在第四章討論，以下是逐點比對找最長軸、次長軸的演算法(圖 3-7)。

```
Model_PointNumbers : 三維模型的頂點各數
temp : 紀錄目前兩點距離
distance_1 : 紀錄最長距離
distance_2 : 紀錄次長距離

for(i=0; i<Model_PointNumbers; i++) {
      for(j=i+1; j<Model_PointNumbers; j++) {
            temp = the distance between points I and j;

            if(temp > distance_1) {
                distance_1 = temp;
                Record i 、 j two points to form the longest axis; }

            else if(temp > distance_2) {
                distance_2 = temp;
                Record i 、 j two points to form the second long axis; }
      }
}
```

圖 3-7、逐點找最長軸、次長軸演算法

第二項　利用點的周圍三角形面積和

其想法是因為旋轉不會影響到模型上三角形的面積,所以在原模型上周圍三角形面積和最大的頂點在旋轉後的新三維模型上仍然應是周圍三角形面積和最大的點。我們可以分別找出原模型與新模型上周圍三角形面積和前四大的點，以第一大跟第二大兩點形成一個軸，第三大跟第四大兩點形成第二個軸，再去進行調正的動作。因為三角形任兩邊向量作外積得到的值再開根號除以二就是此三角形的面積，我們用此公式來計算三角形的面積，如下:

a 、b 、c 此三角形三個頂點，V_1 、V_2 為構成三角形的兩個向量

$V_1 = b - a$

$V_2 = c - a$

$\frac{\sqrt{\| V_1 \times V_2 \|}}{2}$ =此三角形面積

而此法的演算法如下(圖 3-8):

Model_PointNumbers : 三維模型的頂點各數
triangle[i] : 用來紀錄哪些三角形用到此點的鏈結串列
V_1 : 設構成三角形的三個點為 a、b、c，則V_1為 b-a 所形成的向量
V_2 : 設構成三角形的三個點為 a、b、c，則V_2為 c-a 所形成的向量
area : 紀錄三角形面積和
area_1 : 紀錄第一大三角形面積和
area_2 : 紀錄第二大三角形面積和
area_3 : 紀錄第三大三角形面積和
area_4 : 紀錄第四大三角形面積和

```
for(i=0; i<Model_PointNumbers; i++) {
        Pointer = triangle[i].Head->Next;
        area = 0;

        while(Pointer != NULL) {
            Find Next V1 and V2 (V1 and V2 are two vectors forming a
            triangle adjacent to vertex i);
            area += sqrt(|| V1 × V2 ||) / 2;
            Pointer = Pointer->Next;
        }
        if(area > area_1)
          Record i as the biggest area point and it's area;
        else if (area > area_2)
          Record i as the second area point and it's area;
        else if (area > area_3)
          Record i as the third area point and it's area;
        else if (area > area_4)
          Record i as the fourth area point and it's area;
}
```

圖 3-8、找出周圍三角形面積和前四大點

第三項　對齊模型

我們依前述的兩種方法分別在新模型上找出兩條軸，原模型上也找出兩條

軸，利用這兩條軸來固定原模型與新模型，使他們旋轉成一樣的方向。為此，我們先將一條軸(之後用軸 A 來稱呼)移到原點，再讓它與 z 軸對齊，如此一來，模型變動的情形就只會繞著 z 軸旋轉，所以我們再讓另一條軸(之後用軸 B 來稱呼)平行於 xz 平面，如此就可將模型固定成一個特定的方向。

