Introduction to Multimedia

Homework2 Report

102062209 邱政凱

Part 1 - (a)

題目目標是要求我們實作 Bezier Curve。根據我的理解，Bezier Curve 是一種遞迴式的插值法，n 階的 Bezier Curve 是由兩個 n-1 階的 Bezier Curve 插值而來。遞迴插值的結果讓 Bezier Curve 上每點的權重係數都可以表示成二項展開的係數。

因為題目規定 Control Point 取樣點是每四個為一組，所以可以直接套用 3 次二項展開的 4*4 係數矩陣 M= [-1 3 -3 1;3 -6 3 0;-3 3 0 0; 1 0 0 0]，然後用 linspace (0,1,LoD)產生依照不同 Level of Detail 而有不同取樣密度的插值點 t ，將之套用到 1*4 矩陣 T=[t^3 t^2 t 1]上，最後再將四個為一組的取樣點套入 4*1 矩陣 G = [p0 p1 p2 p3]^T 之上，將三個矩陣相乘(T*M)*G，對 x 和 y 分別各做一這樣的運算，然後用 plot 把結果畫在原圖上以方便比較。(另外值得一提的就是結尾點的處理，如果不特別處理的話有時候結尾點會沒辦法和起始點連在一起，所以要作 mod 運算。讓最後的 group 可以和起始的幾個點再做一次插值。)

