牙齿模型的识别

杨国峰，张宏

摘　要：

分割技术的研究始于 20 世纪 70 年代，最初广泛应用于图像分割尤其是医学图像分割领域中，随着计算机图像处理技术以及三维网格模型的发展，分割技术的领域在不断地扩展，从图像分割领域到三维网格模型的分割，分割技术逐渐成为近年来的热点研究课题。三维模型分割技术逐渐被应用于牙齿正畸领域，用于辅助医生进行牙齿模型的解析，减少医生的重复性工作，帮助其快速准确地对不同的牙齿情形制定诊疗方案。本项目致力实现一个可自动对牙齿模型进行分割、编号、缺失识别和特征点提取的系统，该系统通过医生的简单交互，如点击按钮，即可显示牙齿模型的处理结果，供医生分析和使用。本项目以 VS 2017 + Qt5.13.2 + Vtk8.2.0 为开发工具，开发了分割、识别模块，通过解析输入的 STL 牙齿模型，可将处理结果渲染在屏幕上，并设计了简易的用户界面，增强了系统的可用性。

Identification of dental models

Guofeng Yang , Hong Zhang

Abstract:

The research on segmentation technology started in the 1970s. It was initially widely used in the field of image segmentation, especially in medical image segmentation. With the development of computer image processing technology and three-dimensional mesh models, the field of segmentation technology is continuously expanding. In the segmentation field to the three-dimensional mesh model segmentation, segmentation technology has gradually become a hot research topic in recent years. Three-dimensional model segmentation technology is gradually being applied in the field of orthodontics to assist doctors in analyzing tooth models, reduce doctors' repetitive work, and help them quickly and accurately formulate diagnosis and treatment plans for different dental situations. This project is committed to implementing a system that can automatically segment, number, identify missing teeth and extract feature points of a dental model. The system can display the results of dental model processing through simple interactions of doctors, such as clicking a button, for doctors to analyze and use. This project uses VS 2017 + Qt5.13.2 + Vtk8.2.0 as a development tool, developed a segmentation and recognition module, analyzed the input STL tooth model, rendered the processing results on the screen, and designed a simple user interface to enhance System availability.

1 简介与意义/Introduction

1.1 项目意义和依据/Significance

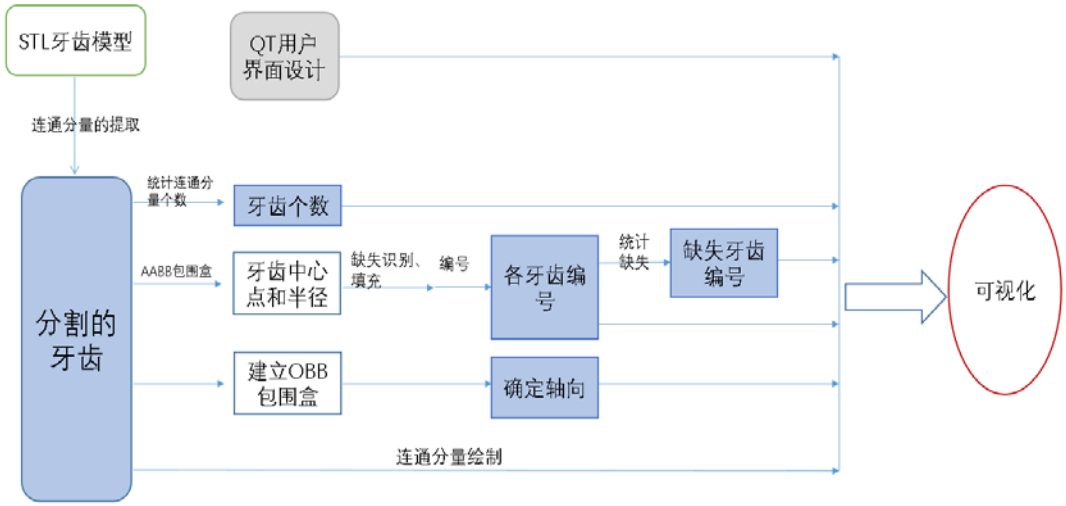
口腔疾病是一中常见的多发性疾病。计算机技术是口腔医学等医学领域的一个重要组成部分。随着计算机软硬件技术、三维数字化成像技术的飞速发展，图像处理技术逐渐应用到口腔医疗领域，用于辅助医生进行牙齿的诊断和矫正。在数字化正畸学科中，医生通过三维扫描设备获取到患者牙齿的模型后，通常需要对获得的模型进行进一步的处理，比如需要将每颗牙齿从模型上分离出来，并根据位置关系对牙齿进行分别命名和排列，对各个牙齿的特征点（如牙尖等）进行标记。人工对牙齿进行分割和命名是一项相当耗时、机械的工作，而由临床医生手动标记每颗牙齿的特征点，成本很高并且较易出错。如果计算机能够自动识别输入的牙齿模型，并对牙齿模型进行快速的分割、命名、以及特征点的提取与标记，医生仅在后期进行确认和手动校错，那么将极大的节省医生的工作量，大大提高临床诊断和治疗方案设计的效率。

关于三维模型的分割和处理算法的研究有很多，常见的三角网格分割算法有分水岭法、区域增长法、Mean-Shift 法、聚类分析方法等。然而对于牙齿的三维模型而言，其高效分割和处理仍然是一大难点，主要原因有：第一，得到的牙齿模型往往不够理想。由于扫描设备的限制和患者口内的卫生状况等综合因素的影响，得到的牙齿模型往往不够精确，而后续的牙齿切分和处理高度依赖于牙齿模型的几何特征（如点、面、曲率关系等），导致最终分割和处理的结果受到影响，这就要求算法对数据要有较好的容错性和适应性。第二，实际的牙齿模型的情况是多样复杂的，不同患者的牙齿个数和形态都各不相同，必须设计一种通用的算法来处理所有可能的情况，包括牙齿缺失、拥挤和邻牙错位等，这对算法的设计提出了挑战。第三，牙齿特征点（如牙尖）的准确识别和标记较为困难，可能会过度划分（识别过多的特征点）的问题，也可能会出现漏掉特征点的情况，当结合曲率来识别特征点时还会容易出现识别错误，因此算法需要按照模型特征仔细设计。第四，处理过程涉及大量的数据访问和计算，算法的时间效率优化是一项很大的挑战。

