biblatex Collection.bib

Mesh Segmentation with Concavity-Aware Fields

Oscar Kin-Chung Au, Youyi Zheng, Menglin Chen, Pengfei Xu, and Chiew-Lan Tai

摘要

本文提出了一种简单高效的自动网格分割算法，该算法仅利用网格形状的凹陷信息。该方法使用一组对凹陷感知标量场来定位凹折痕和接缝。这些场是通过使用新颖的凹度敏感加权方案求解拉普拉斯系统来计算的。从凹面感知场采样的等值线自然地聚集在凹面接缝处，作为良好的切割边界候选。此外，这些字段提供了足够的信息，可以有效地评估候选剪辑。我们对所有场梯度幅度进行汇总，为每条等值线定义一个分数，并采用基于分数的贪婪算法来选择最佳切割。广泛的实验和定量分析表明，我们的分割质量优于或可与现有的最先进的更复杂的方法相媲美。

关键词：凹陷感知场；网格分割

1 引言

将三维模型分成多个有实际意义的部分是几何处理的一个重要问题。许多网格处理算法，如网格编辑，形状提取，物体绑定等都需要网格分割作为预处理阶段。网格分割结果的好坏往往决定了后续网格处理的表现。比如，数字化技术在个性化医疗领域有着广泛的应用，尤其是数字化齿科。数字化牙齿正畸是在数字化牙颌模型的基础上，利用计算机辅助设计技术进行牙齿矫正，并规划出一整套治疗方案的过程. 它主要包括牙齿分割、牙齿修复、牙齿姿态调整、牙龈跟随变形、牙齿姿态调整路径插值、虚拟咬合、附件安装及矫治方案的制定等。其中，牙齿分割是整个正畸系统的第一步，是牙齿正畸的基础。因此, 如何快速准确地将牙齿从整个牙颌数据中单独分割下来将直接影响牙齿正畸的效率及效果。

2 相关工作

现有的网格分割算法可以根据它们要分割的模型类型分为两类。第一类包括设计用于分割 CAD 模型的方法，主要用于逆向工程目的。他们将模型分割成小块，每个小块都是从一些预定义的数学表面（例如圆柱体、球体和平面）中选择的最佳选择。第二类分割算法旨在将有机形状分割成语义部分组件，符合人们对物体各部分的理解。我们的方法属于第二类。[1]

2.1 交互式的分割方法

用户的分割意图，也可以通过交互来体现。比如，用户想分割出兔子的头，在兔头上画了一条蓝色的线，在兔子身上画了一条红色的线，得到了一个分割结果。这个结果不是很理想，于是用户再用红色的线标记了兔子身体，得到了期望的分割结果。交互的方法，需要尽量的简单，并且分割结果要进来的准确。这类分割问题一般采用图割的方法来分出前景和背景区域。

图 1: 交互式分割

2.2 非交互式分割方法

基于非交互式的分割方法有区域生长，分水岭分割，迭代聚类等。区域生长的方法将网格分解为不是严格的凸多边形区域，每个区域均以种子顶点或多边形开始，并以聚簇多边形的数量增长，直到满足停止条件为止。分水岭分割算法将给定的网格分解为“集水盆地”或“分水岭”，分流方法有两种：自下而上和自上而下。自下而上的方法（也称为洪水）旨在寻找集水盆地如下，它以局部最小值开始洪水过程，并逐渐淹没该区域，并在洪水溢出其集水盆地时立即停止；第二种是自上而下的方法，它涉及在特定点放置令牌并将令牌沿最陡峭的下降方向移动，直到达到最小值或先前与最小值相关联的点为止。迭代聚类分割算法建立在著名的 k-means 算法的基础上，首先从一组 k 个代表性种子开始，以生成 k 个区域（或片段），迭代分割，同时至少一个新的或更新的区域代表性种子保留在一系列迭代中，最终收敛。

3 本文方法

3.1 本文方法概述

通过凹性切割场来提取物体上的凹陷区域，将凹陷区域的等值线作为潜在的分割线候选集，利用物体的全局信息来确定最终的分割线。此次复现并非完整复现该论文，而是将该论文的方法用于三维的牙齿模型下。由于牙齿模型的复杂性，文章中在候选分割线集中选最终的分割线并不适用于牙齿，故接下来并不介绍此部分内容。