首先我們先比較不同的取樣點數量(Sampling Rate)的結果：

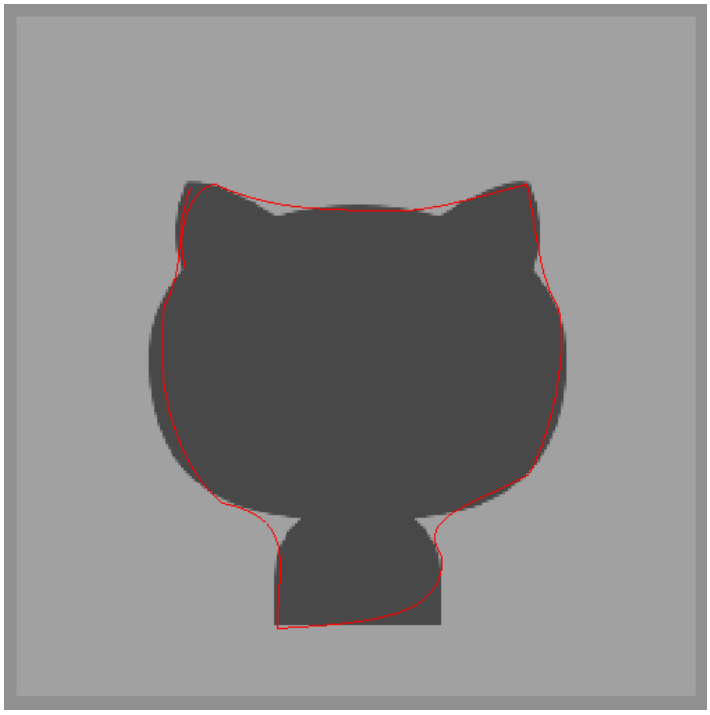


Figure 1 Sample rate 30

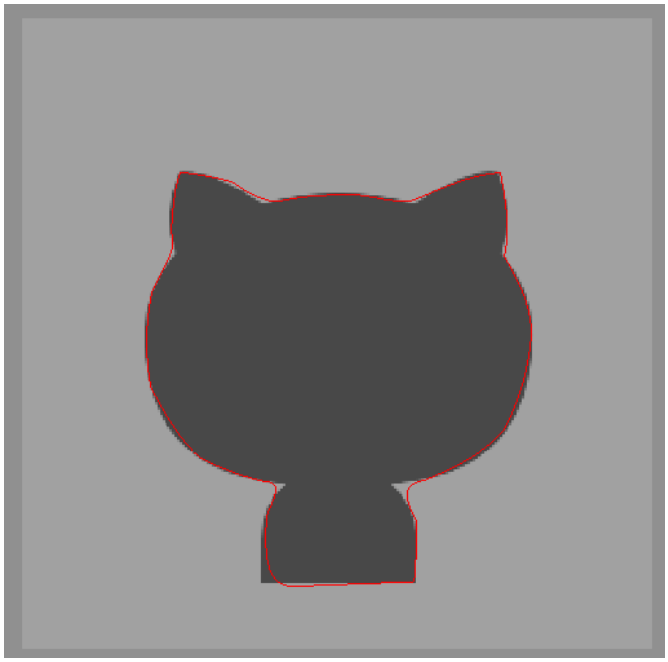


Figure 2 SampleRate60

上面的圖是 30 個 Sample 點的結果，下圖是 60 個 Sample 點的結果，可以看出上圖因為 Sample 點數量較少，沒有辦法把整個圖形的輪廓描繪的很正確，而下圖有兩倍的 Sample 點，對原本圖形的輪廓還原率是較正確的。

接下來讓我們看看同樣的 Sample 點數量，不同的 Level of Detail 的結果(在此定義 Level of Detail 為每個 4 個 Sample 點組成的 group 中，我們用幾個貝茲插值點來還原該曲線)：