本项目在阅读相关文献、结合前人研究的基础上，吸收经验和教训，利用 Vtk 的库函数，设计出了一套完整的牙齿模型处理系统，该系统运行后可以快速的完成 STL 牙齿模型的读入、分割、编号、牙尖识别、缺失牙齿识别的功能，并将处理后的牙齿模型以及结果清晰美观地显示到计算机屏幕上。借助 Vtk 的连通分量函数，可实现对 STL 牙齿模型连通分量的提取，即完成了对各个牙齿的分割工作；之后通过各个牙齿的 AABB 包围盒顶点坐标确定牙齿的中心点坐标和半径，用于后续牙齿的编号和牙齿缺失识别的计算；利用 Vtk 工具函数建立各个牙齿的 OBB 包围盒，通过 OBB 包围盒的轴向，即得到了牙齿的牙轴方向；之后利用 Vtk 的工具函数实现结果的可视化。本项目实现的系统运行快速，算法简约，对不同牙齿模型情形的普适性强，并利用 Qt 设计了良好的用户交互界面，医生可以通过界面的按钮，来决定显示的内容，便于医生按需进行观察和分析，提高了系统的实用性。

1.2 本方法/系统框架/Article Structure

本项目系统设计的框架如下：



读入 STL 牙齿模型，通过连通分量的提取得到分割的牙齿。根据连通分量的个数得到牙齿的个数；计算牙齿中心点坐标和在 XOZ 平面上的投影半径，进行缺失识别和编号；建立 OBB 包围盒确定牙齿轴向；最后结合 Qt 设计用户界面，将所有的处理结果和整个牙齿模型可视化输出到屏幕上。

2 相关工作/Related Works

2.1 三维牙齿模型的获取

三维牙齿模型数据的获取是整个数字化口腔工作流程的第一步，数据的质量、精度直接影响着后续的数据处理流程。随着立体影像，激光扫描等技术的成熟与普及，三维牙齿模型数据的采集过程正朝着精确化和轻量的方向发展。

文献[2]介绍了目前常见的三种三维模型数据获取方式：石膏模型扫描、工业 CT 扫描或 CBCT 系统和口内扫描。文献[7]介绍了国内研制成功的 CXM-I 型牙颌石膏模型层析设备和 Auro-350 光固化设备，用于牙齿矫正系统。文献[22]介绍了基于线结构光扫描牙齿模型获取三维点云信息从而进行三维重建的方法，构建出一套完整低成本、可移动的便携式三维牙模扫描系统。

2.2 牙齿模型的预处理

在数字化口腔领域，由于取模条件的限制，扫描仪的精度以及牙龈和牙缝的遮挡等多种因素，通过扫描等数字技术得到的牙齿模型往往不够理想，导致进行分割之后得到的牙齿模型会在侧面和底部产生网格缺失，这样的网格缺失将不利于后续排牙和评估等操作，从网格质量的角度看，模型也需要满足流型的要求，所以需要通过模型修复去除模型中存在的一些缺陷，提高模型质量从而得到一个适用于后续处理的光滑而完整的输出模型。目前解决牙齿修复问题最常用的算法就是基于网格的修复算法，其中最具代表性的就是文献[17] Liepa 提出的补洞算法，该算法在网格修复领域十分经典，后续的补洞算法大多基于这个算法的思想进行改进。除此之外，文献[18]中 Pernot 等提出先对网格边界进行一定的预处理，然后在孔洞内部生成新的网格拓扑，最后利用变形的方法实现拓扑网格与孔洞周围网格的曲率偏差最小化，从而完成孔洞的修补。文献[19]袁天然等提出一种综合性的牙齿形状修复建模方法，根据牙齿的生理医学特征，以牙齿之的边界信息为约束条件，通过将曲线能量函数最小化的方法对“中间”恢复曲面进行变形调整，从而修复单颗牙齿缺失的部分。文献[21] X ia 等提出了一种基于牙齿模板的修复算法，他们利用 ICP(Iterative Closest Point)算法来进行原始模型和模板的匹配，然后基于形变操作完成对原始模型的修复。

2.3 三维网格分割

2.3.1 三角网格分割方法的发展

目前国际国内对三维网格模型分割问题的研究，多数是面对具体应用问题，从图像分割技术推广而来。较早的三维网格分割工作可以追溯到 1991 年，文献[7]详细介绍了关于三维网格分割的研究的发展历程。

2.3.2 三角网格分割的主要方法

网格分割（也叫网格划分）算法是计算机几何建模和计算机图形学任务和应用程序的关键算法。网格分割有助于参数化，骨骼抽取，碰撞检测，纹理映射，形状匹配，形变，多分辨率建模，网格编辑，几何压缩和动画等工作。除此之外，网格理解和基于语义的对象表示一般也依赖于表示这些对象和形状的 3D 网格的结构提取和特征识别。比如图像分割、有限元网格划分和无监督的机器学习等领域。在多种应用中，需要一个有效且自动的网格分割算法。

通常决定使用哪些元素作为属于同一部分的标准和分区过程中施加的约束条件对于分割是非常重要的，而这些标准通常是通过网格的属性进行提取，一般而言可以作为标准来进行的分割的网格属性主要有八种：平坦度、法线和二面角、曲率、平均测地距离、对称性、凹凸性、中轴、形状直径函数。

文献[1]对包含上述若干种的常用网格属性的概念及分割用法做了详细的阐述。详细介绍了目前通用的几种网格模型分割算法，有区域增长算法、分水岭算法和聚类算法，并对每种算法进行了实验和测试，评估算法性能。文献[7]补充了基于 3D 蛇形的智能网格分割方法，较为新颖。