流程如圖 3-9，底下我們細部解釋各個動作。

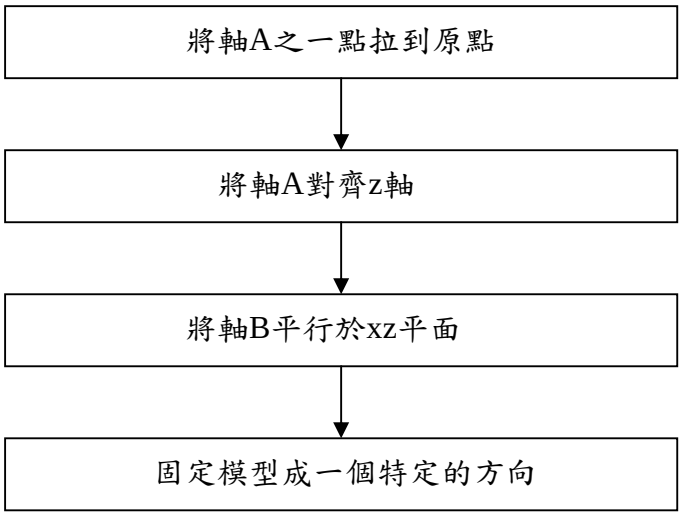


圖 3-9、對齊模型流程圖

(1) 將軸 A 對齊 z 軸的細節如下:

我們先將軸 A 繞著 x 軸旋轉，使得軸 A 躺平在 xz 平面上，繞著 x 軸旋轉的旋轉矩陣如下:

$$R_x = R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

要算出旋轉的角度θ，我們可以想成將軸 A 投射到 yz 平面，在 yz 平面上對 x 軸旋轉(如圖 3-10):

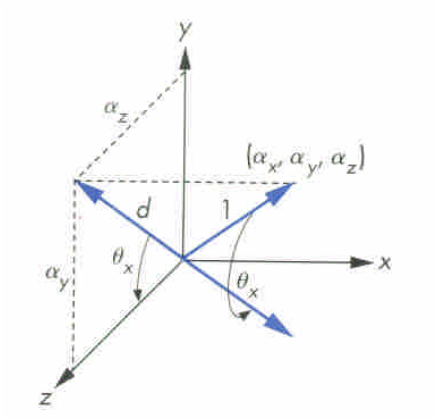


圖 3-10、對 x 軸旋轉

如此繞著 x 軸旋轉的旋轉矩陣為:

$$R_x = R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha_z / d & -\alpha_y / d & 0 \\ 0 & \alpha_y / d & \alpha_z / d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中 $d = \sqrt{\alpha_y^2 + \alpha_z^2}$

我們將模型的每一點都乘上 R_x，如此就可將軸 A 躺平在 xz 平面。接著我們將軸 A 繞著 y 軸旋轉(如圖 3-11):

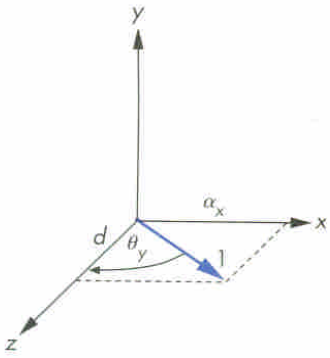


圖 3-11、對 y 軸旋轉

使軸 A 對齊 z 軸，繞著 y 軸旋轉的旋轉矩陣為:

$$R_y = R_y(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

在運算中，R_y 為:

$$R_y = R_y(\theta_y) = \begin{bmatrix} d & 0 & -\alpha_x & 0 \\ 0 & 1 & 0 & 0 \\ \alpha_x & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

繼續將模型的每一點都乘上 R_y，如此就可將軸 A 對齊 z 軸。

(2) 將軸 B 平行於 xz 平面細節如下:

要將軸 B 平行於 xz 平面，我們將軸 B 對 z 軸旋轉，對 z 軸旋轉的旋轉矩陣為:

$$R_z = R_z(\theta_z) = \begin{bmatrix} \cos\theta_z & 0 & \sin\theta_z & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_z & 0 & \cos\theta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

在運算中，R_z 為:

$$R_z = R_z(\theta_z) = \begin{bmatrix} \alpha_x / d & 0 & \alpha_y / d & 0 \\ 0 & 1 & 0 & 0 \\ -\alpha_y / d & 0 & \alpha_x / d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

將模型的每一點都乘上 R_z，如此就可將軸 B 平行於 xz 平面。

依以上步驟，原模型與新模型都用一樣的軸做一樣的旋轉動作，我們就可將原模型與新模型都固定成相同的方向:

$$R_y R_z R_x OriginModel = R_y^{'} R_z^{'} R_x^{'} NewModel$$

接著將新三維模型倒著做回原模型所經調正的動作就可將新三維模型對齊原本的三維模型:

$$OriginModel = R_x^{-1} R_z^{-1} R_y^{-1} R_y^{'} R_z^{'} R_x^{'} NewModel$$

第四章 實驗結果

第一節　實驗環境

我們的實驗平台與軟體如下表所示(表 4-1):

