King Abdullah University of Science and Technology

Fall 2015 CS 247 Scientific Visualization Assignment 2

Gang Liao [*]   ID: 133267

Thursday 8th October, 2015

1 2D Iso-contours Rendering

To make sure when you change the slice everything gets updated correctly, before extracting the data, we need to diverge the code segment for different **current_axis**.

```
if (current_axis == 0)
{
int x = current_slice[current_axis];
for (int y = 0; y < vol_dim[1] - 1; y+= 1) {
for (int z = 0; z < vol_dim[2] - 1 ; z+= 1) {
        cell = 0;
        if (Data(x, y, z) < current_iso_value - epsilon) cell |= 8;
        if (Data(x, y + 1, z) < current_iso_value - epsilon) cell |= 4;
        if (Data(x, y + 1, z + 1) < current_iso_value - epsilon) cell |= 2;
        if (Data(x, y, z + 1) < current_iso_value - epsilon) cell |= 1;
        GetLinePoints(cell, z, y, vol_dim[2], vol_dim[1], Data(x, y, z),
           Data(x, y + 1, z), Data(x, y + 1, z + 1), Data(x, y, z + 1));
        }}
}
```

[*] Extreme Computing Research Center, Department of Computer Science, King Abdullah University of Science and Technology (KAUST). Email: liao.gang@kaust.edu.sa

Since there exists 16 intersected models in 2D cells, it can be reduced to 8. I set **edgeTable2D** as:

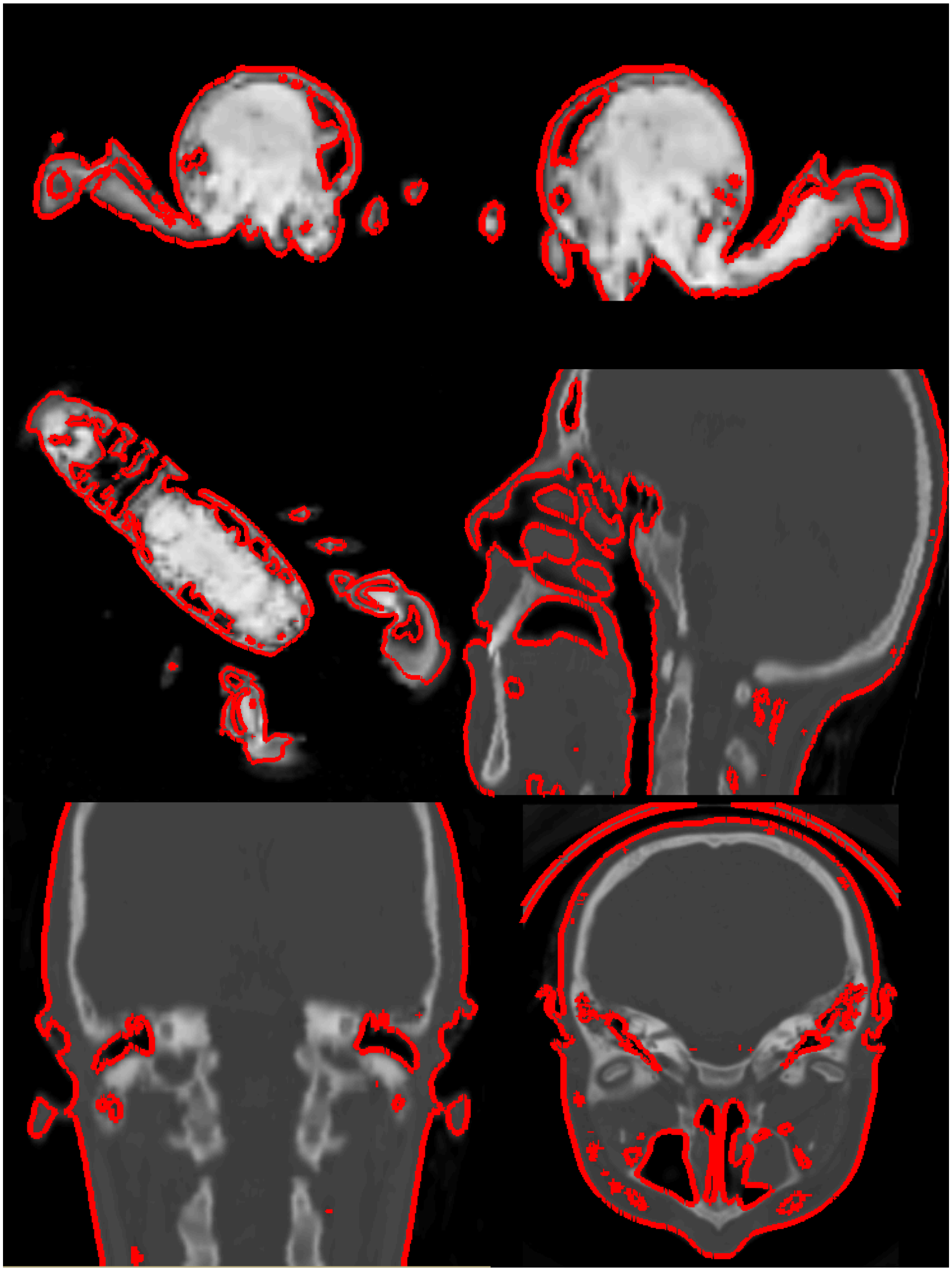
```
static char edgeTable2D[16] = {
// =========================================
// TODO: fill marching squares table
// =========================================
        0x0, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
        0x07, 0x06, 0x05, 0x04, 0x03, 0x02, 0x01, 0x0
//To recognise the same intersected models
};
```