2.4 牙齿分割提取

首先需要分开牙齿模型的牙龈和牙齿部分。文献[7]介绍了目前牙齿矫正过程中主要的牙冠与牙龈的分割技术，有边界识别法、矢量逼近法和三维交互标记控制法，并对该三种方法进行了分析和比较。

分离和去除牙龈部分后，对于如何将整口牙分割成单独的牙齿，以供医生在电脑屏幕上对牙齿的位置进行分析和重新排列，文献[20]KW 等提出了一种基于形态学骨架的牙齿分割方法，通过改进的形态学骨架算法，计算出单顶点的牙齿边界，通过骨架线描述了牙齿与牙齿之间之间的关系，之后通过自动的算法对邻牙进行分离。文献[15]提出了一种基于网格抽取的牙齿模型快速分割算法。文献[3]林海、宁楠等设计了一种基于特征识别确定牙齿边界，通过预分割和改进的 Isophotic 度量的区域增长算法进行牙齿分割的方法，实现效果较为理想。

2.5 牙齿特征点的识别

牙齿表面显著隆起的似锥体是牙尖，牙尖最顶端的点是牙齿尖点。通常牙齿特征点识别就是识别这个尖点，这个尖点对于正畸治疗效果的评判和配准有着非对于正畸治疗效果的评判和配准有着非常重要的作用。对于牙齿特征点识别的算法，绝大多数本质都是对网格进行分割，正如前述的三角网格分割算法，然后识别每块区域最值点。文献[9]一种基于局部坐标的牙齿模型特征点自动识别算法。文献[1]提出了基于混合聚类的特征点识别，分模型的预处理、DBSCAN 聚类[6]和 K-means 聚类[6]三个阶段，对不同牙齿的牙尖识别取得了较为精确的成果。

3 研究内容与方法(或算法)/Contnts and Methods(or Algorithm)

本项目得到了经过扫描和前期处理完毕的 STL 牙齿模型文件，由于模型得到了很好的前期处理，因此较为精确、光滑，本项目的工作只需直接读取和处理这些 STL 模型文件。

本小组一开始计划使用 VS2017+OpenGL 库来进行开发。编写 C++程序，直接读取给定的 STL 模型文件，将文件各面片信息存入定义好的数据结构中。由于不同的牙齿之间没有公共边，可采用广度优先搜索的方式对牙齿进行聚类来分割：任意选择一个面片，采用广度优先搜索的方式，搜索与它有公共边的面片，所有相连（含直接相连和间接相连）的面片存入一个牙齿类中，已经遍历过的面片会被标记，当所有的面片都被标记完成后，标志着所有牙齿聚类完成。然后，对牙齿进行缺失识别、编号和牙尖识别，再使用 OpenGL 的函

数进行绘制可视化。牙齿分割算法复杂度为 n^2，时间性能较差，经过测试，含有 14 个牙齿（8 万个面片）的模型需要 40min 才能分割完毕。经过分析确定，导致算法时间性能差的主要原因是，寻找一个面的相邻面片，需要遍历搜索整个模型的所有面片，来判断两者是否相邻，而这种遍历是相当耗时的。由于后续的编号、缺失识别等工作均需要建立在牙齿分割的基础上进行，这要求设计一个高效的牙齿分割算法，故本小组随即放弃了这种方案。

之后，通过查阅资料，了解到了强大的 Vtk 库，转而计划利用 Vtk 库来进行系统的开发，并成功地实现了项目目标。借助 Vtk 库和 Qt，在 VS 平台上设计了一套完整的牙齿模型识别和处理系统。在该系统中，可以直接读取整口牙的 stl 文件格式的数据并进行绘制，可以实现去除牙龈、连通分量提取、牙齿编号显示、牙齿计数、牙齿缺失情况识别、牙齿轴向绘制等相关功能。接下来将详细介绍项目实现的方法和过程。

3.1 Vtk简介

Vtk 的全称是 The Visualization Toolkit, 是一种面向对象的 3D 图形的渲染工具，是对 opengl 的封装。Vtk 系统主要由 C++类库、解释包装层两个基本子系统构成。Vtk 是一个基于面向对象的系统，因此提高 vtk 开发效率的关键因素就是建立一个好的、易于理解的、优化的对象模型，而本小组的选题就是“牙齿模型的识别”，可以将牙齿模型视为对象模型进行处理，因此本小组选择了 vtk 作为开发工具。

Vtk 是一个免费、开源的软件开发包，主要用于计算机图形学、图像处理和可视化等方面。Vtk 是采用面向对象技术设计和实现的，内核是用 C++构建的，包含大约 250000 行代码，2000 多个类，提供了丰富的可视化方法。Vtk 以灵活性和方便性为主要开发原则，具有如下几个特点：具有强大的三维图形显示功能。Vtk 既支持基于体素的体绘制法，又保留了传统的面绘制法，从而能够在最大限度的改善可视化效果的同时，又充分利用了现有的图形库和图形加速硬件。vtk 的体系结构具有强大的流处理和高速缓存能力，在处理的数据非常大时，不会受内存资源限制的影响。vtk 能够很好地支持网络工具的开发和应用。vtk 具有设备无关性的特征，使得用其开发的代码具有良好的可移植性。vtk 中有许多宏定义，这些宏极大地简化了编程工作，并且加强了一致的对象行为。vtk 具有更丰富的数据类型，具有多种数据类型的处理能力。vtk 具有跨平台的特性，既可以工作于 Windows 操作系统，也可以工作于 Unix 等其他操作系统，极大地方便了用户。