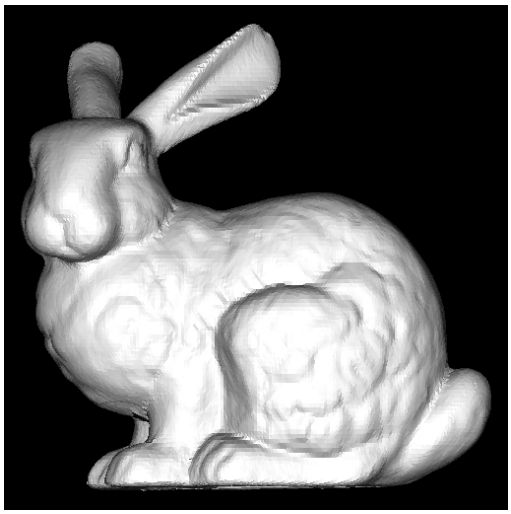
中央處理器	Pentium IV 2.4G
主記憶體	512MB
作業系統	Windows XP
軟體	Visual C++ 6.0

表 4-1、實驗環境

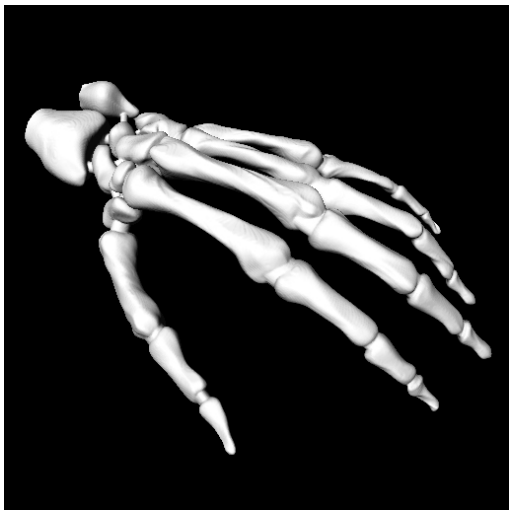
我們處理的三維模型為 PLY 格式，在 PLY 檔案前面有標頭紀錄此模型的頂點個數與三角形個數，並說明頂點由哪些資訊組成，如：點的座標、法向量等。在標頭後就記載每個點的座標、法向量等及每個面由哪些頂點構成。

第二節　實驗方法

當模型讀進來後，我們用亂數決定旋轉角度，並用亂數決定依 X 軸、Y 軸、Z 軸各多少來旋轉模型，接著用亂數決定放大縮小及平移。此新模型在旋轉角度、大小比例、平移位置皆與原模型不同，我們利用第三章所介紹的方法依序解決平移位置、大小比例、旋轉角度三個問題，將新模型還原對齊為原本的模型。此論文中，我們將列出測試了 20 個模型後的實驗數據。圖 4-1 是部分測試模型的圖。