That's can be used to interpolate the positions of vertices in specified edges. All pairs of vertices are pushed into C++ Vector **contour**. In order to find the correct position in the screen, their position should be scaled via right aspect.

```
for (int i = 0; i < contour.size(); i += 4)
{
        x1 = contour[i] * h;
        y1 = contour[i + 1] * w;
        x2 = contour[i + 2] * h;
        y2 = contour[i + 3] * w;

        glColor3f(1.0, 0.0, 0.0);
        glLineWidth(5);
        glBegin(GL_LINES);
        glVertex2d(y1, x1);
        glVertex2d(y2, x2);
        glEnd();
}
```

1 Result



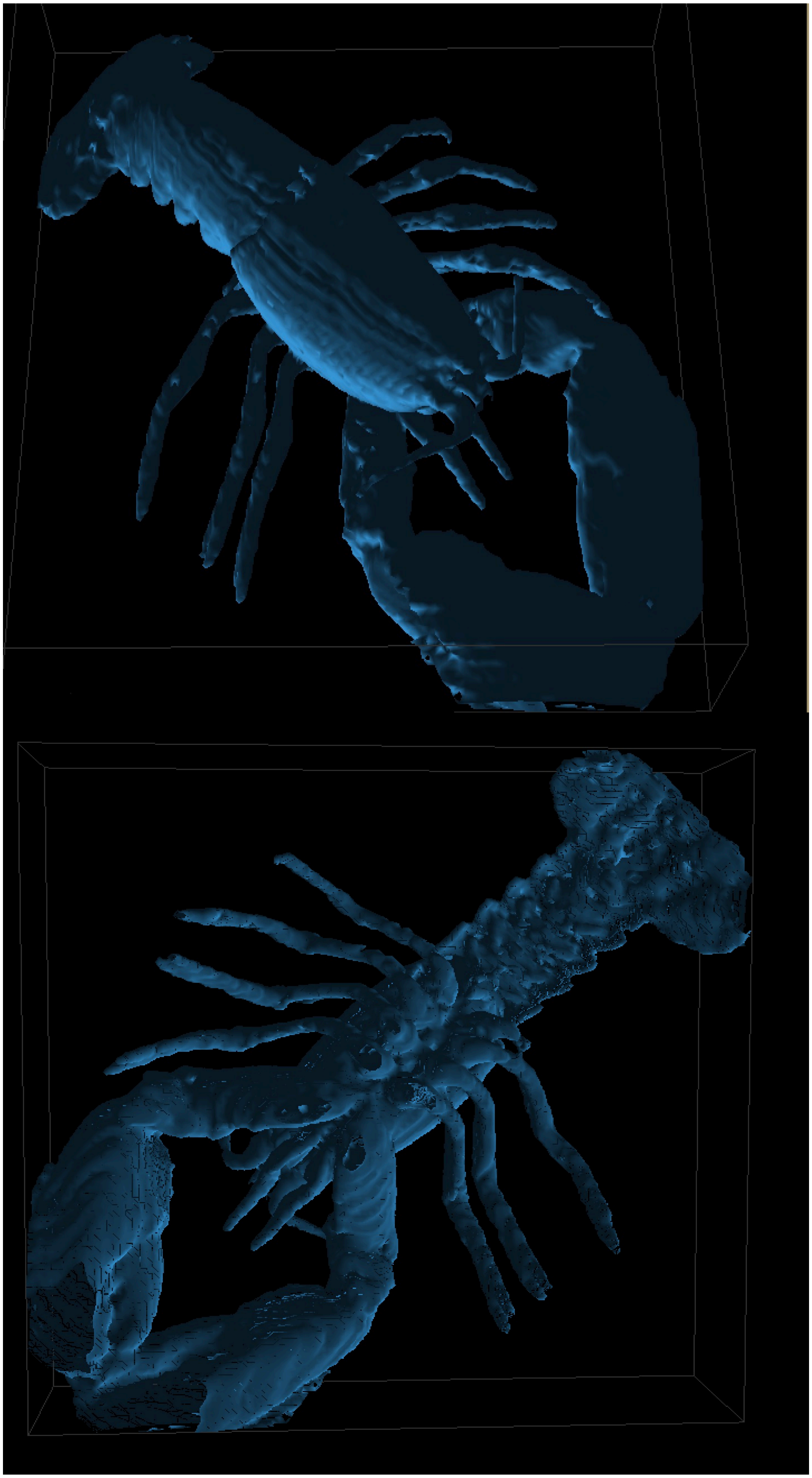
2 3D Iso-surface Rendering

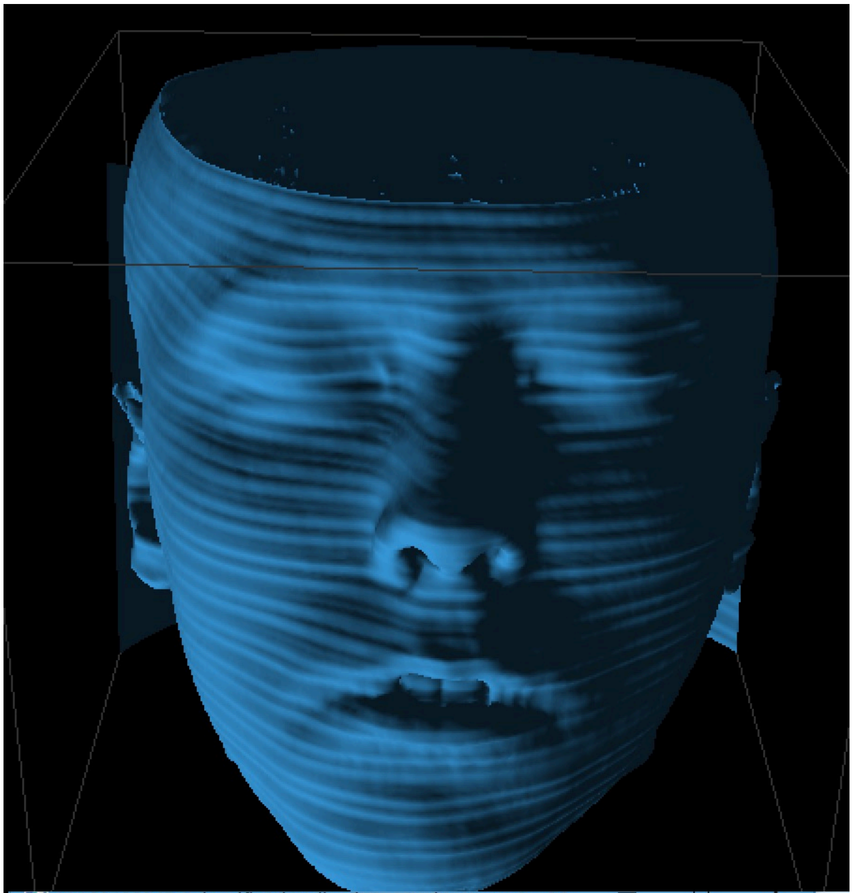
3D Iso-surface algorithm is similar to 2D Iso-contour. First, we need to get 8 points (virtual cube) according to current points' neighbors. Then, calculating cube index via the specified vertices inside or outside of iso value.