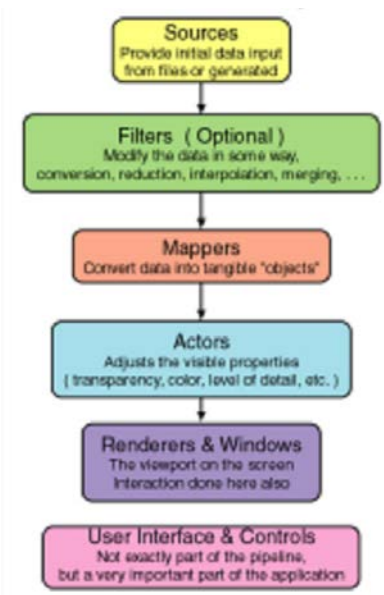


图 1 VTK 可视化流水线

Fig.1 Vtk Visualization Pipeline

本文所使用的 vtk 使用的可视化管线结构如图 1 所示，主要包含以下的类：vtkObject 类是 VTK 类库中

的基类，提供基本方法给可视化管线；vtkSource 类用于创建和读取数据，提供 vtk 的数据源；vtkFilter 类处理原始数据，使其能够被后续类直接运用；vtkMapper 类将数据转换为几何图元并进行渲染，它是数据与模型之间的桥梁，数据通过这种转换才能被直接地观察到；vtkActor 类将几何图元转换为实体以显示在绘制场景中；vtkCamera 类负责绘制场景中摄像机的位置，通过设置其参数可以控制观察的方式；vtkLight 类产生光照，只有当 vtkLight 类和 vtkActor 类同时存在且发生作用时，才可以通过 vtkCamera 类观察到模型；vtkRender 类创建绘制场景，为以上相关类提供展现平台；vtkRenderWindow 类指定 VTK 程序的绘制窗口。

数据对象是由几何和拓扑结构组成的，同时数据对象和属性对象是对应的。属性数据可以用来描述数据对象中的点集和单元集的属性，单元集是数据的基本组成单位。过程对象一般称为过滤器，按照某种运算法则对数据对象进行处理，对数据对象的数据进行优化，过程对象表现系统中的几何形状，数据对象和过程对象连接在一起形成可视化流水线。

源对象：源对象是没有输入并且有一个输出或多个输出的对象。

过滤器：过滤器对象有一个或多个输入与输出，依据类层次图可以确定过滤器输入输出的数据类型，类层次图才开发时是非常有价值的，它是可视化流水线架构的详细描述。

映射器：映射器对象有一个或多个输入，没有可视化数据输出，在 vtk 中有两种不同类型的映射器：图形映射器和复写器，图形映射器将数据对象的几何结构和属性数据映射到图形系统进行绘制，复写器将数据对象转成文件的形式存储。映射器对象，可视化流水线的终点，是图形模型和可视化模型之间的接口，其主要作用是将数据对象转换成图形对象，然后由图形引擎绘制出来。

3.2 Qt简介

Qt 是 1991 年开发的一个跨平台的、具有图形用户界面的、用于 c++应用程序的开发框架，它既可以开发 GUI 程序，也可以开发控制台工具、服务器等非 GUI 程序。Qt 是面向对象的应用程序开发框架，采用组件编程，使用大量的宏定义，容易扩展。同时 Qt 具有跨平台、面向对象。提供大量 API、支持 2D/3D 图形渲染，以及开发文档丰富等特性。Qt Creator 是用于 Qt 开发的，轻量级跨平台集成开发环境。

3.3 系统需求分析

本文所研究的牙齿模型识别系统是面向口腔科医护人员的一种软件，功能明确，操作简单，人机交互性强，便于医护人员的使用。在传统口腔科临床诊断中，医护人员对于扫描到的整口牙的数据的处理，必须进行手动的标记才能进行分割出单个牙齿，然后再人工进行编号，而且对于牙齿轴向的确定也不是很精确，更是增加了误判的可能性。为了克服这种传统方法中的缺陷，本文开发的系统实现了自动的分割和识别功能，在系统界面医护人员只需要点击按钮即可实现预期效果，极大地减轻了口腔科医护人员的工作负担。

牙齿自动编号是系统的核心和基础，因为在正确编号的情况下也就为后续的牙齿计数、牙齿缺失识别等相关工作打下了坚实的基础。同时牙齿自动编号也是医学图像可视化技术在本文所研究系统中的重要体现。在保证牙齿分割和编号结果正确的前提下，考虑到医护人员需要多角度观察口腔结构，系统可视化过程中的实时性尤其是一个重要的考量。本系统借助 vtk 这个强大的 3D 渲染工具实现了准确性和实时性的完美结合。

3.4 系统结构设计

根据系统需求的分析，牙齿模型识别系统主要具备以下功能：

（1）数据读取。可以解析整口牙的 stl 格式存储的 3D 扫描数据，获取整口牙数据中的图像信息。

（2）图像显示。原始数据是标准的 stl 格式，stl 格式的原理是用有限个三角面片来拟合复杂的表面，因此原始数据仅仅包含三角面片信息。因此本小组的系统需要把几何数据转换成为图形数据，而 vtk 封装的功能之一就是可以用数据流的方式把几何体数据转换成为图形数据并进行渲染显示。

（3）去掉牙龈。能够对提取的整口牙数据进行处理，去掉牙龈，只保留单独的牙齿。

（4）牙齿连通分量的提取。在去掉牙龈之后，单个牙齿之间不再连通，使用提取连通分量的方式来实现牙齿模型的分割。

（5）牙齿计数。牙齿的数量即为第四步中得到的去除牙龈之后连通分量的个数。

（6）绘制包围盒。包围盒原本用于碰撞检测，但是本小组 vtk 碰撞检测源码中抽取出绘制包围盒相关的代码，可以同时绘制单个连通分量的包围盒，同时输出包围盒角点的坐标值供后续处理。

（7）牙齿编号。FDI 牙位标记法，首先分辨所要标记的牙齿属于上颌还是下颌，确定牙齿编号的十位数，上颌的十位数是 1 和 2，下颌的十位数是 3 和 4。接下来根据绘制包围盒过程中得到的单个牙齿的包围盒顶点坐标值，构建牙齿类，每个牙齿类中包含中心点坐标和牙齿半径，通过计算牙齿半径和牙齿中心点之间的距离的大小关系来确定是否有牙齿缺失的情况，若有还需补充“空牙齿”，以保证编号正确，不会因为牙齿的缺失而编号出现紊乱。然后由门牙开始从中间向两边编号，最后再去掉“空牙齿”及其所对应的编号。

