# 運用 GPU 進行快速立體像素化

# Fast Voxelization of Polygonal Meshes on GPU

**陳煒松**
**亞洲大學**
**g95252020@ms1.asia.edu.tw**

**陳宏光**
**國立勤益科技大學**
**hank@ncut.edu.tw**

**摘要**

在本文中探討即時的實體三維模型使用硬體framebuffer 生成 voxel 的方法，生成的目標可分為實心立體像素化(solid voxelization)與表面立體像素化(surface voxelization)。在三維模型中我們將模型分為外部與內部，使用遮罩的方式來找出模型的表面輪廓線；在實體生成部份我們可以將遮罩關閉，如此即可以在模型的內部著色方式來生成實心立體相素化。整體生成過程將三維模型作空間上的切片，每一個切片經由硬體 rasterization 將資料轉換成 pixel 並在 fragment unit 使用影像遮罩來找出輪廓線，最後由 framebuffer 來傳回每一個切片的圖素資料並轉換回三維空間上的頂點資料。

**關鍵詞**：立體像素化(voxelization)。

## 1. 前言

立體像素(voxel = volume + element)是用來描述三維模型離散的數據單元，與模型的三角面表示比較上立體像素具有簡單且不需要拓樸的關係，立體像素化(voxelization)是指在確保模型表現具有一定準確度下，將模型的三角片轉換成離散的頂點集合來表示的過程。在應用層面有，體積模型建模[7]、三維空間域分析[8]、虛擬醫學成像[9]、幾何模型的視覺化成像[10]、碰撞偵測[11]等等。在立體像素表現方式可分兩種類型為表面立體像素化[1][2]將模型的表面轉換成離散的頂點數據來表示，與實心立體像素化[3][4]除了表面之外也將模型的內部轉換，在依照立體像素生成的資料型態分類上分為 binary voxelization 與 non-binary voxelization，前者是僅表示資料在三維空間上的有無，後者則是包含頂點的法向量或貼圖座標資訊等等，使用上較具多的應用範圍。在**錯誤! 找不到參照來源。**提出一個三角片面上增加區域的包覆性來使voxel能有效的增加反鋸齒效果與模型在多解析度的立體像素化下能更精確的表現出來，在[6]提出一個使用octree的方式來將空間物件分割出來，並依照分割的物件內部來填充voxel來產生三維模型的實心立體像素。在[12][13]使用三維空間切片的方式來分析模型每一切片的像素填充，投影方法[14]將模型表面投影到立方體上，接著使用depth buffer來將表面立體像素化，但這樣的方式對於複雜的模型會無法得到資訊，原因在於某些表面會被遮蔽。

在該研究中，我們探討一個使用硬體加速的立體像素化方法，對於一些複雜的模型能獲得完整的資訊。使用空間切片方式來合併完成最終的voxel模型，在本文voxel表現形態上可以生成實心與表面的立體像素，在離散的頂點資料型態上是以binary voxelization為資訊，使用離散頂點或是立方體來顯示最終結果。

## 2. 研究方法

本整體流程使用fragment unit來計算空間上(x, ,y ,z)每一軸上的切片的輪廓線所在位置，最後將資料傳回application memory中來作資料重複的偵測，最後回傳的voxel頂點資料即時將資料使用vertex array的方式上傳到video memory中以達到即時的效果呈現，在原始模型的rendering部份我們也使用相同方式來加速。整體流程分為兩個部份，第一個部份將三維模型頂點資料上傳到video memory中並平行計算(x, y, z)軸的切片資料，經由rasterization將資料轉為pixel型態來作影像遮罩處理，最後將結果從video memory讀取回application memory中來合併完成的資料。第二部份是將傳回來的每一切片資料作數值整理與合併回三維空間上的voxel頂點位置。如圖 1. 整體架構流程。

