DGP 第二次作业：Poisson 曲面重建

SA22001009 陈泽豪

March 23, 2024

1 作业介绍

Poisson 曲面重建是一种在计算机图形学和计算机视觉领域广泛使用的技术，旨在从一组散乱的点云数据中重建出平滑的曲面。这种方法最初由 Michael Kazhdan 等人在 2006 年的论文《Poisson Surface Reconstruction》中提出。基于点云数据的曲面重建是三维模型获取、计算机辅助设计（CAD）、虚拟现实（VR）和医学成像等领域的一个核心问题。Poisson 曲面重建方法通过解决 Poisson 方程的偏微分方程（PDE）来实现这一目标，该方法的关键优势在于它的鲁棒性和高质量的重建结果。

点云数据是三维空间中一组离散点的集合，通常由三维扫描仪捕获，代表了物体表面的样本点。然而，这些点本身并不构成一个完整的曲面，它们缺乏连接信息和表面的拓扑结构。Poisson 曲面重建方法的核心思想是利用点云的局部特征来推导出一个全局一致的曲面，具体来说，是通过估计每个点处的表面法向量，并将这些信息作为 Poisson 方程的输入，来计算一个标量函数的梯度场。然后，通过求解这个 Poisson 方程找到一个标量场，其梯度场最接近于输入的法向量场。最终，通过提取这个标量场的等值面，得到一个平滑且连续的表面，从而完成从点云到曲面的重建。

Poisson 曲面重建是三维数据处理领域的一项关键技术，它通过数学上优雅的方法将离散的点云数据转换为连续的曲面，为三维建模、视觉效果制作、文物保护和生物医学研究等多个领域提供了强大的工具。

2 整体的算法框架

整个代码的实现主体实际上可以分为如下的几步。

2.1 数据准备和预处理

Poisson 曲面重建算法是从一组散乱的点云数据中重建出平滑曲面的一种方法。它基于解决 Poisson 方程，通过估计点云的法向量来推导出整个表面。以下是 Poisson 曲面重建的整体算法框架叙述：

- 输入点云数据：算法以一组三维空间中的散列点云作为输入，这些点通常通过三维扫描仪或其他成像技术获得。

- 估计法向量：对于点云中的每个点，估计其所在表面的法向量。这一步是重建过程中的关键，因为法向量将作为后续 Poisson 方程求解的重要输入。法向量的估计可以通过考虑点及其邻近点的局部几何结构来完成。
- 在本次实验中，我们直接读取一个.obj 文件或者其他网格形式的文件，并利用 Open3d 库直接计算出这个文件在点上的法向量。

2.2 对空间进行划分

- 首先将所有的点全都包在一个包围盒内，然后把包围盒所在的空间进行划分。
- 在论文中采用了八叉树划分，但是论文对于八叉树的构建较为特殊，需要所有的叶子节点都落在深度为 D 的叶子节点内。但是在库函数内都没有做额外划分，有些叶子节点只划分到深度 D 以下。在我的处理里采用了体素划分。
- 建立划分的空间上的基函数，让基函数去张出整个函数空间，因此为了进行 Poisson 表面重建，可以用示性函数去进行隐式表示，在物体内部为 1，外部为 0，那么隐式曲面为 0 的表面就是要重建出的 surface，如下图所示。

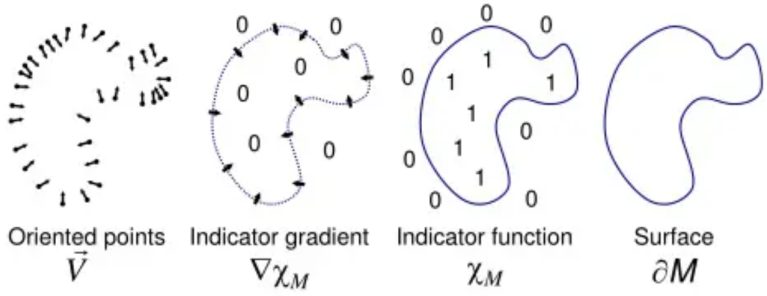

Figure 1: 函数示意图

2.3 设定空间上的基函数

当我们采用体素划分时，每个体素都是一个立方体，我们在这个立方体上建立一个具有紧支集的基函数。在实现的时候我采用的是二次 B 样条基函数。$[-1.5, 1.5]$ 上的基函数如下所示：

$$f(x) = \begin{cases} 9/8 + 1.5x + 0.5x^2, -1.5 \le x \le -0.5 \\ 3/4 - x^2, -0.5 < x < 0.5 \\ 9/8 - 1.5x + 0.5x^2, 0.5 \le x \le 1.5 \end{cases} \tag{1}$$