（8）牙齿缺失情况识别。由牙齿编号的数组可以得到正确的牙齿缺失情况信息，因为完整的牙齿编号 11~18,21~28,31~38,41~48，从上一步得到的编号和数组和完整牙齿编号进行比对，就可以判断牙齿的缺失情况并输出结果。

（9）绘制牙轴方向。利用包围盒的顶面和底面连线来确定牙轴方向，同时也用曲率的相关方法进行了辅助分析。

3.5 系统开发环境

本文所研究系统的环境为 Visual Studio 2017 + Vtk 8.0.2 + Qt 5.13.2，并用 C++语言进行编程。由于 Vtk 是使用 C++语言开发的平台无关的类库，为了搭建开发环境，需要将 Vtk 类库在 Visual Studio 2017 平台进行配置，其过程如下：

（1）安装 Visual Studio 2017，VS 是一个基本完整的开发工具集，选择套件时只需选择“使用 C++的桌面开发”；

（2）安装 Qt 5.13.2，Qt 是一个跨平台的 C++图形用户界面应用程序开发框架，易扩展，可以嵌入 VS 进行开发使用，使用 Vtk 开发时会用到 Qt 进行 GUI 开发；然后在 VS 中安装 Qt 扩展插件，下载并安装 QT Visual Studio Tools；

（3）安装 Cmake，Cmake 是一个跨平台的编译工具，可以用简单的语句来描述所有平台的编译过程；

（4）编译安装 Vtk，用 Cmake 选择路径，然后 Configure 和 Generate，就可以得到 Vtk.sln 文件，用 VS 2017 打开此工程文件，点击重新生成，开始长时间的编译过程，重新生成工作结束之后，环境配置完毕。

3.6 系统界面设计

一个简洁明了的系统界面能够给医护人员使用系统进行工作带来极大的便利，是系统人机交互性能好坏与否的重要体现。为了方便医护人员学习以及操作本文所研究的系统，系统根据所要实现的功能在 GUI 页面绘制了八个按钮，分别实现对应操作，并且鼠标可以交互操作渲染窗口中的牙齿模型对象的平移旋转、放大缩小等操作。为了在系统中展示容易操作而美观整洁的界面，使用了 Qt 界面设计。牙齿模型、去除牙龈、连通分量提取、牙齿数量、FDI 牙位标记法、牙齿缺失情况、包围盒、轴向。如图 2 所示。

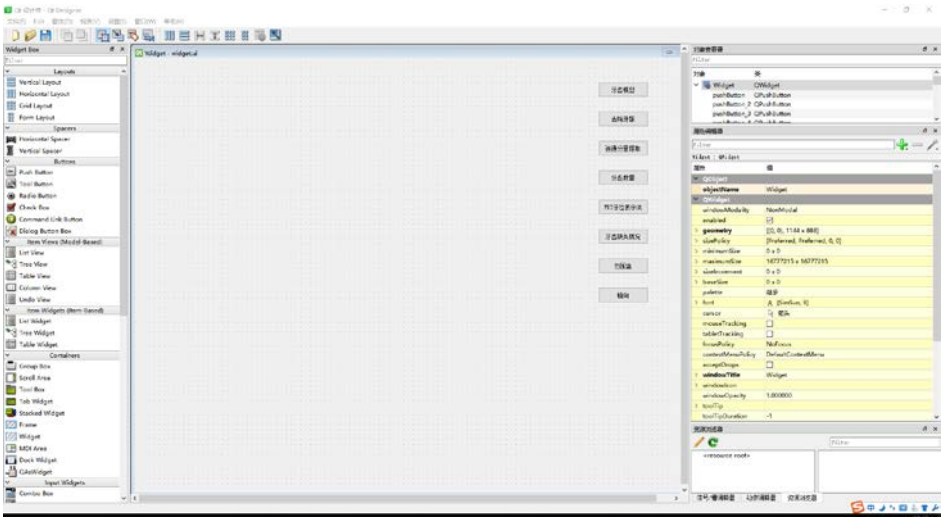


图 2　系统界面设计

Fig2 System interface design

3.7　Vtk重要的工具函数

Table 1　Important function in Vtk

表 1　Vtk 重要的工具函数

VtkPolyDataConnectivityFilter	SetExtractionModeToSpecifiedRegions()	连通分量提取
VtkPolyDataConnectivityFilter	GetNumberOfExtractedRegions()	得到连通分量数目
VtkOBBTree	GenerateRepresentation	生成 OBBTree 的体表示
VtkOBBTree	ComputeOBB	计算包围盒顶点坐标
VtkVectorText	SetText	绘制标记指定文本内容
VtkAxesActor	SetXAxisLabelText()	设定 x 坐标轴的标签
VtkAxesActor	SetYAxisLabelText()	设定 y 坐标轴的标签
VtkAxesActor	SetZAxisLabelText()	设定 z 坐标轴的标签
VtkLineSource	SetPoint1()	绘制轴向指定端点 1
VtkLineSource	SetPoint2()	绘制轴向指定端点 2

3.8　系统功能的实现

3.8.1　UI 界面设计

Vtk 显示过程中，使用 Qt 的界面设计，使用 QVTKWidget 将 Vtk 的渲染窗口显示到 Qt 的组件中。Main 函数对 Qt 界面类 Widget 进行实例化。Qt 界面类 Widget 中实现 vtk 和 qt 的结合，Widget.ui 使用 Qt 的图形化界面进行编程，自动生成 ui_widget.h 头文件，设置 8 个按钮：Widget.h 设置相应的 button slot 函数。Widget.cpp 中 Vtk 的编写，实现对牙齿模型处理的相关功能。

```
#include "widget.h"
#include <QApplication>
#include <qsurfaceformat.h>
#include <QVTKOpenGLWidget.h>
int main(int argc, char *argv[])  {
    QSurfaceFormat::setDefaultFormat(QVTKOpenGLWidget::defaultFormat());
```

```
7.      QApplication a(argc, argv);
8.      Widget w;
9.      w.show();
10.     return a.exec();
11. }
```

```
1.  class Widget : public QWidget  {
2.      Q_OBJECT
3.  public:
4.      explicit Widget(QWidget *parent = 0);
5.      ~Widget();
6.  private slots:
7.      void on_pushButton_clicked();
8.      void on_pushButton_2_clicked();
9.      void on_pushButton_3_clicked();
10.     void on_pushButton_4_clicked();
11.     void on_pushButton_5_clicked();
12.     void on_pushButton_6_clicked();
13.     void on_pushButton_7_clicked();
14.     void on_pushButton_8_clicked();
15. private:
16.     Ui::Widget *ui;
17. }
```

