CSC4140 Final Project

Computer Graphics

May 23, 2022

Point Cloud to Mesh

This assignment is 30% of the total mark.

Student ID: 120090266

Student Name: Feng Yutong

This assignment represents my own work in accordance with University regulations.

Signature: Feng Yutong

1 Overview

I choose point to cloud as my topic for final project. Due to time limit, I only successfully implement the algorithm of Ball-Pivoting Algorithm (BPA). For poisson surface reconstruction, I will introduce it and compare it with PBA theoretically.

Ball-pivoting Algorithm

BPA rolls an imaginary tiny ball across the "surface" of points. This tiny ball is dependent on the scale of the mesh. When a ball is stuck by three points, they prop the ball up form a seed triangle, which becomes part of the mesh. From that location, roll the ball, pivoting along the triangle edge formed from two points. The ball then settles in a new location: a new triangle is formed from two of the previous vertices and one new point, the expansion triangle is added to the mesh. We repeat this process untill the mesh is fully constructed.

Poission surface reconstrution

The idea of poisson surface reconstruction is to construct watertight implicit surfaces by converting discrete sample point information on the surface of an object into a continuous (integrable, which is the core reason) surface function. The problem can be convert to find the best least-squres approximate solution of

$$\Delta\chi = \nabla\vec{V}$$

2 Ball-pivoting Algorithm

2.1 Implement flow

Find Seed Triangle

1. Pick any point σ not yet used by the reconstructed triangulation.

2. Consider all pairs of points σ_a, σ_b in its neighborhood in order of distance from σ.

3. Check that the triangle normal is consistent with the vertex normals, i.e., pointing outward.

4. Test that a ρ-ball with center in the outward halfspace touches all three vertices and contains no other data point.

5. Stop when a valid seed triangle has been found.

Ball pivoting

All c_x are locted on circular trajectory γ. By calculating the intersection of ρ-ball and circle γ, we can calculate the σ_x at the center of the circle. Record the coordinates of the first hit point and its corresponding spherical center.

Join

Rotate around e_{ij} until hit σ_k. If σ_k is not used, output triange $(\sigma_i, \sigma_j, \sigma_k)$, delete e_{ij} and add

new edges e_{ik}, e_{kj}. Otherwise:

1. σ_k is an internal mesh vertex, (i.e., no front edge uses σ_k). Mark e_{ij} as a boundary edge.

2. σ_k belongs to the front. We first check that adding the candidate new triangle would not create a nonmanifold or nonorientable manifold. This is easily accomplished by looking at the existence and orientation of edges incident on σ_k, Then we output triange $(\sigma_i, \sigma_j, \sigma_k)$ and glue.

Glue

The glue operation removes from the front pairs of coincident edges, with opposite orientation.

2.2 Data Structure

Vertex

A Vertex contains a integer indicating its index, two 3D vectors containing a point's position and the point's normal, a flag indicating whether is has been used (inner point).

Edge

An edge is represented by its two endpoints and the opposite vertex (Vertex), the center (3D vector) of the ball that touches all three points, and links to the previous and next edge along in the same loop of the front. Additionally, it holds a flag of whether if (1) it is an inner edge, if (2) it is on a boundary, and if (3) it is an active edge. An acive edge an be use as the pivot axis. An inner edge has already been processed and is part of the front. A boundary edge is on the border of the mesh, where there is only one surface on its side.

Triangle

A triangle contains its three Vertexs and its surface normal.

Voxel

A voxel array based on bucket-sort is uesed to get the neighbour of a vertex. The voxel is a 3D grid that stores Vertex. The side length of voxel is 2ρ. We can easily get the neighbour of a vertex by dividing its coordinate by 2ρ ($3 \times 3 \times 3$ voxels in total). The program convert it to a 1D vector of Vertex to save memory.

Front

The front is a deque of edges. When finding active edge, we can easily delete inner edge in deque to improve efficiency.

Note Only Voxel stores vertices instance, others stores the pointers of certain data structure. For simplicity, the report just use its name.

2.3 Result

Bunny (radius = 0.005)

Bunny (radius = 0.005) nuance

Bunny (radius = 0.003)

Bunny (radius = 0.003) nuance

Bunny (radius = 0.003)

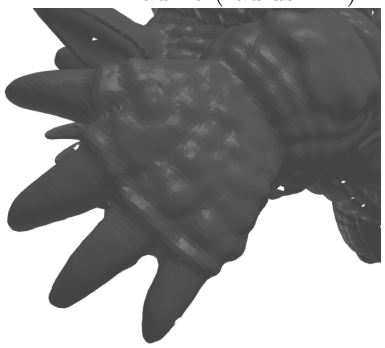
Bunny (radius = 0.002) nuance

Figure 1: Reconstrution of bunny.ply with different ball radius

Armadillo (radius = 1)

Armadillo (radius = 1) nuance

Armadillo (radius = 0.5)



Armadillo (radius = 0.5) nuance

Figure 2: Reconstrution of Armadillo.ply with different ball radius

2.4 Challenges and solutions

1. **How to decide ball radius ρ?** As we can see from Figure 1 and 2, different ball radius can result in different reconstruted mesh. The radius is obtained empirically based on the size and density of the input point cloud. In theory, it should be a little larger than $\frac{AverageDistance}{2}$. However, calculating the average distance is time-consuming, so I just fix a radius ρ and test with $\{0.1\rho, 0.5\rho, ..., 5\rho\}$ to find the best result.