3.2 凹性感知切割场

我们的方法依赖于分割字段的几个关键属性。具体来说，我们要求场足够平滑并且对曲率敏感，以便可以在凹区域（所需的分割切口所在的区域）对场的更密集的等值线进行采样。我们将分割场定义为调和场的变体，它是通过求解泊松方程 $\Delta\Phi = f$, 计算的，其中 Δ 表示离散拉普拉斯算子。[2]等价为求解 $A\Phi = b$

$$\mathbf{A} = \begin{bmatrix} \mathbf{L} \\ \mathbf{C} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B} \end{bmatrix}.$$

L 为拉普拉斯矩阵，B 和 C 分别为代表边界条件 (即切割场的前景点和后景点) 的向量和矩阵

$$\mathbf{L}_{ij} = \begin{cases} -1 & \text{if } i = j \\ \dfrac{\omega_{ij}}{\sum_{(i,k)\in\mathbf{E}} \omega_{ik}} & \text{if } (i,j) \in \mathbf{E} \\ 0 & \text{otherwise,} \end{cases}$$

权重设置：

$$\omega_{ij} = \frac{|e_{ij}| \cdot \beta}{G_{ij} + \epsilon},$$

若顶点 i 或 j 中有一个是 concave 点，则 β 设置为 0.001，否则为 1。判断凹（concave）点的方法：对于顶点 i，若它与其任意相邻顶点 j 满足下面方程，则为凹点。

$$\frac{(v_i - v_j)}{\|v_i - v_j\|} \cdot (n_j - n_i) > \varsigma.$$

3.3 基于 harmonic field 的牙齿分割

基于每颗牙齿牙冠部分的龈缘线存在的凹性特征，计算相对应的切割调和场 (0 - 1)。该调和场为凹点赋予较低的权重，从而使得切割场能在牙齿的龈缘线处迅速衰减。然后，选取特定的阈值（0.75），选取大于该阈值的部分作为最终的分割结果。