3.8.2 连通分量的提取

Algorithm 1: Extraction of connected components

```
for i ← 0 to n − 1 do
    connectivityFilter[i] = vtkSmartPointer< vtkPolyDataConnectivityFilter >::New();
    connectivityFilter[i]− >SetInputConnection(reader− >GetOutputPort());
    connectivityFilter[i]− >SetExtractionModeToSpecifiedRegions();
    connectivityFilter[i]− >AddSpecifiedRegion(i);
    connectivityFilter[i]− >update();
    extratedMapper[i] = vtkSmartPointer< vtkPolyDataMapper >::New();
    extratedMapper[i]− >SetInputConnection(connectivityFilter[i]− >GetOutputPort());
    extratedMapper[i]− >update();
    extratedActor[i] = vtkSmartPointer< vtkActor >::New();
    extratedActor[i] = SetMapper(extratedMapper[i]);
    Renderer[i] = vtkSmartPointer< vtkRenderer >::New();
    Renderer[i]− >setViewPort(a[i+1]);
    Renderer[i]− >AddActor(extratedActor[i]);
```

3.8.3 牙齿编号 FDI 牙位标记法+牙齿缺失情况

首先要确定各个牙齿的中心点坐标和在 XOZ 平面上投影的半径，以便用于后续的计算。通过连通分量的边界变量 Bound，得到各个牙齿的 AABB 包围盒的顶点坐标，再通过简单的计算，即可得到牙齿的中心点坐标和投影半径。接下来，在此基础上，进行牙齿编号和缺失识别的计算。

牙齿的编号建立在牙齿缺失识别的基础上进行，在得到正确的牙齿编号后，便可以通过遍历现有牙齿的

编号，与完整的牙齿编号进行比对，输出牙齿缺失情况。

牙齿缺失识别的方法是，从左往右扫描牙齿模型（即按照中心点坐标 x 递增的顺序遍历各个牙齿），根据每颗牙齿的中心点坐标，计算相邻两颗牙齿的中心点坐标之间的距离在 XOZ 平面上的投影，如果这个值大于这两颗牙齿的半径之和，则表明这两颗牙齿之间存在缺失的牙齿。为了便于后续编号，对于缺牙的地方，填充一个“空牙齿”（没有面片，但有中心点和半径等信息），“空牙齿”会被标记为“空”。

接下来进行牙齿的编号，牙齿编号采用的方法如下：

1.首先确定第一颗右门牙。确定方法：中心点 x 坐标为正值且最小的那颗牙齿即为右门牙。注：即便右门牙缺失，鉴于在缺失识别阶段已经填充了“空牙齿”，所以没有影响。

2.对右门牙进行编号。

3.以右门牙为基准，沿中心点 x 坐标值递增的方向，对右侧的牙齿进行挨个编号。

4.确定左门牙。确定方法：利用牙齿中心点坐标来确定，左侧离右门牙最近的牙齿即为左门牙。

5.进行左侧牙齿的命名，沿中心点 x 坐标值递减的方向，对左侧的牙齿进行挨个编号。

6.将各个非空牙齿的名称按照连通分量的顺序保存，输出。

由此就得到了各个牙齿连通分量对应的编号，完成了牙齿的命名工作。相关代码如下：

```
1.  char *id[49] = { "id:0","id:1","id:2","id:3","id:4","id:5","id:6","id:7","id:8
    ","id:9","id:10","id:11","id:12","id:13","id:14","id:15","id:16","id:17","id:18
    ","id:19","id:20","id:21","id:22","id:23","id:24","id:25","id:26","id:27","id:2
    8","id:29","id:30","id:31","id:32","id:33","id:34","id:35","id:36","id:37","id:
    38","id:39","id:40","id:41","id:42","id:43","id:44","id:45","id:46","id:47","id
    :48" };
1.  int* p = identify(regionNum, bounds, up_or_low);
1.  /*给定一系列 bounds，按照 bounds 原先的顺序，返回对应的牙齿编号*/
2.  textSource[i]->SetText(id[p[i]]);
```

其中，identify 函数计算并返回各个连通分量的编号。

identify 函数的算法如下：

Algorithm 2: Computing the id for each teeth by FDI method

Input: Num: The number of teeth in a dental model
Input: Bound[0,1,...,n - 1][0,1,...,5]: Bound[i][j] represents the jth boundary coordinate of the ith tooth
Input: UpOrLow: UpOrLow = 0: the upper jaw, UpOrLow = 1: the lower jaw
Output: $p[0, 1, \ldots, n-1]$: $p[i]$ represents the id of the ith tooth

