本工程完全由本人自行编写，并且提交到了github中：https://github.com/lanpokn/mesh_segmetation。我可以证明该github账号为本人所有。该作业未向任何人分享，如有雷同则为他人网上借阅所致。

原理说明

该工程主要的算法原理都来自于论文Hierarchical Mesh Decomposition using Fuzzy Cluster，其中已有的原理不再重复叙述。该工程完成了**层次化 k 路分解**，其中层次化完全是代码层面的技巧，将在下一节介绍。

具体部署中出现了一些比较细节的问题，在此处会进行一些原理上的说明。首先，该工程**以python语言为基础，使用open3d**进行点云的读取与展示，在生成dual graph时首先使用了networkx存储graph。但是在后续测试中，networkx的dijistra算法过于缓慢，无法处理大型图形，因此后续使用了**igraph库**，可以让时间复杂度在可接受的范围。

同时，代码中有一些针对速度优化的部分。比如生成dual图是一个高重复度，高并行的任务，本工程使用了**cpu级别的并行**处理该任务，将该算法运行提升了3-4倍左右。此外，寻找dual graph中与所有mesh距离最近的点，在大模型中几乎是不可能完成的复杂任务，因此我将其抽象距离某几个随机点的距离和最小，让其时间复杂度在可接受范围内。

最后，原论文在计算面片间距离权重时，对凹凸性有不同的判断，但是open3d没有凹凸性判断的能力。（实际上open3d几乎只负责读取和展示）为了解决这个问题，我首先**借用openmesh生成标准的法向量**（保证法向量指向物体的外部），然后对于相邻的两个面片，计算其中一个面片法向量射线o+td交于另一个法向量时的t值，如果t为正则为凹，否则为凸。如此一来，整个算法便没有问题了。

值得注意的是，对模糊区域进行min-cut经常出现各种fail，原因很可能是原文的k分割就无法保证连通性，模糊区域可能出现各种不适用最小割的情况，这大大影响了该算法的稳定性。解决这个问题的最好办法是取e=0，即不要再考虑模糊区域的计算。经过实验验证，这样分割的结果也是很优秀的，因此后续如果没有说明的话，**默认取e=0**.

代码结构说明

此处简化版总结可见readme.txt

工程结构

trans to english:data文件夹存储输入的ply数据

fig文件夹存储输出的结果

doc文件夹是各种相关的文档

src文件夹是源码所在地（重要）

代码说明

src中，mesh_io.py是用于点云读入与展示的文件 main.py是运行示例（根据需要更改ply文件路径）segmentation.py是最重要的，其主要逻辑如下：

```
def Segementation_recursion(self,recursion_time = 2):
```

该函数是入口函数，recursion_time是指递归的次数，取1是不递归，只运行以此，其返回可用于分割后mesh展示的数据，其调用下列函数：

```
def Segmentation(self, triangle_indices_list):
```