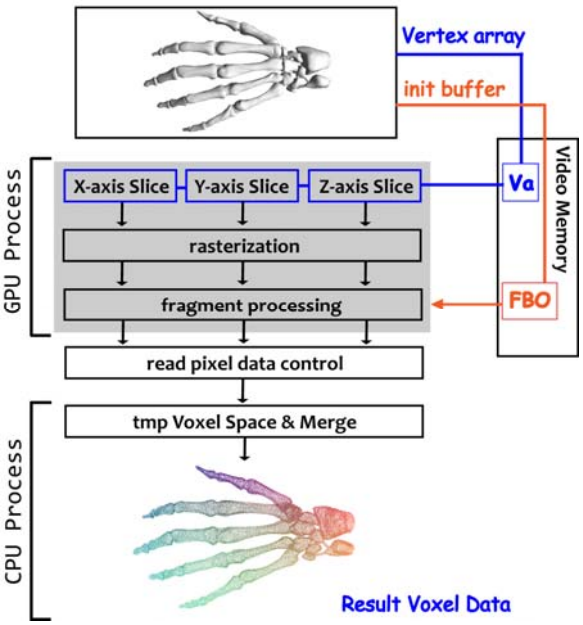


**圖 1. 整體架構流程**

## 2.1 模型內部與外部著色

由於我們目標是將rasterization後的模型資料分別標籤出表面區域與內部區域，所以必須要將面資料做處理，圖 2. 面索引順序定義逆時針為正面，紅色線順時針為反面，如此經由兩次的著色可以標籤出每一次切面的模型內部與外部區域，使用這樣的方式也可以避免當模型某一表面區域跟投影視角成平行的情況下，而無法快速且正確的找出該表面區域的voxel資料問題。在這原始模型資料部分使用vertex buffer object將資料預載到video memory上以改善面數過大的三維模型的繪製時間。

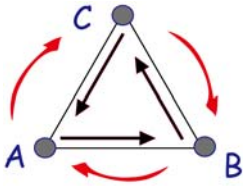


**圖 2. 面索引順序**

## 2.2 三軸投影切片與輪廓線擷取

首先將三軸投影畫面以off-screen buffer的方式來rendering到材質影像ID上面，經由硬體rasterization後我們可以得到每次切片的顏色標籤，在這裡我們設定模型大小為0~1之間而切片的次數跟切片影像的大小與使用者設定的影像解析度一致，如此作法將空間上的(x, y, z)三軸切割成等份大小的voxel space來作往後的資料對應。在輪廓線擷取部份，由於模型切片後該區域的內部標籤資料會顯現出來，所以在fragment process部分是使用邊緣線的擷取sobel edge detection即可以快速的計算出目前該切片的模型voxel表面，如圖 3. z軸上的切片位置與對應的輪廓線位置，紅色部分代表為設定的內部顏色標，在三軸的計算部分將三張畫面以才質影像ID0~2的方式設定，使用multiple render targets來作同時間的輪廓線處理運算與結果輸出。

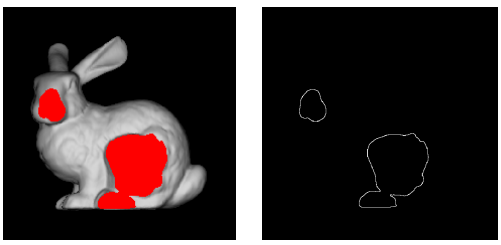

**圖 3. z 軸上的切片位置與對應的輪廓線位置**

在材質影像的輪廓線資料輸出是使用一個顏色通道來回傳到 application memory。

## 2.3 合併切片資料為 voxel

為了達到快速的篩選重複voxel頂點資料，我們建立一個暫存的三維陣列(slice_num$^3$)的bool值表格來填放空間上的voxel頂點位置的存在與否並對相鄰頂點作資料整理，使用索引陣列的方式可以快速的將各個方向軸切片重複的voxel資料剔除，在重整相鄰頂點方式是以索引該目標voxel的相鄰空間位置，若有找到則對設定改兩筆相鄰資料結構位置如表 1 Voxel Space & Merge pseudo code。

**表 1 Voxel Space & Merge pseudo code**

```
For each axis slice image
  If ( image[i] == 1 AND
  correspond voxel space[x][y][z] == false)
    Label voxel[x][y][z] = true;
    For each neighbor space[nx][ny][nz]
      If(neighbor space == true){
        Label voxel[x][y][z] neighbor struct AND
        Label neighbor[nx][ny][nz] each other; }
End each axis slice image
Delete voxel space;
```