/*Initialize the vector of teeth*/
$vector < Tooth > TeethList$;
for $i \leftarrow 0$ *to* $n-1$ **do**
  $tmp.abb.center[0] \leftarrow 0.5 * (Bound[i][0] + Bound[i][1])$
  $tmp.abb.center[1] \leftarrow 0.5 * (Bound[i][2] + Bound[i][3])$
  $tmp.abb.center[2] \leftarrow 0.5 * (Bound[i][4] + Bound[i][5])$
  $tmp.abb.R \leftarrow 0.5 * \sqrt{(Bound[i][1] - Bound[i][0])^2 + (Bound[i][3] - Bound[i][2])^2}$
  $tmp.isNULL \leftarrow false$
  $tmp.originalorder \leftarrow i$
  $TeethList.pushback(tmp)$
1st sort TeethList by x coordinate of the center of each tooth
/*Scan from left to right for empty tooth filling*/
for $i \leftarrow 0$ *to* $n-1$ **do**
  $SumOfR \leftarrow TeethList[i].aabb.R + TeethList[i+1].aabb.R$
  $xyDistance \leftarrow sqrt((TeethList[i].aabb.center[0] - TeethList[i+1].aabb.centerb[0])^2 + (TeethList[i].aabb.center[1] - TeethList[i+1].aabb.centerb[1])^2)$
  if $xyDistance > SumOfR$ **then**
    $empty.aabb.center[0] \leftarrow (TeethList[i].aabb.center[0] + TeethList[i+1].aabb.center[0]) * 0.5$
    $empty.aabb.center[1] \leftarrow (TeethList[i].aabb.center[1] + TeethList[i+1].aabb.center[1]) * 0.5$
    $empty.aabb.center[2] \leftarrow (TeethList[i].aabb.center[2] + TeethList[i+1].aabb.center[2]) * 0.5$
    $empty.aabb.R \leftarrow (xyDistance - SumOfR) * 0.5$
    $tmp.isNULL \leftarrow true$
    $TeethList.pushback(empty)$
2nd sort TeethList by x coordinate of the center of each tooth
/*FDI method*/
if $UpOrLow == 0$ **then**
  $quadrant \leftarrow 10$
else
  $quadrant \leftarrow 40$
for $i \leftarrow mid$ *to* $n-1$ **do**
  $TeethList[i].id \leftarrow quadrant + i - mid + 1$
if $UpOrLow == 1$ **then**
  $quadrant \leftarrow 20$
else
  $quadrant \leftarrow 30$
for $i \leftarrow mid - 1$ *to* 0 **do**
  $TeethList[i].id \leftarrow quadrant + mid - i$
for $i \leftarrow 0$ *to* $n-1$ **do**
  if $TeethList[i].isNULL == NULL$ **then**
    continue
  $p[TeethList[i].originalorder] \leftarrow TeethList[i].name$
return $p[0, 1, \ldots, n-1]$;

根据各连通分量的编号，统计牙齿缺失情况的算法如下：

Algorithm 3: A record of tooth miss

Input: UpOrLow,Num,$p[0, 1, 2, \ldots, n-1]$:$p[i]$ represents the id of the ith tooth
Output: final: string made up of all elements of miss

```
if UpOrLow == 0 then
    for i ← 0 to 7 do
        if FindArrKey(p,Num,11+i) then
            miss.pushback(11+i)
        if FindArrKey(p,Num,21+i) then
            miss.pushback(21+i)
else
    for i ← 0 to 7 do
        if FindArrKey(p,Num,31+i) then
            miss.pushback(31+i)
        if FindArrKey(p,Num,41+i) then
            miss.pushback(41+i)
if !miss.empty() then
    strcpy(final,id[miss[i]]);
    for i ← 0 to miss.size()-1 do
        strcat(final,id[miss[i]]);
return final;
```

3.8.4 ObbTree 包围盒

利用 Vtk 库的工具函数，根据每颗牙齿的面片和点的特征建立各自的 OBB 包围盒。

Algorithm 4: OBBTree

Input: Num

```
for i ← 0 to Num do
    polyData[i] = connectivityFilter[i]− >GetOutput();
    obbTree[i] = vtkSmartPointer< vtkOBBTree >::New();
    obbTree[i]− >SetDataSet(polyData[i]);
    obbTree[i]− >SetMaxLevel(maxLevel);
    obbTree[i]− >BuildLocator();
    obbTree[i]− >ComputeOBB(polyData[i], corner[i], max[i], mid[i], min[i], size[i]);
    polydata[i] = vtkSmartPointer< vtkPolyData >::New();
    obbTree[i]− >GenerateRepresentation(0, polydata[i]);
    obbtreeMapper[i] = vtkSmartPointer< vtkPolyDataMapper >::New();
    obbtreeMapper[i]-¿SetInputData(polydata[i]);
    obbtreeActor[i] = vtkSmartPointer< vtkActor >::New();
    obbtreeActor[i]− >SetMapper(obbtreeMapper[i]);
    obbtreeActor[i]− >GetProperty()− >SetInterpolationToFlat();
    obbtreeActor[i]− >GetProperty()− >SetOpacity(.5);
    obbtreeActor[i]− >GetProperty()− >EdgeVisibilityOn();
    obbtreeActor[i]− >GetProperty()− >SetColor(colors− >GetColor4d("SpringGreen").GetD
```

3.8.5 牙齿轴向

牙轴方向的识别：采用 OBB 包围盒的方式近似的识别牙齿轴向，将从 OBB 包围盒底面中心指向顶面中心的向量作为牙轴的方向。

Algorithm 5: Draw the tooth axial line

Input: Num

```
for i ← 0 to Num − 1 do
    lineSource[i] = vtkSmartPointer< vtkLineSource >::New();
    lineSource[i]− >SetPoint1(corner[i][0] + (max[i][0] + mid[i][0]) / 2 - min[i][0],
     corner[i][1] + (max[i][1] + mid[i][1]) / 2 - min[i][1], corner[i][2] + (max[i][2] +
     mid[i][2]) / 2 - min[i][2]);
    lineSource[i]− >SetPoint2(corner[i][0] + (max[i][0] + mid[i][0]) / 2 + 2 * min[i][0],
     corner[i][1] + (max[i][1] + mid[i][1]) / 2 + 2 * min[i][1], corner[i][2] + (max[i][2] +
     mid[i][2]) / 2 + 2 * min[i][2]);
    lineSource[i]− >Update();
    mapper[i] = vtkSmartPointer< vtkPolyDataMapper >::New();
    mapper[i]− >SetInputConnection(lineSource[i]− >GetOutputPort());
    mapper[i]− >Update();
    actor[i] = vtkSmartPointer< vtkActor >::New();
    actor[i]− >SetMapper(mapper[i]);
    actor[i]− >GetProperty()− >SetColor(0, 1, 1);
    actor[i]− >GetProperty()− >SetLineWidth(10);
```