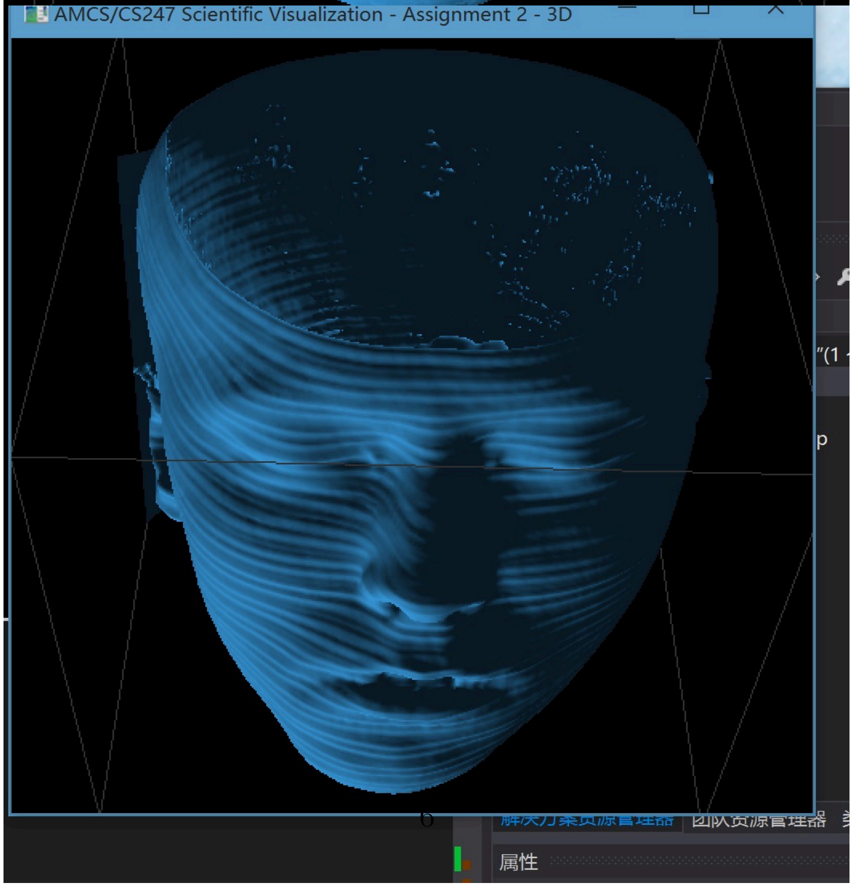
```
cubeindex = 0;
if (cube.val[0] < isolevel) cubeindex |= 1;
if (cube.val[1] < isolevel) cubeindex |= 2;
if (cube.val[2] < isolevel) cubeindex |= 4;
if (cube.val[3] < isolevel) cubeindex |= 8;
if (cube.val[4] < isolevel) cubeindex |= 16;
if (cube.val[5] < isolevel) cubeindex |= 32;
if (cube.val[6] < isolevel) cubeindex |= 64;
if (cube.val[7] < isolevel) cubeindex |= 128;
```

Pre-defined **edgeTable** includes which pair of vertices should be interpolated in bits. Finally, we can get all vertices for triangles via **triTable**.

```
std::vector<float> trian;
for (i = 0; triTable[cubeindex][i] != -1; i += 3)
{
float x1 = vertlist[triTable[cubeindex][i]].x;
float y1 = vertlist[triTable[cubeindex][i]].y;
float z1 = vertlist[triTable[cubeindex][i]].z;
float x2 = vertlist[triTable[cubeindex][i+1]].x; ... //also float y2, z2
float x3 = vertlist[triTable[cubeindex][i+2]].x; ... //also float y3, z3
//store normal per triangle into normal vector
normal.push_back(TriangNorm(vertlist[triTable[cubeindex][i]], vertlist[
    triTable[cubeindex][i + 1]], vertlist[triTable[cubeindex][i + 2]]));
trian.clear();
trian.push_back(x1); ...//also store y1, z1
trian.push_back(x2); ...//also store y2, z2
trian.push_back(x3); ...//also store y3, z3
//store triangle into iso-surface vector for post-rendering
isosurface.push_back(trian);
}
```






AMCS/CS247 Scientific Visualization - Assignment 2 - 3D
”(1
p
解决方案资源管理器
团队资源管理器
属性