每个立方体存在一个立方体中心 c^0，立方体的宽度为 w，那么一个三维空间立方体上的二次 B 样条基函数可以表示为：

$$g_o(x, y, z) = f(\frac{x - c_x^0}{w}) f(\frac{y - c_y^0}{w} f(\frac{z - c_z^0}{w}) \tag{2}$$

因此在整个基函数张成的空间上去表示示性函数如下：

$$\chi(x, y, z) = \sum_o x_o g_o(x, y, z) \tag{3}$$

同时用已知点的法向插值出物体表面的向量场 V：

$$\vec{V} = \sum_{s \in S} \sum_{o \in Ngbr_D(s)} \alpha_{o,s} g_o(x,y,z) s.\vec{N} \tag{4}$$

其中 $Ngbr_D(s)$ 表示的是离采样点 s 最近的八个最近邻。$\alpha_{o,s}$ 则是三线性插值系数。

2.4 建立 Poisson 方程

1. 定义标量场：已经定义了一个标量场 χ，其梯度场接近物体表面的向量场 $\vec{V}$。
2. 构建方程：基于上述假设，构建一个 Poisson 方程，使得该方程的解决方案能够最小化梯度场与估计出的法向量之间的差异。

因此需要进行如下的优化：

$$min_{\chi} ||\nabla\chi - \vec{V}|| \Rightarrow \nabla\chi \approx \vec{V} \Rightarrow \Delta\chi = \nabla \cdot \vec{V} \tag{5}$$

将其转化为内积相等，利用有限元处理，因此有：

$$< \Delta\chi, g_{o'} > = < \sum_o x_o \Delta g_o(x,y,z), g_{o'} > \tag{6}$$

$$= \sum_o x_o < \Delta g_o, g_{o'} > \tag{7}$$

同理有：

$$< \nabla \cdot \vec{V}, g_{o'} > = \sum_{s \in S} \sum_{o \in Ngbr_D(s)} \alpha_{o,s} s.\vec{N} < \nabla g_o(x,y,z), g_{o'} > \tag{8}$$

因此可以直接建立方程组接触未知系数 x_o。

2.5 求解 Poisson 方程

1. 使用数值算法求解该离散化方程，获得基函数的系数 x_o。
2. 得到 $\chi = \sum_o x_o g_o(x,y,z)$，将体素网格上的点带入 χ 的表达式里得出体素网格上的函数值。
3. 将体素网格连同函数值输入到 marching cube 函数内，得到离散点云的曲面三角化重建结果。

2.6 后处理

这样得到的 marching cube 等值面可能有多个连通区域，其中有我们想要的，也有无用的重建曲面，因此需要进行连通区域的划分筛选。

3 最终结果展示

实验采用 python 3.10，下载 open3d 0.18.0 进行模型的读取与三角网格显示（open3d 暂不支持 python 3.12），marching cube 采用 skimage 库的 measure.marching_cube 算法，同时利用 scipy 库建立稀疏矩阵以及 quad 积分，利用 networkx 库提取出最后重建结果里的多个连通部分。