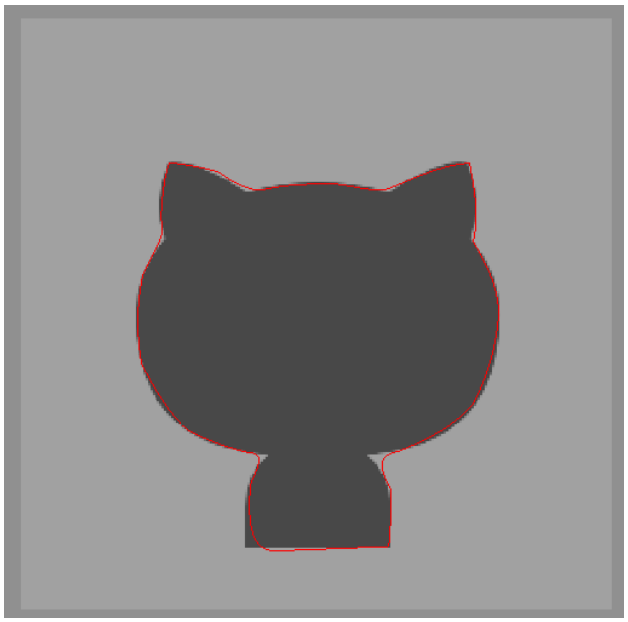


Figure 3 LOD 50

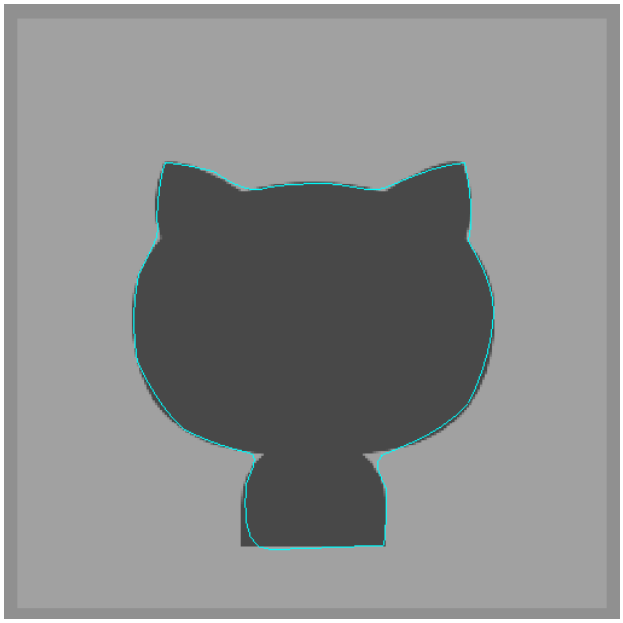


Figure 4 LOD6

上圖是 LOD 50 的結果，下圖是 LOD 6 的結果。

若沒有放大來看，會發現其實結果並沒有差很多(跟 Sample