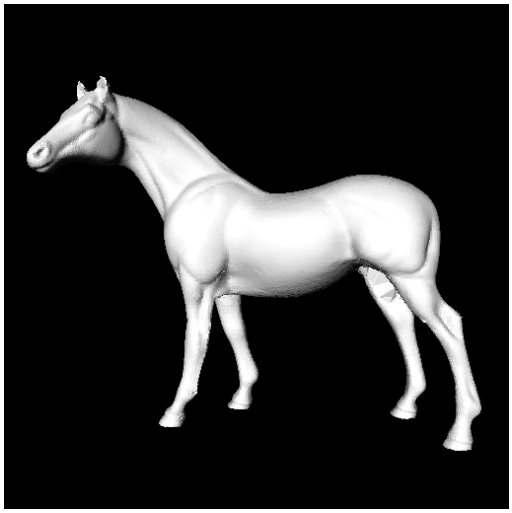


Bunny

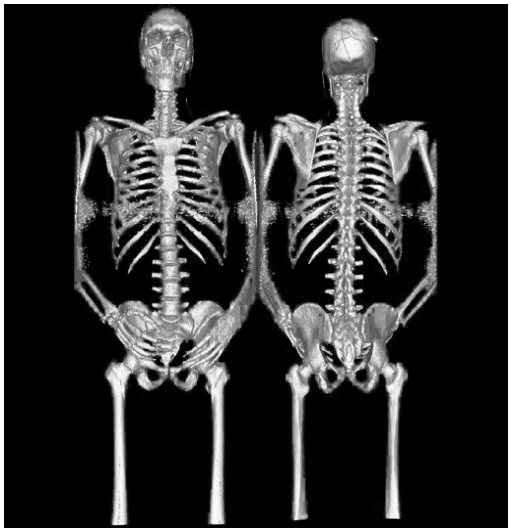


Skeleton Hand

Dragon



Horse



Visible Man Bone

Happy Buddha

圖 4-1、部分測試模型

第三節　調整平移實驗數據

Name	Bunny	Hand	Dragon	Bone0	Bone1	Bone2
Vertex	35947	327323	437645	4090	41217	177907
Face	69451	654666	871414	7990	81656	355511
Size(K)	2962	31878	33039	350	3689	16683
Average Time(S)	0.0014	0.01	0.018	0.0003	0.0015	0.005

Name	Bone3	Bone4	Bone5	Bone6	Bone7	Bone8
Vertex	43402	260464	45449	569636	14904	61904
Face	86210	522567	88497	1136745	29331	124018
Size(K)	3937	24927	4051	54562	1272	5568
Average Time(S)	0.0019	0.007	0.0015	0.023	0.0006	0.0028

Name	Skin0	Skin1	Skin2	Skin3	Skin4	Skin5
Vertex	10112	19106	34390	27237	23779	4445
Face	19642	37754	68467	53952	46734	8100
Size(K)	866	1680	3068	2463	2126	358
Average Time(S)	0.0004	0.0011	0.0041	0.0036	0.0021	0.0005

表 4-2、調整平移實驗數據

在表 4-2 中，Name 為模型名稱，Vertex 為模型頂點個數，Face 為模型三角形個數，Size 為模型大小，單位為 Kbytes，Average Time 為調整平移所需時間，所列為執行十次後所取的平均值，單位為秒。

因為在調整平移方面，我們是算出兩邊模型的重心差，再將平移後的新模型減去此差值，而此動作所需時間很短，所以我們可以發現每個模型完成調整平移的動作平均所需的時間都很短，幾乎都可在 0.01 秒內完成。

第四節　調整縮放實驗數據

Name	Bunny	Hand	Dragon	Bone0	Bone1	Bone2
Vertex	35947	327323	437645	4090	41217	177907
Face	69451	654666	871414	7990	81656	355511
Size(K)	2962	31878	33039	350	3689	16683
Average Time(S)	0.0059	0.06	0.072	0.0008	0.0065	0.023

Name	Bone3	Bone4	Bone5	Bone6	Bone7	Bone8
Vertex	43402	260464	45449	569636	14904	61904
Face	86210	522567	88497	1136745	29331	124018
Size(K)	3937	24927	4051	54562	1272	5568
Average Time(S)	0.0071	0.045	0.0072	0.092	0.0023	0.0126

Name	Skin0	Skin1	Skin2	Skin3	Skin4	Skin5
Vertex	10112	19106	34390	27237	23779	4445
Face	19642	37754	68467	53952	46734	8100
Size(K)	866	1680	3068	2463	2126	358
Average Time(S)	0.002	0.0023	0.0076	0.0035	0.0028	0.001

表 4-3、調整縮放實驗數據

在表 4-3 中，Name、Vertex、Face 及 Size 所代表之意義同前，Average Time 為調整縮放所需時間，所列為執行十次後所取的平均值，單位為秒。

在調整縮放方面，我們也只是算出兩邊模型重心到所有點距離和的差值再做調整，所以所需的計算時間也很少，幾乎都在 0.02 秒內。

第五節　調整旋轉實驗數據(使用最長軸、次長軸來調整)