3.1 示例一

给出如下的 bunny 点云以及点云上的法向

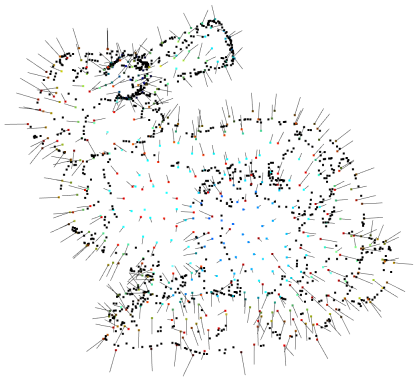

Figure 2: bunny 点云以及法向

建立 36 * 36*32 大小的体素网格，只需在 main.py 点击运行就可以得到结果如下：

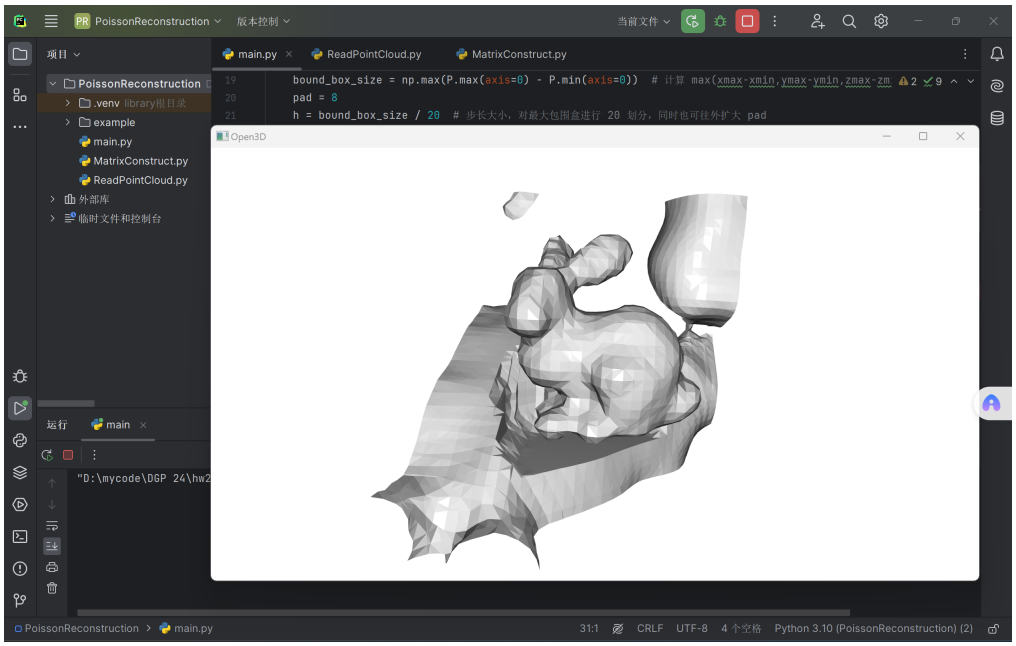

Figure 3: 运行结果

通过提取连通网格取出其中的兔子重建结果如下：

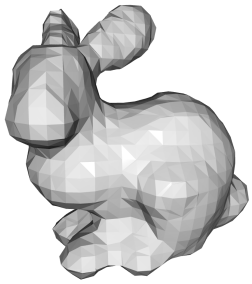

Figure 4: 运行结果

我们再给出更多的示例。

3.2 示例二

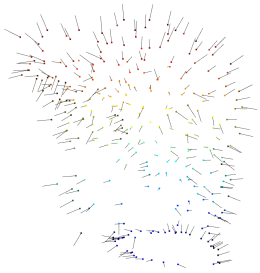

Figure 5: David 点云以及法向

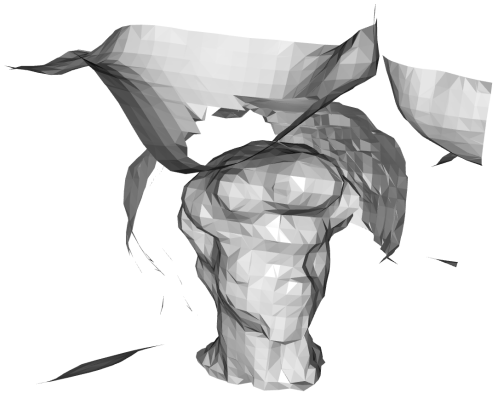

Figure 6: 重建结果

Figure 7: 提取出其中的 David 部分

此时是 33*33*36 的体素网格，可以看出即使划分的不够细，也可以得出一个比较好的结果。下面再给出一个例子。

3.3 示例三

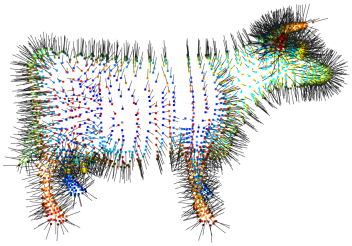

Figure 8: Cow 点云以及法向

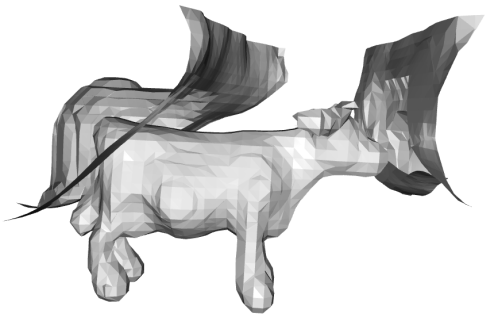

Figure 9: 重建结果

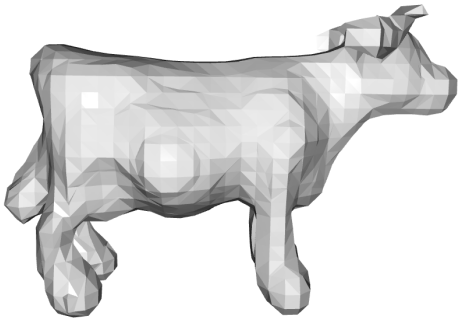

Figure 10: 提取出其中的 Cow 部分

此时的体素网格采用的是 46*35*26 的网格，对于精细处如牛角的恢复没有很好，此时如果加细网格会让整体体素过多导致难以计算，或许可以用并行计算解决。

4 总结

可以发现 Poisson 重建算法确实可以得到不错的结果，但是它的计算量比较大，如果能与 GPU 进行结合，可以在短时间内计算出很好的结果。