2. **BPA (without extension) is suitable for what kind of cloud point?** BPA has better result when the point is uniformly sampled on the surface, which means distances between connected verices do not change much. We can use different radius performing on one model to simluate different sampling denstiy situation. When the sampling density is too low (smaller radius), some of the edges will not be created, leaving holes. We can see from the bunny' feet. When the cutvature of the manifold is larger than $\frac{1}{\rho}$, some of the sample points will not be reached by the pivoting ball, and features will be missed. We can see from the smoother mesh and the nuances in the connection between bunny's neck and Armadillo's fingers.

3. **How to find the first hit point in ball pivot?** Check whether three points' normals are in the same direction and the new ball center is above the traingle. Check whether the point already has an inner edge. Use direction vector of the ball center in the plane of circular trajectory

γ to calculate the smallest pivoting angle.

4. **What if the surface has a crease or valley, s.t. the distance between the surface and itself is less than the size of the ball?** The ball will just roll over the crease and ignore the points within the crease, resulting in an inaccurate mesh. To fix this, multiple ball radius must be used in BPA (extension multiple passes).

3 Building Environment And Running

3.1 Environment

The project is not based on assignment 5 for simplexity. Please install the following dependencies and environment:

- python3
- Eigen *This environment has been setup in assignment 1*
- Trimesh *refer to https://trimsh.org/install.html and use 'pip install trimesh[all]'*
- c++17

3.2 Folders

The are four folders:

data stores input files, downloaded from http://graphics.stanford.edu/data/3Dscanrep/.

result stores output file in xyz, dae and txt format. Since I do not color mesh (default is black), its is suggested use a online viewer https://3dviewer.net/ for a quick view of the dae file. Also, you can view the result using its corresponding txt file by **viewer.py**. For example, if you want to see teapot.dae, modify the **input.txt** to **teapot.txt** in the **viewer.py** and enter

```
python3 viewer.py
```

It supports some simple UI (press -w to see triangles).

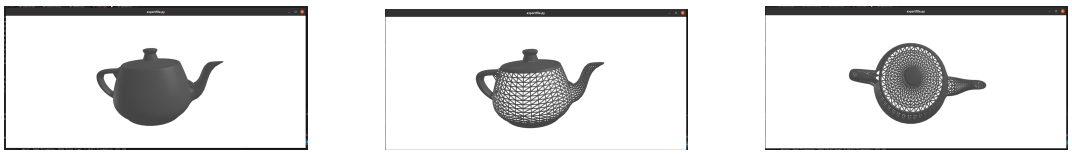


Figure 3: Visual by viewer.py

src stores three files. **main.cpp** implements BPA algorithm and calls **loadfile.py** and **exportfile.py** to parse and export file respectively. **To compile the file, use 'bash compile.sh'**

instruction. **exportfile.py** will generate a UI window to see the result. To run the executed file **bpa**, you need enter the file and ball radius. The recommended radius for bunny and Armadillo are 0.002 and 1 respectively. For example

```
bash compile.sh
./bpa ../data/bunny.ply 0.002
./bpa ../data/Armadillo.ply 1
```

result_pic stores result in png format.

4 Poisson Surface Reconstruction

4.1 Algorithm flow

Octree construction

An adaptive spatial mesh division method is adopted (the depth of the grid is adjusted according to the density of the point cloud). Octree is defined according to the location of the sampling point set, and then subdivided into octree to make each sampling point fall on the leaf node with the depth of D.

Set function space

Set space function F for each node of octree. The linear sum of all node function F can represent vector field $\vec{V}$, and the base function F adopts n-dimensional convolution of box filtering.

Vector field creation

In the case of uniform sampling, assuming that the blocks divided are constant, the gradient of the indicator function is approximated by the vector field $\vec{V}$. Using cubic spline interpolation.

Solving Poisson equation

The solution of the equation is obtained iteratively by Using Laplace matrix;

Isosurface extraction

In order to obtain the reconstructed surface, the threshold value should be selected to obtain the isosurface. First, the position of sampling points is estimated, and then the mean value is used to extract the isosurfaces, and Marching Cubes (moving Cubes) algorithm is used to obtain the isosurfaces.

4.2 Comparison with BPA

The time complexity of BPA and Poisson are $O(N)$ and $O(N \log N)$ in theory. However, time complexity of BPA can also be expressed as $O(B^3 N)$ where B is the number of vertices in the 27 neigbourhood voxels. B is determined by the size and denstiy of the point cloud and ball radius.

Intuitively, we can set smaller ball radius to reduce the number of verices in one voxel, but this can lead to holes in mesh (details see challege 2). Table 1 shows the impact of constant B.

Test Case	Points	Triangles	Total time	Triangles/second
bunny radius = 0.005	34835	64451	39.5419	1629.94
bunny radius = 0.003	34835	68168	8.6765	7856.64
bunny radius = 0.002	34835	69290	4.0269	17206.7
Armadillo radius = 1	172975	339274	23.8948	14198.77
Armadillo radius = 0.5	172975	345478	7.47055	46245.3

Table 1: Performance comparison between different radius

I refer to the internet and find B can be reduced by branch and cut method in search (I do not implement), so we can still consider it as $O(N)$. The constant for possion surface reconstruction is relatively small, so we can just assume time complexity is $O(N \log N)$. Therefore, poisson is slower than BPA.

Both BPA and poisson assump uniform samples. Posisson can fix this limit by modifying a little on its equation while BPA can fix this by using mulitple passes. With these improvement, BPA perform better in preserving the feature. However, poisson is more resilient to noise by using larger values of samples per node. At last, poisson consumes more memory space, especially for the matrix.

5 Acknowledgement

BPA refers to paper1 and Poisson Surface reconstrucion refers to paper2. Also, I want to thank my classmate Jing Derong who helps me understand some content of the paper. And we work together to analyse the performance of the algorithm.