Rate 相比影響不那麼明顯)，但是如果放大來看：

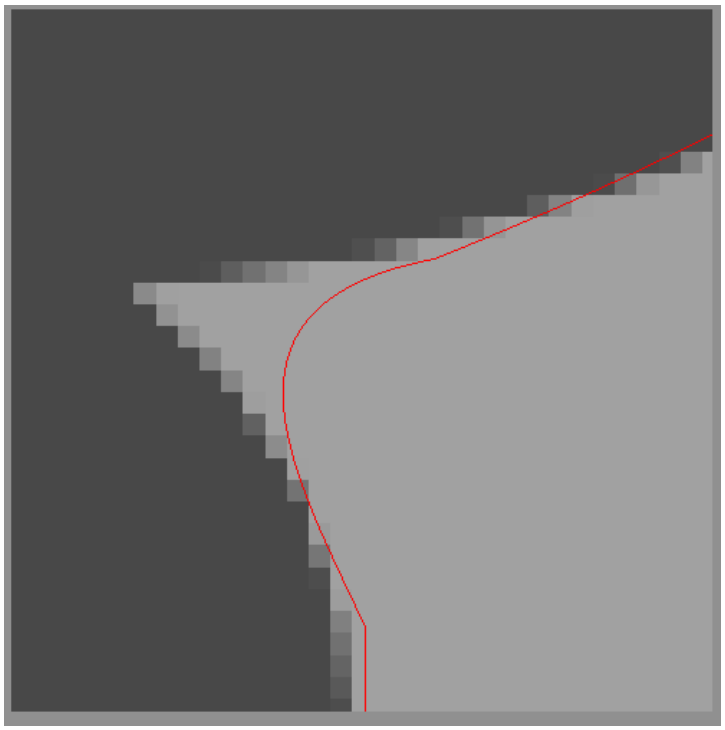


Figure 5 LOD 50 Zoom In

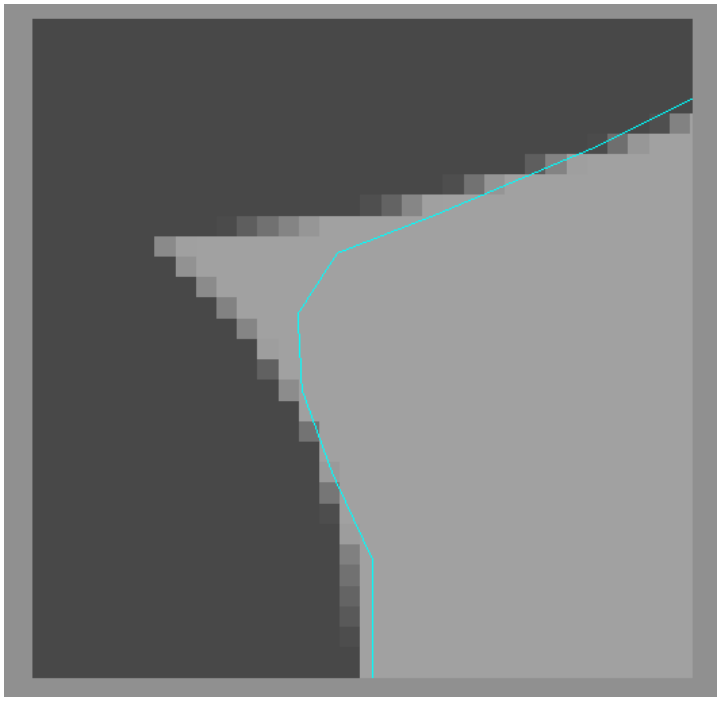
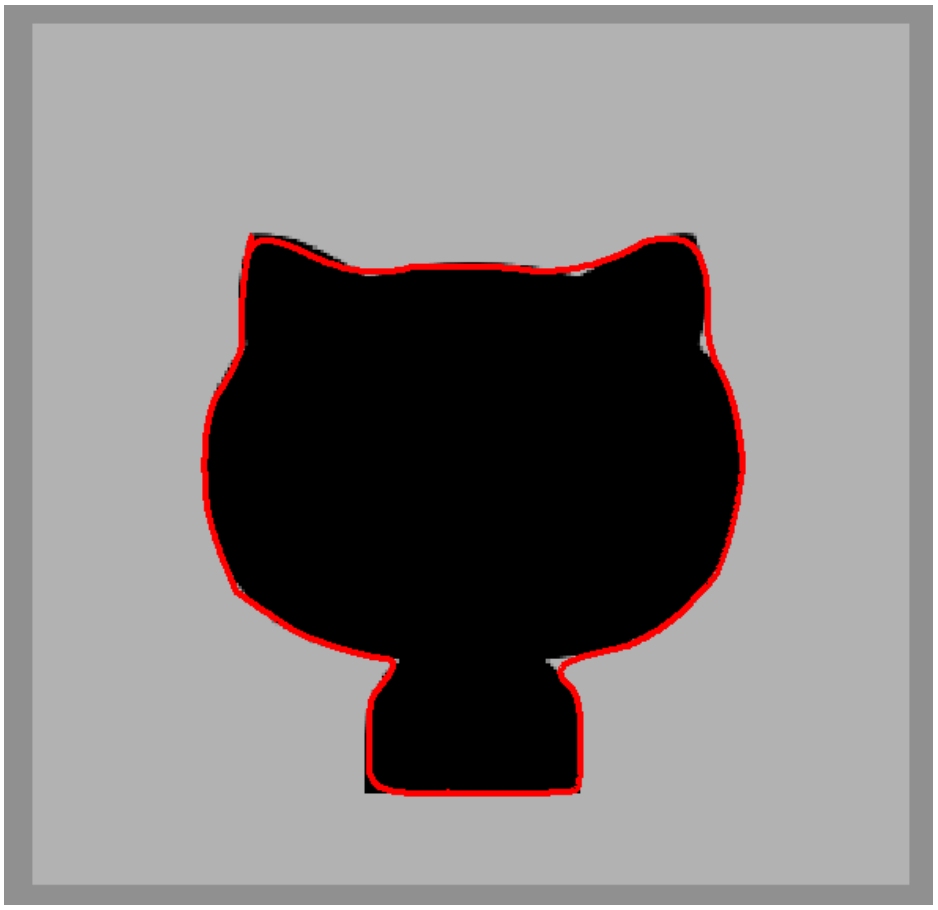