该函数也是包装函数，他对于list中所有的子mesh都进行分割，并返回粒度更细的mesh，他调用

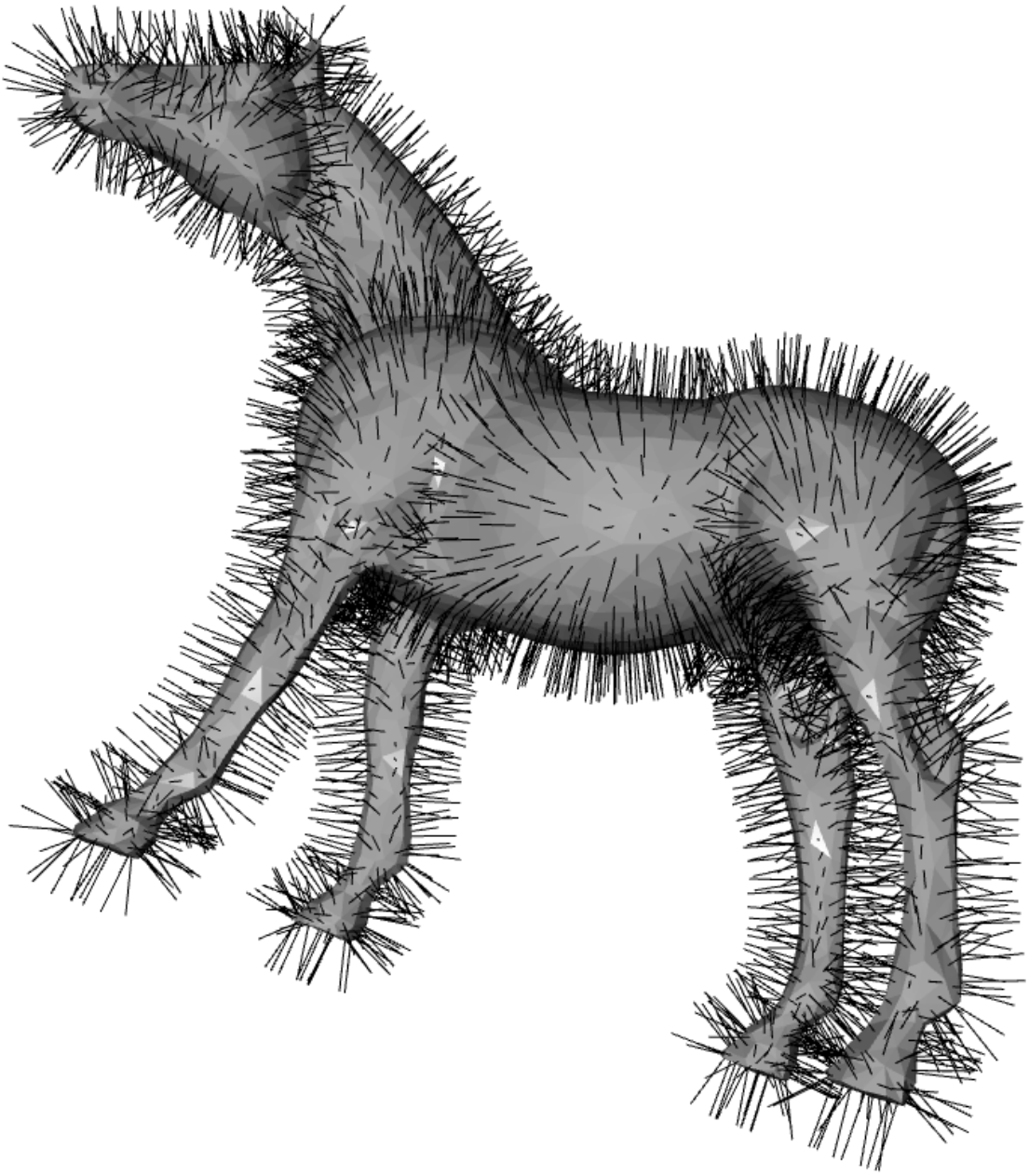
`def Segmentation_Base(self,triangle_indices):` 这是最底层的分割，输入就是一个mesh的，输出是对这个mesh的分割，它的核心是：

`seed_node_list = self.Determine_kseed(triangle_indices)`

`new_group_indices = self.Calculate_groups(seed_node_list,triangle_indices)`

第一行确定k个种子（同时确定k数和种子位置），第二行根据种子情况进行mesh分割。至此层次化分解的逻辑结构便清晰明了。具体细节可见源码。

其中，我半原创的凹凸性判断代码在Calculate_groups中 。其灵感主要来自于图形学中的ray，在normal都朝向物体外侧的前提下，用参数化直线的结果t判断两个面的关系。法向量的正确性已经经过了验证，如图：