## 3. 實驗分析

整體效能上對於模型資料量的多寡具有明顯的影響，以實驗數據顯示，越多的模型面數所消耗的硬體處理rasterization時間(毫秒)就越多，如表 2 模型資料量對每一切片平均效率分析(2563)，而本實驗對於(x, y, z)三軸上雖然總共只上傳模型資料到video memory一次來加速繪製效率，但是對每一軸都要繪製兩次的模型(正面、反面)，所以模型資料量越大對於硬體的負擔越重，平行計算三軸來說等於是要繪製六次模型，這個部份是需要改善的地方，以模型happy彌勒佛為例，在三軸的256個切片的尚未完成的時間內即時繪製voxel資料畫面更新速度為14FPS左右，完成後繪製voxel資料畫面更新速度為150FPS左右。實驗平台在Windows Vista、Duo T7700、ATI HD2600。

**表 2 模型資料量對每一切片平均效率分析(256$^3$)**

| Mode | Face | Rasteri-zation | fragment & Read pixel | Merge |
|---|---|---|---|---|
| cow | 5804 | 0.687 *ms* | 4.315 *ms* | 0.335 *ms* |
| bunny | 69443 | 7.168 *ms* | 4.336 *ms* | 0.402 *ms* |
| dragon | 871414 | 49.945 *ms* | 4.383 *ms* | 0.411 *ms* |
| happy | 1087425 | 62.356 *ms* | 4.418 *ms* | 0.417 *ms* |

## 4. 結論與未來展望

在本文中提出的方法可以正確的提取對於模型表面與視角平行的區域部分，對於形態複雜的三維模型亦能夠正確的提取出voxel資料，下圖為幾種範例，彩色為voxel採樣的頂點結果，藍色則為將voxel頂點資料以立方體的形態表現出來，如圖 4. 針對奇異外型的三維模型(2563)圖 5. 針對高面數的三維模型表面細節(5123)。而在本實驗裡有兩個重要部份需要做改善的，第一部分為減少硬體對於繪製大型模型資料的次數，第二部份為暫存的voxel space，雖然使用該方式可以快速的完成voxel資料的整理，且切片完成後即刪除該暫存空間，但是若要將切片的解析度提高到 $1024^3$ 的話，在application memory的消耗是非常不切實用的，這部分可以參考[15]將每一片取得的資料使用材質合併的方式來儲存voxel資料。

## 參考文獻


[1] F. Dachille and A. Kaufman. Incremental triangle voxelization. In Proceedings of Graphics Interface, pages 205 – 212, May 2000.

[2] J. Huang, R. Yagel, V. Filippov, and Y. Kurzion. An accurate method for voxelizing polygon meshes. In IEEE Symposium on Volume Visualization, pages 119 – 126, 1998.

[3] D. Haumont and N.Warzee. Complete polygonal scene voxelization. ACM Journal of Graphics Tools, 7(3):27–41, 2002.

[4] M. Sramek and A. Kaufman. Alias-free voxelization of geometric objects. IEEE Transactions on Visualization and Computer Graphics, 5(3):251–267, 1999.

[5] F. Dachille and A. Kaufman. Incremental triangle voxelization. In Proceedings of Graphics Interface, pages 205–212, May 2000.

[6] D. Haumont and N.Warzee. Complete polygonal scene voxelization. ACM Journal of Graphics Tools, 7(3):27–41, 2002

[7] S. Wang and A. Kaufman. Volume-sampled 3d modeling. IEEE Computer Graphics and Applications, 14(5):26–32,1994.

[8] S. Beckhaus, J. Wind, and T. Strothotte. Hardware-based voxelization for 3d spatial analysis. In Proceedings of the 5th International Conference on Computer Graphics and Imaging, pages 15–20, Canmore, Alberta, Canada, August 2002. ACTA Press.

[9] K. Kreeger and A. Kaufman. Mixing translucent polygons with volumes. In Proceedings of IEEE Visualization, pages 191–198, USA, October 1999.

[10] R. Westermann, O. Sommer, and T. Ertl. Decoupling polygon rendering from geometry using rasterization hardware. In Proceedings of the 10th Eurographics Workshop on Rendering, pages 53–64, 1999.

[11] M. Boyles and S. Fang. Slicing-based volumetric collision detection. ACM Journal of Graphics Tools, 4(4):23–32, 2000.

[12] S. Fang and H. Chen. Hardware accelerated voxelization. Computers and Graphics, 24(3):433–442, 2000.

