数据可视化期末报告

DonghaoLi

JingWang

TianranZhang

一、选题介绍

任务介绍

在三维数据可视化中，对三维物体的形变控制是重要的一部分。我们希望可以通过对一些点的控制来对三维物体进行形变，进而产生我们希望的形状，并且可以用于研究物体在受到外力影响下的形变。我们主要借助Brigham Young University 的 Sederberg 和 Parry 提出的自由变形算法（FFD）进行探究及实践，其核心思想是通过一系列控制点的位移确定物体形变程度。

目前这一物体形变的方法已经被广泛应用，各种商业软件内置了这种变形方式。我们希望在VTK中实现这种变化，并且定制一些功能如播放动画等，使用户使用更加方便。

三维物体的构成

为了产生形变，我们首先要分析三维物体的组成。在计算机中，我们使用多面体存储三维物体，其分为两部分：

- 顶点：计算机记录每个顶点的坐标；

- 多边形面：计算机记录需要连接的点的序号，连接这些点即可以构成面。

类比于之前完成的平面仿射变化算法，我们可以将控制点理解为标记点，将顶点理解为像素点。DDF算法提供了一个从旧的坐标到新的坐标的映射。

我们可以得到一个由顶点控制位置，由面控制拓扑结构的三维物体。在产生形变的时候，我们维持拓扑结构不变，只改变点的坐标，这样的形变才是合理的。

二、算法及实现逻辑

本部分介绍代码的实现逻辑。

前期准备及转换逻辑

荀子曾说，“吾尝终日而思矣，不如须臾之所学也”，当我们在网上搜索FFD相关算法时偶然找到徐辉学长去年的代码，拜读后深觉有用，于是参照其完备的交互思路，仿照其代码框架，继续我们的探索。

- 控制格点网络生成：

  我们对一个3D模型通过一些控制点来进行自由变换操作，FFD将控制点构成三维格点坐标系，三个坐标轴由向量s, t, u表示，其中每条坐标轴上控制点个数为l, m, n，这些点分别在s, t, u三个方向将坐标轴分出 l+1，m+1，n+1个样条。此时，坐标系中的每一个格点对应一个固定的“局部坐标”。可以通过可以拖动的球体传输形变控制点的位移；

- 临接球相连：

  将邻近的球体链接起来，形成可视化形变场，同时触发回调函数；

- FFD：

  生成一个新的多面体加入渲染，完成更新。另外，创建的点通过位置索引，完成更新操作。

  （此过程将在下方详细讲解）

FFD过程

- 三元Bernstein张量进行deform：

  形变过程本质上就是求控制点的加权和，即对任一变形后的点的绝对坐标为：

  $s(s,t,u)=\sum_{i=1}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}B_{i,l}(s)B_{j,m}(t)B_{k,n}(u)P_{i,j,k}$ （1）

  - 函数s：包含变换位置后的点坐标；
  - $B_{\theta,n}(x)$：n阶Bernstein多项式，$B_{\theta,n}(x)=\frac{n!}{\theta!(n-\theta)!}x^{\theta}(1-x)^{n-\theta}$，主要用于设置控制点坐标对变换坐标的影响；
  - $P_{i,j,k}$：控制点位置。
  -

- 算法改进：

  这种求加权和的计算时间复杂度是O(n^3)，且当l，m，n越大，耗时越多，于是我们分别采用2种方法对其进行优化：

  1. map函数法：

     FFD算法复杂度较高，因此我们需要进行一些代码优化以及矩阵化操作。我们首先尝试了简单的循环，之后尝试了python自带的map函数，最后使用了numpy进行矩阵运算，发现对于性能效果提升显著，以下为对比结果

  2. 矩阵法：

     由于（1）是一个关于P的线性方程，我们可以将其写成矩阵形式：

     S = BP

     - $S \in \mathbb{R}^{N*3}$ ： 三维坐标系中的格点坐标；
     - $B \in \mathbb{R}^{N*M}$ ： 形变矩阵；
     - $P \in \mathbb{R}^{M*3}$： 控制点坐标；

  （其中N，M分别为变形点和控制点坐标的编号）

l = m = n	map算法	矩阵算法	原始算法
2	0.3s	0.25s	0.35s
4	0.6s	0.28s	0.65s
6	1.45s	0.34s	1.3s
8	2.50s	0.50s	2.6s

可以看出，2种算法都将计算复杂度大大降低，而在格点较多时，矩阵算法表现更优。

播放动画

根据VTK特性，对于动画支持不友好，不能直接读取动画格式文件。因此我们选择将动画分解为许多帧，每次刷新渲染器中的物体，从而做到播放效果。

四、GUI使用说明与应用介绍

GUI开发情况

1. 开发环境：基于 python3.5 + pyQt5 + VTK。
2. 由于Mac OS系统不支持使用 pyQt + VTK 的 GUI ，暂仅能在Windows系统上实现拖动操作。
3. 实现步骤：
   - 使用了qt creator 设计界面，并转化为python代码（转化的代码为ui.py文件）；

    - 在ui.py中完成 connect 按钮和 main.py 中的 MainWindow 类函数。
4. 参考GitHub上的 pyQt + VTK 的框架代码（详见参考文献）。

界面介绍及使用手册

我们的应用软件左边为功能选项按钮，右边为可视化界面。

左边按钮从上而下分为三个部分：

- 第一部分有关于视角：选择看哪一个平面，Show scene按钮是是否展示右边的交互界面。
- 第二部分用于读取文件：选择obj文件地址，选择FFD文件地址，还原所有的控制点的位置，加载FFD文件，保存FFD文件，加载OBJ文件。
- 第三部分用于动画演示：有下一帧(next one)，上一帧（last one）以及返回第一帧（reset）。

右边为可视化互动界面，可以通过拖拽控制点来控制物体形变，以及拖动旁边背景转换视角。通过Show scene按钮开关。

使用步骤

软件打开时默认打开一个示例物体，之后需要切换物体。

需要切换物体时，我们必须首先点击choose obj path，之后才可以进行load obj操作。顺序不对会出现错误。同理，更换和保存FFD文件时也需要先指定文件地址（choose FFD path）。其中，load obj可以读入两种格式的文件，第一种是文本txt格式文件，这是为了动画准备的，我们需要将动画里面的每一帧按顺序存放，每行一帧。另外，我们也可以读入单个obj文件。值得注意的是，在windows系统下，由于VTK对中文支持不友好，地址中不能存在中文。

可能存在的一些问题：

- 由于QT的问题，我们的程序在Windows10下会出现字体大小问题，整个界面会有部分重叠，主要是Windows自带的放缩造成的。
- 每次使用QT按钮时，图像有时不会立即刷新，我们需要将焦点重新定位于VTK之后才可以刷新图像。
- 中文目录将出现不可预知的错误。

五、参考文献

1. https://github.com/Anthony-Xu/3D-FFD-in-VTK
2. Jack D, Pontes J K, Sridharan S, et al. Learning Free-Form Deformations for 3D Object Reconstruction[J]. 2018.
3. https://github.com/spherik/pyQtVTKSkeleton