运行结果分析

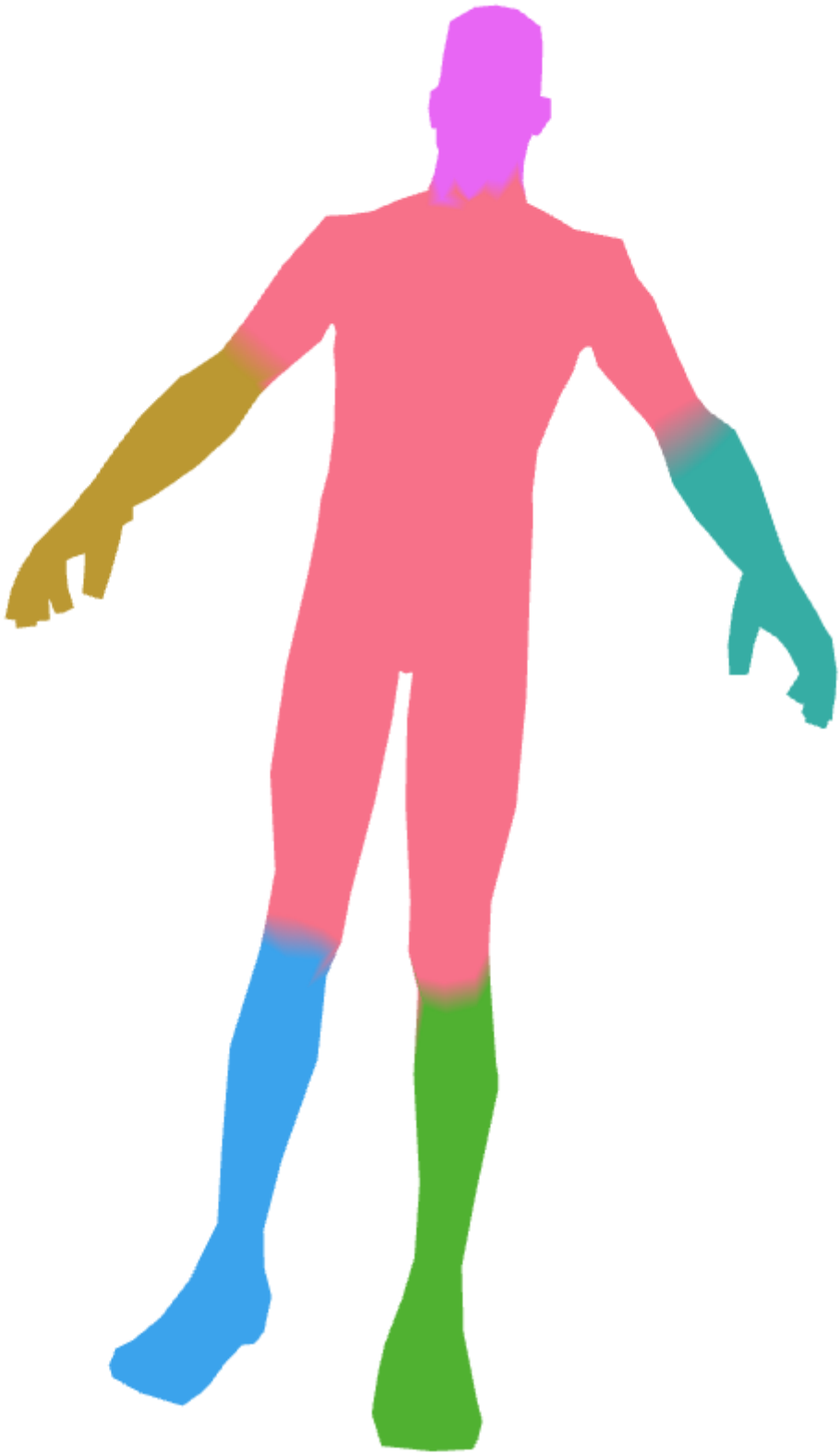
运行时间：

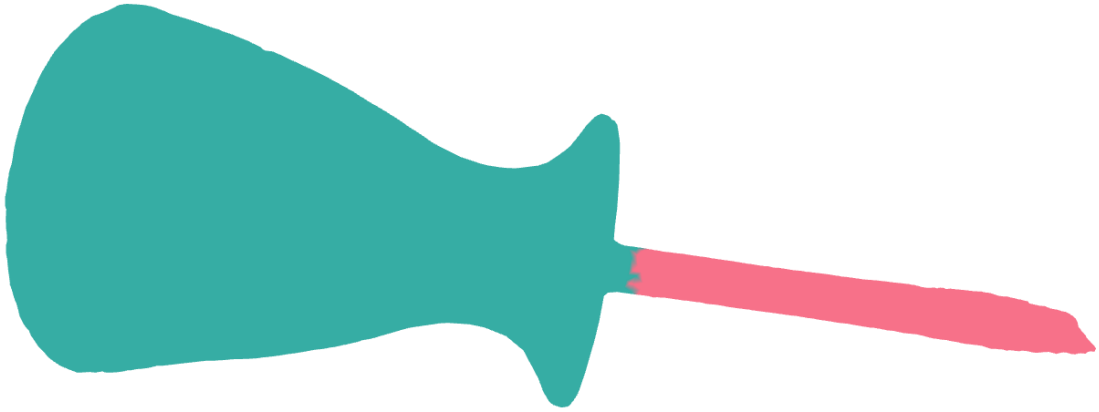
2000mesh左右需要10s（K=6分割）

4000mesh需要30-40s

运行结果：

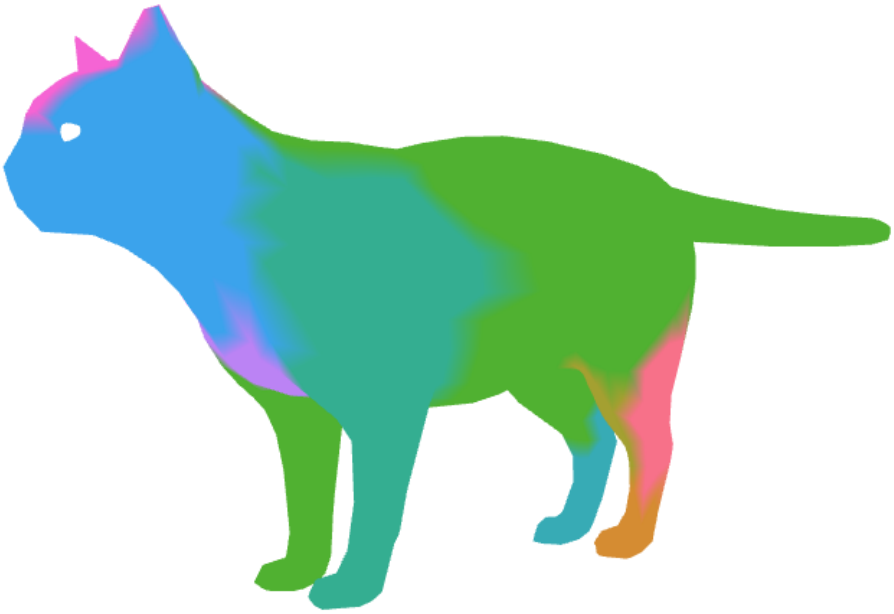
单纯K分割：





可以看到对于这种结构比较明显的mesh，一次K分割一般可以分的比较好

层次化K分割：



可以看到经过层次化K分割，更多的细节被分割了出来。不过由于第二次分割没有第一次的全局信息，这样往往不如第一次分割时强制取高K值的效果好。但是其优点的是更快（因为子mesh节点数量大幅度下降）

复杂度分析

复杂度分析比较容易，只需要考虑最高项。dijistra时间复杂度为O（（V+E)logV),V为mesh数，E为边数（两个mesh相邻则有一条边）在种子的生成中需要遍历整个dual图，即使忽略边数，其仍有复杂度为O(V^2log(V))。（注意如果不简化第一个种子的选取，严格按照原论文执行，则复杂度变为O(V^3log(V))，这几乎是令人难以接受的）。

其他的子算法复杂度都不高于O(V^2log(V))，因此可以认为总的复杂度为O(V^2log(V))，这与上一节的结论也基本一致。