[13] S. Fang and D. Liao. Fast csg voxelization by frame buffer pixel mapping. In Proceedings of the ACM/IEEE Volume Visualization and Graphics Symposium 2000, pages 43–48, Salt Lake City, UT, USA, October 2000.

[14] G. P. Evaggelia-Aggeliki Karabassi and T. Theoharis. A fast depth-buffer-based voxelization algorithm. ACM Journal of Graphics Tools, 4(4):5–10, 1999.

[15] Zhao. D, Wei. C, Hujun. B, Hongxin. Z, Qunsheng. P, Real-time Voxelization for Complex Models. In Proceedings of Pacific Graphics 2004, pp.43-50.

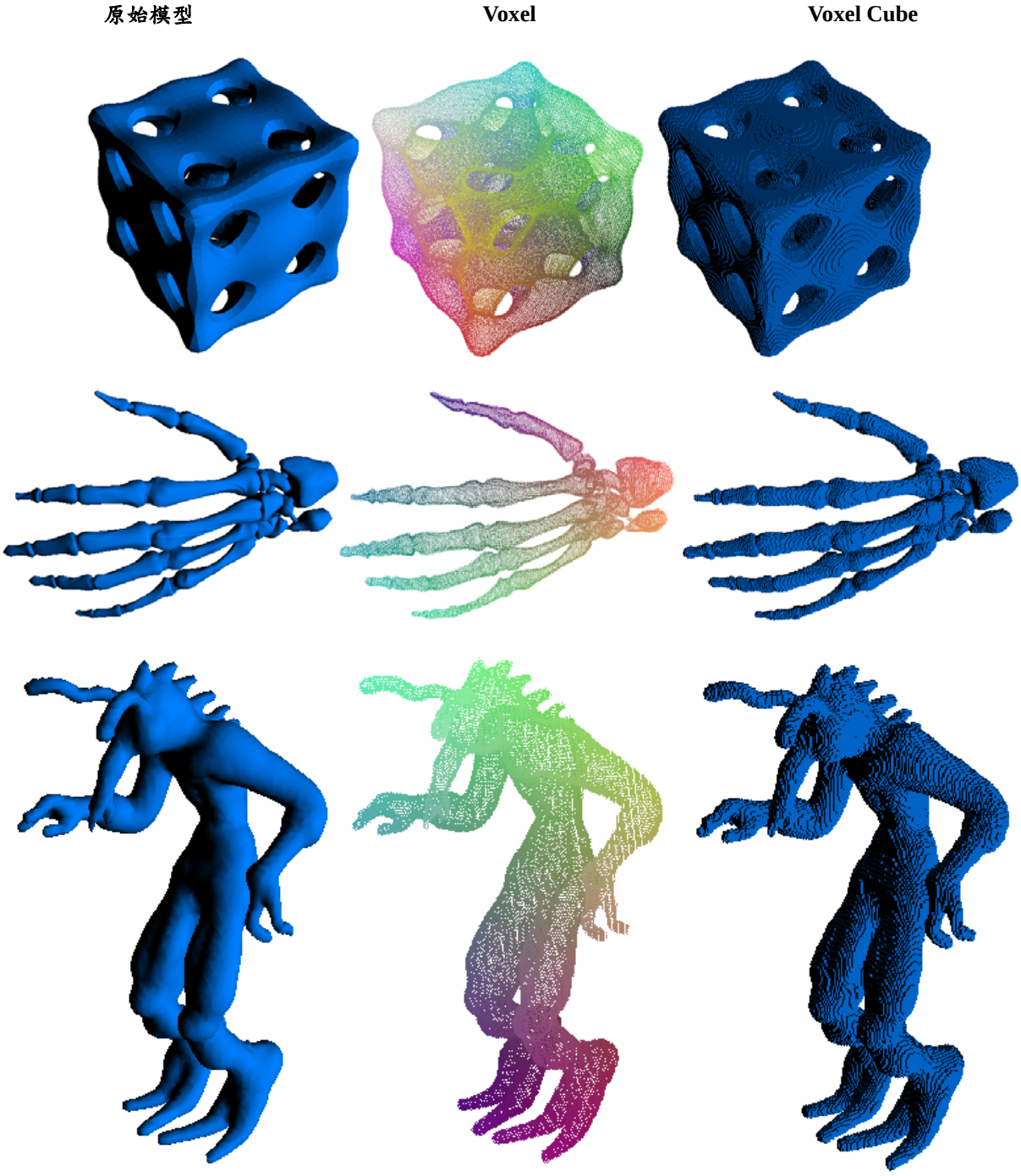


圖 4. 針對奇異外型的三維模型($256^3$)為例

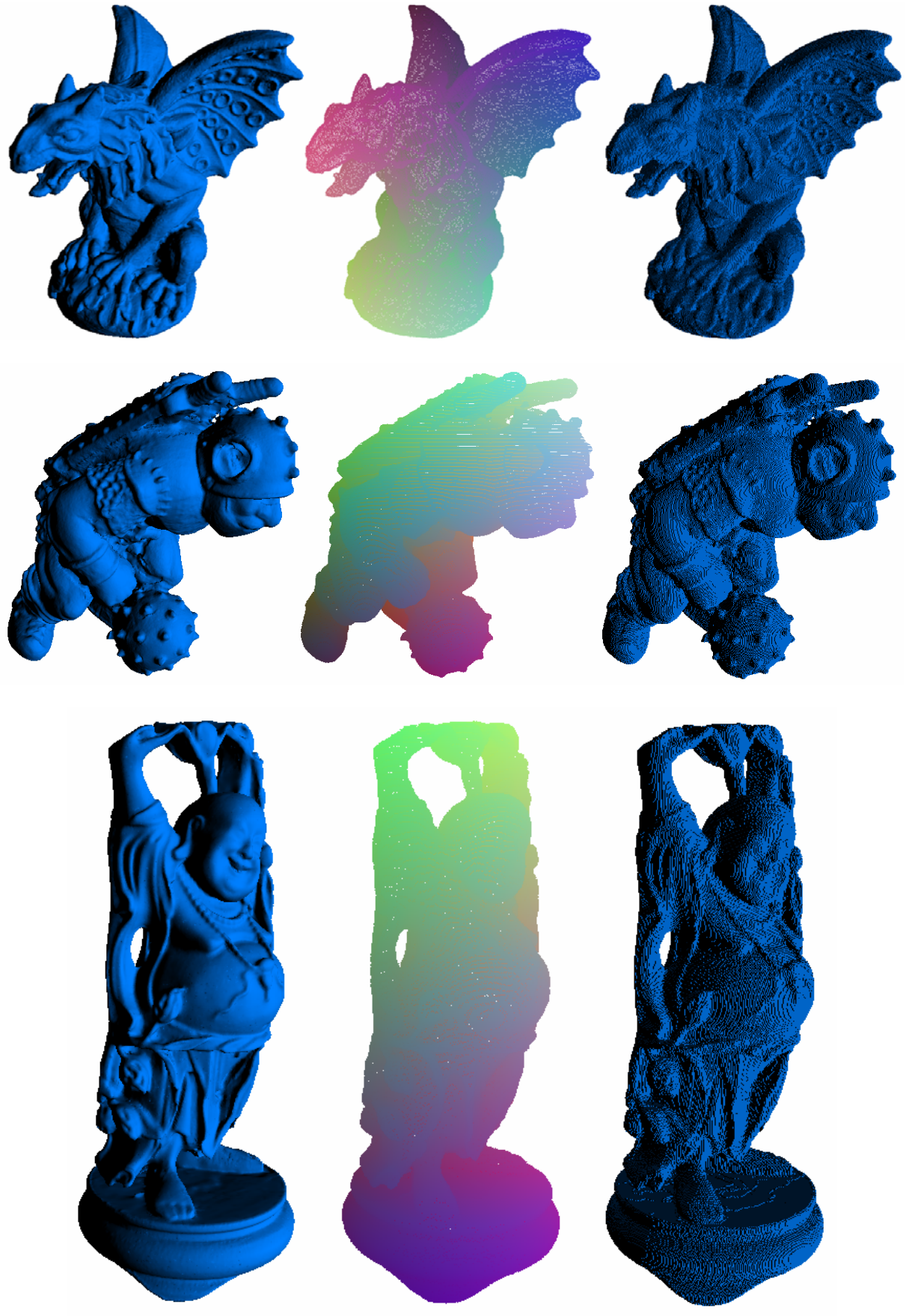

圖 5. 針對高面數的三維模型表面細節($512^3$)為例