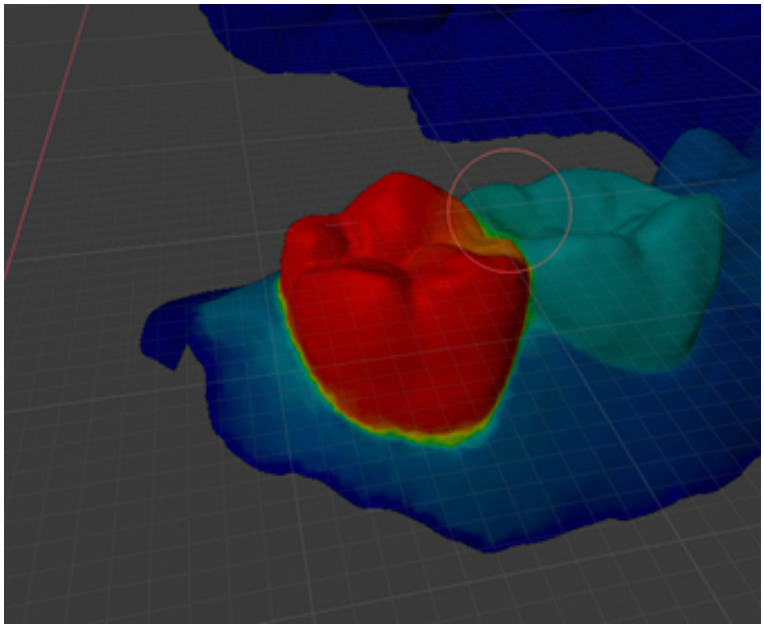

图 2: harmonic field

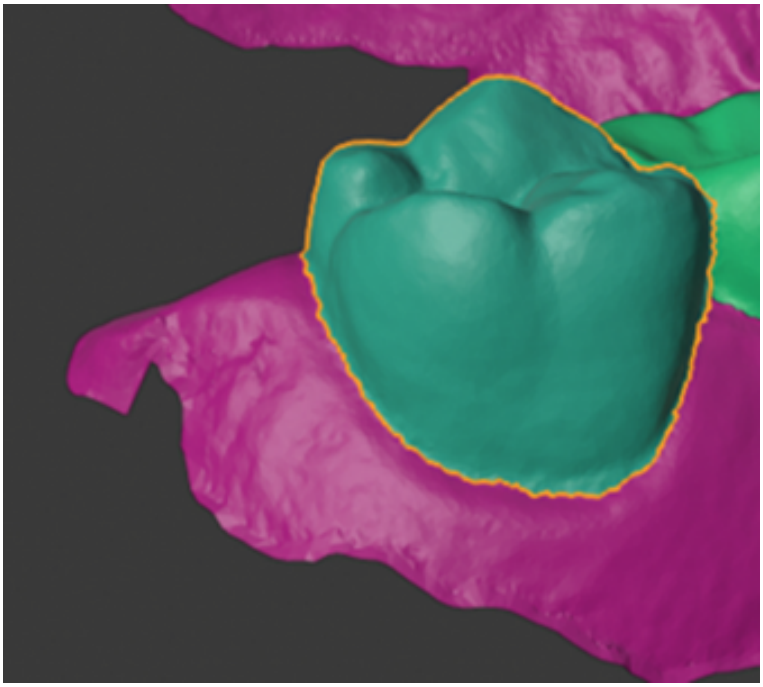

图 3: 单个牙齿分割结果

4 复现细节

4.1 代码介绍

本论文无相关开源代码，各模块基本由自己独立完成。一些具体的小细节参考过部分网上他人分享的代码。

简单介绍两个重要的功能：凹陷点的判断以及拉普拉斯矩阵计算

凹陷点判断对每个顶点，根据顶点的距离建立一棵 KDD 树，从而获取每个点的最近的若干零点。根据上文介绍的判断凹陷点的方法，确定顶点是否为凹陷点。

```python
def concave_list(obj, num):
    #obj = bpy.context.active_object
    mesh = obj.data
    size = len(mesh.vertices)
    kd = kdtree.KDTree(size)
    theta = 0.01
    concave = [False] * size
    for index, vertice in enumerate(mesh.vertices):
        kd.insert(vertice.co, index)
    kd.balance()
    for index, vertice in enumerate(mesh.vertices):
        co_find = vertice.co
        co_find_norm = vertice.normal
        for (co, id, _) in kd.find_n(co_find, num):
            co_norm = mesh.vertices[id].normal
            dif_co = co_find - co
            dif_co = np.array(dif_co)
            dif_co_unit = dif_co / np.linalg.norm(dif_co)
            dif_norm = co_norm - co_find_norm
            # normals have normalized
            #dif_norm = np.array(dif_norm)
            #dif_norm_unit = dif_norm / np.linalg.norm(dif_norm)
            #res = np.dot(dif_co_unit, dif_norm_unit)
            res = np.dot(dif_co_unit, dif_norm)
            if(res > theta):
                concave[index] = True
                break
    return concave
```

图 4: 凹陷点判断

拉普拉斯矩阵计算该功能主要实现如何获取矩阵中非 0 元素的行列以及对应的元素值。遍历每个顶点的 one-ring，根据权重公式计算这两个顶点间的权重，同时分别保存中心顶点的行以及邻接顶点的列。

```python
def get_concavity_aware_laplacian_matrix(obj,bm):
    curvatures = mean_curvature( obj, bm, method = 'MINMAX')
    is_concave = concave_list(obj ,5)
    gama = 0.0001
    beta = 0.000001
    row_indices = []
    col_indices = []
    weight = []
    for vert in bm.verts:
        pv = vert.co
        vid = vert.index
        for edge in vert.link_edges:
            vvert = edge.other_vert(vert)    # Return the other vertex on this edge
            ppv = vvert.co
            vvid = vvert.index
            elen = (ppv - pv).length
            w = elen / (abs(curvatures[vid] + curvatures[vvid]) + gama)
            if is_concave[vid] or is_concave[vvid]:
                w *= beta
            row_indices.append(vid)
            col_indices.append(vvid)
            weight.append(-w)
    diag = [0] * len(bm.verts)
    for i in range(len(weight)):
        diag[row_indices[i]] -= weight[i]
    for i in range(len(diag)):
        row_indices.append(i)
        col_indices.append(i)
        weight.append(diag[i])
    #weight = np.array(weight)
    return row_indices, col_indices, weight
```