Name	Bunny	Hand	Dragon	Bone0	Bone1	Bone2
Vertex	35947	327323	437645	4090	41217	177907
Face	69451	654666	871414	7990	81656	355511
Size(K)	2962	31878	33039	350	3689	16683
Average Time(S)	72.04	7886.67	8962.83	4.12	82.62	2519.68

Name	Bone3	Bone4	Bone5	Bone6	Bone7	Bone8
Vertex	43402	260464	45449	569636	14904	61904
Face	86210	522567	88497	1136745	29331	124018
Size(K)	3937	24927	4051	54562	1272	5568
Average Time(S)	65.48	5326.79	100.32	11725.18	29.86	129.02

Name	Skin0	Skin1	Skin2	Skin3	Skin4	Skin5
Vertex	10112	19106	34390	27237	23779	4445
Face	19642	37754	68467	53952	46734	8100
Size(K)	866	1680	3068	2463	2126	358
Average Time(S)	20.26	38.28	50.32	46.56	39.63	13.89

表 4-4、調整旋轉實驗數據(使用最長軸、次長軸來調整)

在表 4-4 中，Name、Vertex、Face 及 Size 所代表之意義同前，Average Time 為解決旋轉問題所需時間，所列為執行五次後所取的平均值，單位為秒。在此我們使用最長軸、次長軸來調整。

在找最長軸、次長軸時是係採用逐點比對的方式，其時間複雜度為$O(n^2)$，n 為模型頂點個數，因此可看到隨著模型的複雜度、頂點數的增加，所需的執行時間也增長。

第六節　調整旋轉實驗數據(利用點的周遭三角形面積和)

Name	Bunny	Hand	Dragon	Bone0	Bone1	Bone2
Vertex	35947	327323	437645	4090	41217	177907
Face	69451	654666	871414	7990	81656	355511
Size(K)	2962	31878	33039	350	3689	16683
Average Time(S)	0.18073	1.36102	2.41431	0.01804	0.18284	0.74732

Name	Bone3	Bone4	Bone5	Bone6	Bone7	Bone8
Vertex	43402	260464	45449	569636	14904	61904
Face	86210	522567	88497	1136745	29331	124018
Size(K)	3937	24927	4051	54562	1272	5568
Average Time(S)	0.18463	1.27886	0.19327	2.04153	0.05155	0.30347

Name	Skin0	Skin1	Skin2	Skin3	Skin4	Skin5
Vertex	10112	19106	34390	27237	23779	4445
Face	19642	37754	68467	53952	46734	8100
Size(K)	866	1680	3068	2463	2126	358
Average Time(S)	0.03515	0.07049	0.16990	0.13583	0.12123	0.01813

表 4-5、調整旋轉實驗數據(利用點的周遭三角形面積和)

在表 4-5 中，Name、Vertex、Face 及 Size 所代表之意義同前，Average Time 為解決旋轉問題所需時間，所列為執行五次後所取的平均值，單位為秒。在此我們利用點的周遭三角形面積和來調整。

在此我們可以發現利用頂點的周遭三角形的面積和來調整旋轉角度所需的時間非常短，幾乎都可在 0.1 秒內完成，此乃因為整個演算法的時間複雜度為 $O(n)$，比使用最長軸、次長軸的方式來調整快了許多。

第五章 結論

因為製作三維模型的工具以及掃描裝置的進步，使得製作三維模型比較容易，並且由於網路的發展，現今在網路上可以很方便、迅速地找到三維模型，並且由於 3D 加速卡及 CPU 的速度越來越快，硬體效能的提升亦使得複雜的 3D 計算可以快速地完成，因此個人電腦在三維模型的應用上也越來越廣泛。如何將一個被平移、縮放、旋轉的三維模型還原成原本的模型是本篇論文所欲探討的問題。

我們的方法是先算出原模型與新模型重心的差來調整新模型的位置，使新模型移回原點；接著找出原模型與新模型重心及其模型周圍頂點的距離總合之間的比率來調整模型的大小，讓新模型大小跟原模型完全一樣；最後找出模型的特徵軸，將原模型與新模型調成一樣的方向，接著將新模型倒著做回原模型所經調正的動作即可將新三維模型對齊為原本的三維模型，進而即可解決旋轉的問題。