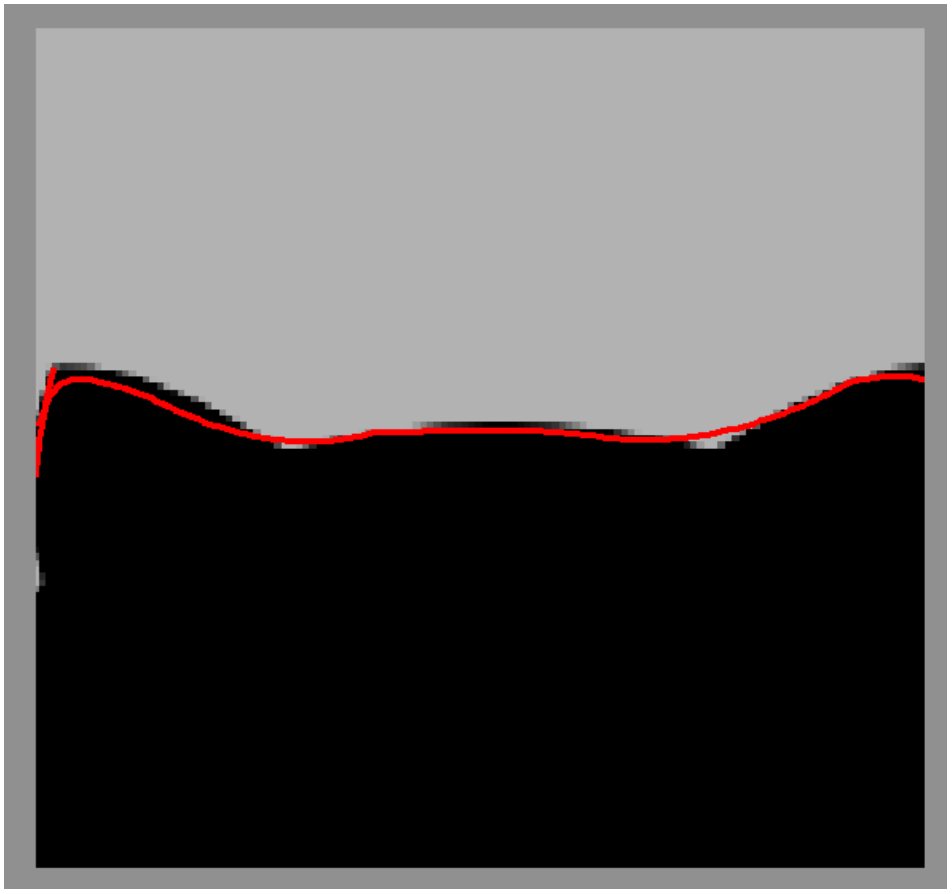
Figure 6 LOD 6 Zoom In

我們可以發現較高的 Level of Detail (插值點數量)形成的曲線是較為圓滑的。其實也是理所當然的結果，因為它的繪圖方式就是把每個點用直線連在一起，如果插值點多，每個線段間隔短，看起來就像是弧形，而插值點少，看來自然就像是一段一段的鋸齒而已。

Part 1 –(b)

這題叫我們比較Bitmap和Vector Graphic的放大效果。Bitmap使用的是跟上題一樣的圖片，我們直接呼叫`imresize(catImage,4,'nearest');` 把原圖放大四倍，並存成`bitmap4X.jpg`。而另一方面，呼叫 `temp4x = temp.*4` (temp為存放`Bezier Curve`算出來的點的矩陣)，來用向量方式把前一題得出來的圖形放大四倍。
比較兩者的方式我選擇先用`imshow`把放大四倍過後的原圖畫出來，再用紅線把放大四倍的`Recovered Object shape`畫出來 (見下圖)





其實，在上一題的部分，為了比較不同 LOD 的差異放大去觀察圖片就已經可以看到 Bitmap 放大時失真的情況。然而相比起來，我們直接用座標點當作描繪點的 BezierCurve 算是一種向量圖，我們直接把所有的點座標放大四倍，然後讓 matlab 的 plot 函數去把這些點連接在一起(這個過程類似向量圖的顯示過程)，可以看出來就算放大四倍也不會有邊緣鋸齒的狀況發生，這結果跟講義上所描述的”Vector Graph 放大縮小效果較好”不謀而合。

(助教在這邊可以直接跑一次程式看 Matlab Figure 視窗跳出來的結果會比較清楚，因為要把圖片截圖放進 Report 中間已經把圖通通存成點陣圖了……)