图 5: 拉普拉斯矩阵计算

4.2 实验环境搭建

本代码基于 blender3.0 平台，安装相应的库环境可以直接运行于该平台

4.3 界面分析与使用说明

开始脚本编辑器，打开相应文件，选中牙齿模型，切换至编辑模式，选择待分割牙齿上的点，分别为两个侧面的对角点，取 4 个。然后运行脚本。切换至编辑模式，选中模型点击右键选择 separate by material，即可完成单个牙齿分割

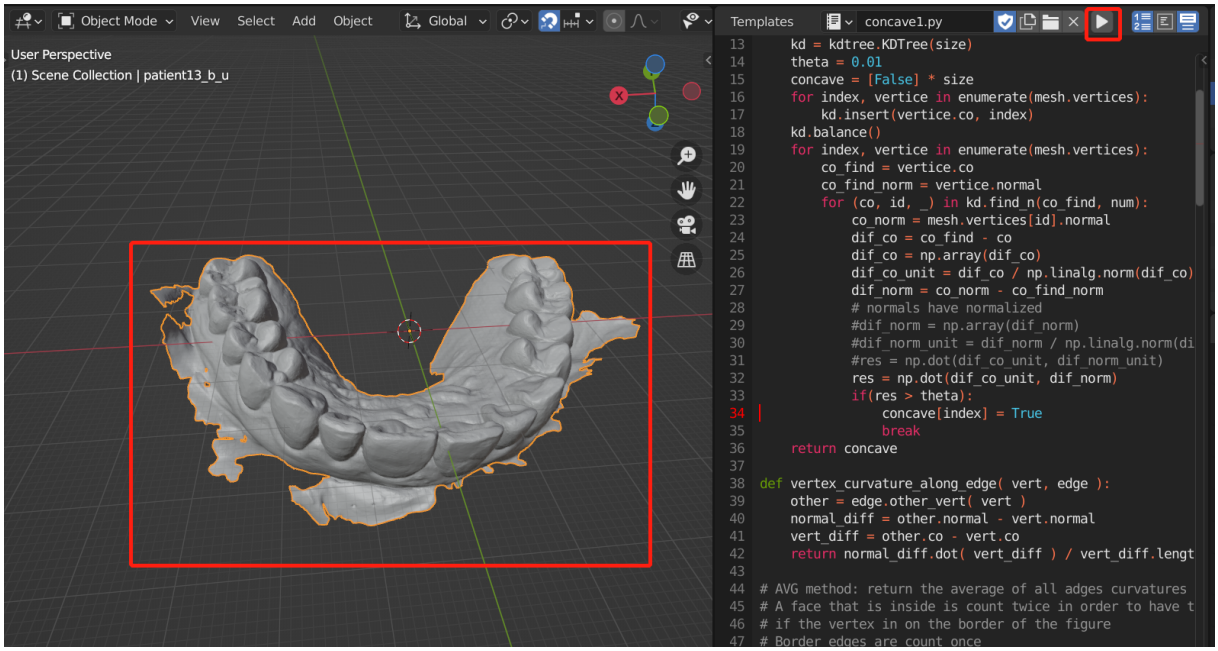

图 6: 操作界面示意

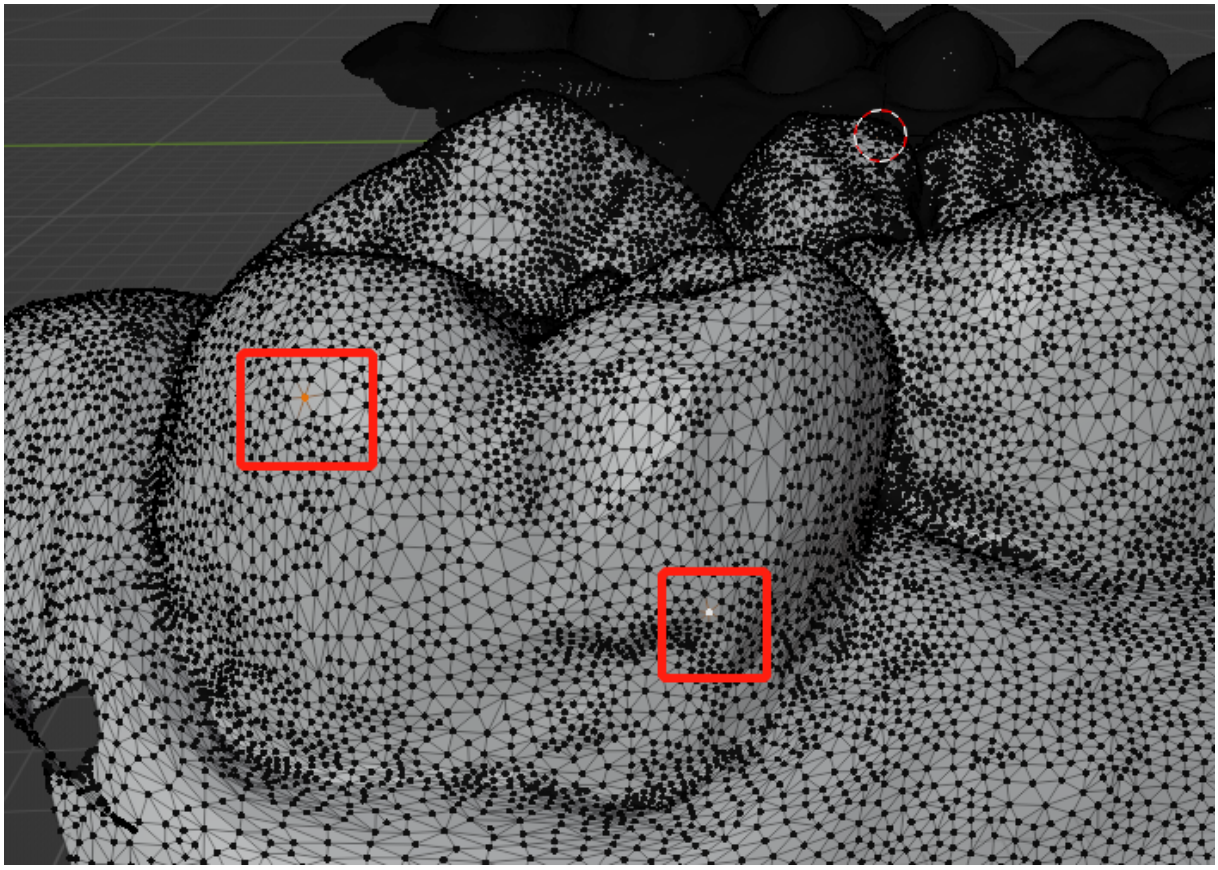

图 7: 一侧选点

4.4 创新点

1、原论文所使用的方法主要适合于闭合的网格模型，我将该方法用于非闭合的牙齿网格模型，并利用非闭合的网格模型存在边界这一特点，将网格边界点设置为背景点。因此只需要人工选取前景点即可，减少了选点工作的复杂性。

2、根据了牙齿咬合面存在大量凹陷点以及调和场本身的特性，提出了新的选点方案，对于不同牙齿的形状都有具有较好的分割效果。

5 总结

本次复现论文任务中，作者将凹陷切割场用于三维牙齿模型中，实验结果证明该算法对于分割边界存在明显凹陷区域的牙齿模型十分适用，只需要少数人为介入就可以实现较高质量的切割分割，提升后续的牙齿正畸的效果。目前，该算法仍存在部分缺陷，对于每一颗牙齿进行分割，需要计算一次切割场。因此，对整排牙齿的分割来说，效率较低。这个问题也是后续可以进行改进的点。

参考文献

[1] O.K.-C.Au. Mesh Segmentation with Concavity-Aware Fields[J]. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, 2011.

[2] O.K.-C.Au. Dot Scissor: A Single-Click Interface for Mesh Segmentation[J]. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, 2012.