未來可能試著用 Convex Hull[15]先找出模型最外圍頂點，再於在其中尋找最長軸次長軸，因為理論上最長軸應該是落在模型的最外圍，如此一來找最長軸次長軸的演算法就只需在最外圍的頂點上運算，以達到加速的效果。

我們也可試著用此篇論文來對付用平移、縮放、旋轉來攻擊加了浮水印之三維模型的攻擊方式。此乃因為現行許多在 spatial domain 加浮水印[13]的方式，如果將模型平移、縮放、旋轉則模型的頂點都與原本不同，自然浮水印也隨之改變，進而可能使著作權遭破壞。

我們也希望未來能處理當原模型與新模型不完全相同的情形，例如能判斷不同模型間的相似度，而將兩個模型以其最相似的方向對齊，凡此皆是未來可以繼續研究的方向。

參考文獻

[1] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin and D. Dobkin and D. Jacobs, “A Search Engine for 3D Models”, ACM Transactions on Graphics, 22(1):83-105, Jan. 2003.

[2] R. Osada, T. Funkhouser, B. Chazelle and D. Dobkin, “Shape Distributions”, ACM Transactions on Graphics, 21(4):807-832, Oct. 2002.

[3] E. Paquet, M.Rioux, A. Murching, T. Naveen and A. Tabatabai, “Description of Shape Information for 2-D and 3-D Objects”, Signal Processing: Image Communication, 16:103-122, Sept. 2000.

[4] R.Ohbuchi, T. Otagiri, M. Ibato and T. Takei, “Shape Similarity Search of Three-Dimensional Models Using Parameterized Statistics”, Proc. Of 10th Pacific graphics, 265-273, Beijing, China, Oct. 2002.

[5] D. V. Vranic and D. Saupe, “Description of 3D-Shape using a complex Function on the Sphere”, Proc. Of IEEE International Conference on Multimedia and Expo (ICME), 177-180, Lausanne, Switzerland, Aug. 2002.

[6] I. Kolonias, D. Tzovaras, S. Malassiotis and M. G. Strintzis, “Fast Content-Based Search of VRML Models based on Shape Distributions”, Proc. of International Conference on Image Processing (ICIP), 133-136, Thessaloniki, Greece, Oct. 2001.

[7] M. Hilaga, Y. Shinagawa, T. Kohnura and T. L. Kunii, “Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes”, Proc. of ACM SIGGRAPH. 203-212, Los Angeles, USA, Aug. 2001.

[8] B. K. P. Horn, “Extended Gaussian Images”, Proceedings of the IEEE, 72(12):1671-1686, 1984.

[9] M. Ankerst, G. Kastenmuller, H. P. Kriegel and T. Seidl, “3D Shape Histograms for Similarity Search and Classification in Spatial Databases”,

Proc. of 6th International Symposium on Advances in Spatial Databases(SSD), Hong Kong, China, 207-228, 1999.

[10] D. Y. Chen and M. Ouhyoung, "A 3D Object Retrieval System Based on Multi-Resolution Reeb Graph", Proc. of Computer Graphics Workshop, 16, Tainan, Taiwan, June 2002.

[11] S. Jeannin, L. Cieplinski, J. R. Ohm and M. Kim, "MPEG-7 Visual part of experimentation Model Version 7.0", ISO/IEC JCT1/SC29/WG11/N3521, Beijing, China, July 2000.

[12] M. Elad, A. Tal, and S. Ar. "Content Based Retrieval of VRML Objects – An Iterative and Interactive Approach", Proc. of 6th Eurographics Workshop on Multimedia, 97-108, Manchester UK, Sept. 2001.

[13] I. Cox andM. L.Miller, "A review of watermarking and the importance of perceptual modeling", in Proceedings of SPIE, Human Vision &Electronic ImagingII, vol. 3016, 92-99, 1997.

[14] Ding-Yun Chen and Ming Ouhyoung, "A 3D Model Alignment and Retrieval System", Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 1997.

[15] Convex Hull Algorithm,
http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html

[16] 蘇龍祥, "3D Model Retrieval Using Geodesic Distance", 國立成功大學資訊工程系碩士論文, 2001.