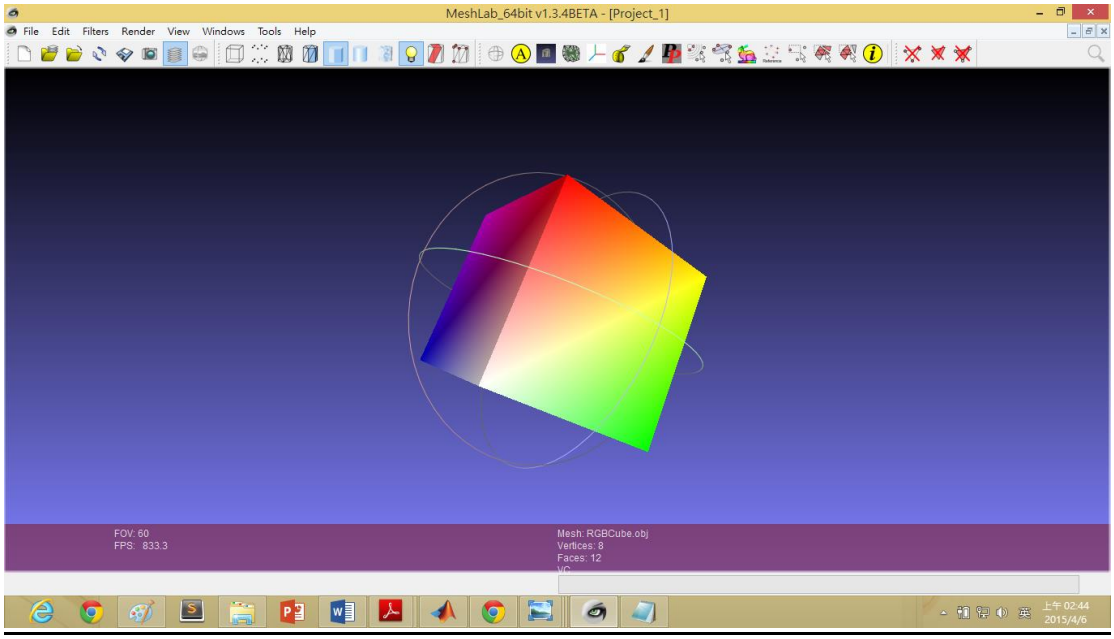
Part 2 – (a)

這一題是叫我們看懂 wavefront 的.obj 模型格式，然後把會產生殘缺不堪的立方體的程式修改，已生成完整且彩色的立方體。

其實因為這學期我有同時修計算機圖學，所以.obj格式其實已經算很清楚了，只是這次不是用openGL來畫模型，而是要用MATLAB來產生.obj檔。第一題的部分，其實只是側面的一半三角形沒有被畫到，所以很直接的，稍微修改助教產生另一伴三角形的code修改成如下:

```
for x = 1 : 4
    v1 = botVertIndex( x );
    v2 = botVertIndex( mod(x,4)+1);
    v3 = topVertIndex(mod(x,4)+1 );
    faces = [ faces ; v1 v2 v3 ];
end
```

就是把原本兩個 top 頂點一個 bot 頂點的三角形敢成兩個 bot 頂點跟一個 top 頂點罷了。顏色的部分就是照著 PDF 上面的圖形一一把八個頂點對應的顏色指定。



Part2 – (b)

這一題就比較有趣了，要我們從無到有生成一個描述 HSV 色彩模型的圓柱。
我的作法是在圓柱的上圓形跟下圓形的圓心都生成一個 VERTEX，然後讓兩個圓心分別跟周遭六十個頂點形成共一百二十個三角形。(頂點座標產生方法類似 PDF 所述：

```
vertAngle = linspace(0,2*pi,vertNum+1);
vertX =  cos(vertAngle);
vertY =  sin(vertAngle);
```

，Z 值我通通設為 1 跟-1。
側面的三角形也是跟上一題有點類似，由 120 個三角形構成，其中 60 個三角形是由兩個 top 頂點一個 bot 頂點，另外 60 個三角形則是兩個 bot 頂點一個 top 頂點構成。顏色的部分我先用 HSV 去模擬，上下圓形外側的頂點對應的 HSV，H 值繞一圈由 0 來到 1，S 統一為 1，V 的部分上圓形的頂點全為 1，下圓形的頂點全為 0，中心值的部分 S 和 H 則上下都為零。設定好了後最後別忘了把 HSV 轉成 RGB，畢竟 meshlab 只看的懂 RGB 模型(呼叫 hsv2rgb 函數)
。然後讓 MeshLab 自己去做顏色插值運算，形成的結果就像是 HSV 色彩模型的圓柱了！