4 实验结果与分析/Experiment Results and Analysis

运行本小组的程序，运行时间大约 5 秒，读取任意一个 STL 牙齿模型，会出现一个用户交互界面。通过鼠标的点击和拖动，都可以进行旋转和缩放，实现全方位的观察。

4.1 读取数据与图形显示

点击**牙齿模型**，会显示读入的牙齿模型，如图 3；

4.2 去除牙龈

点击**去除牙龈**，会显示去除牙龈后的牙齿模型，如图 4；

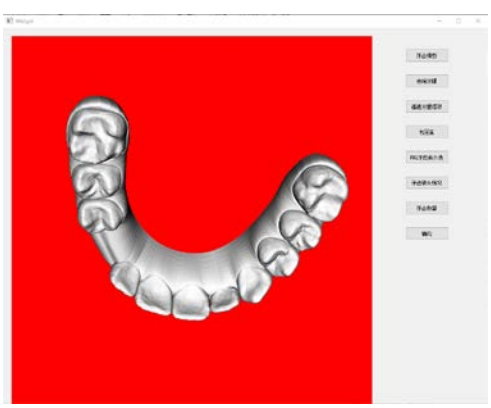


图 3 读取整口牙数据

Fig.1 Read and display the data of the whole tooth

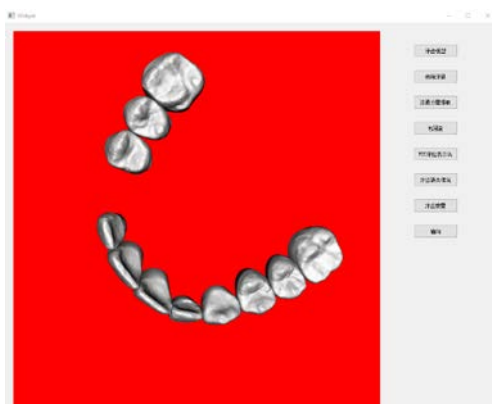


图 4 去除牙龈

Fig.2 Remove gums

4.3 提取连通分量

点击**连通分量提取**，会显示将牙齿分成独立的各个牙齿的画面，其中包含去除牙龈后的整口牙的画面，和每个窗口显示单独的牙齿的画面，如图 5 所示；

4.4 绘制包围盒

点击**包围盒**，会显示出各个牙齿的 OBB 包围盒，同时会输出每个包围盒顶点的坐标，而这些坐标正是后续牙齿编号的基础，如图 6 所示；

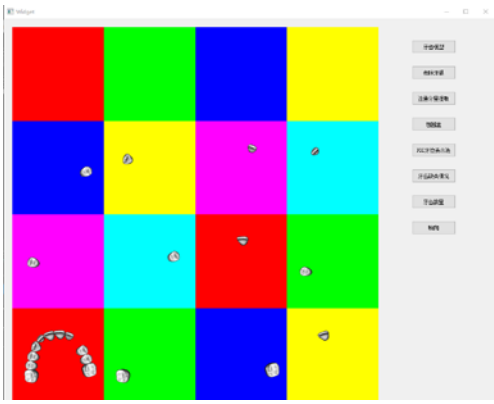


图 5 提取连通分量

Fig.5 Extraction of connected components

图 6 绘制包围盒

Fig.6 Draw the OBB bounding box

4.5 FDI牙位标记法进行编号

点击 **FDI 牙位表示法**，会显示各个牙齿使用 FDI 牙位表示法编号后的画面，红色的字体即为各个牙齿的编号。

本小组对助教老师提供的牙齿数据进行了测试，标记结果都是正确的。如图 7 和 8 所示，选择了有代表性的整口牙的数据。上颌缺失的牙是 23,17,18,27,28，下颌缺失的牙是 38,48，程序运行结果都是正确的，其中 18,28，38,48 是智齿，可以看出智齿的缺失情况也可以正确识别。

4.6 牙齿缺失情况识别

点击**牙齿缺失情况**，若有缺失情况，会上半部分显示“miss”，下半部分显示缺失的牙齿编号（包括智齿）；若无缺失情况，上半部分则显示“complete”。

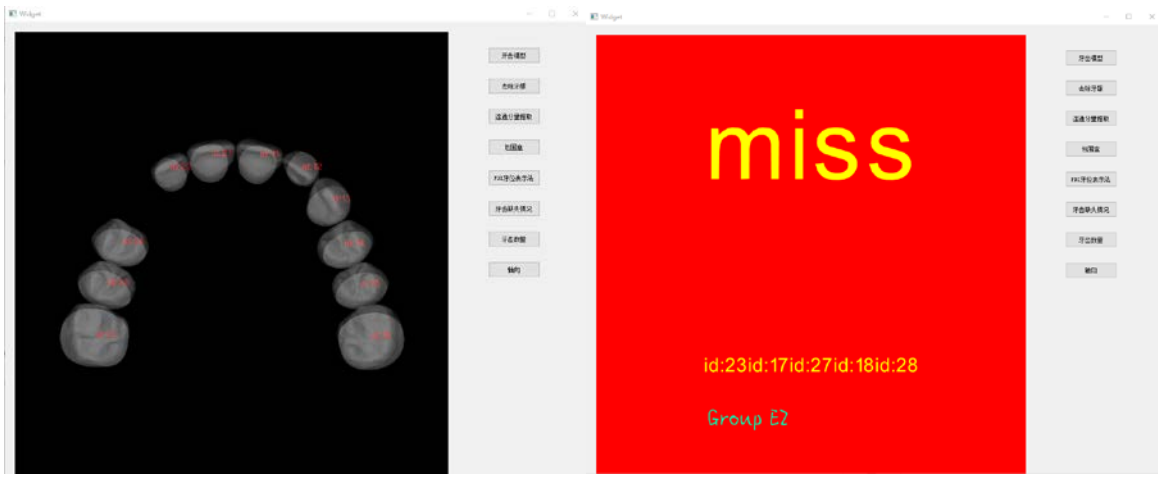


图 7　上颌 FDI 牙位标记法+缺失情况显示，缺失 23,17,18,27,28

Fig.7 FDI dental position labeling method + Absense display，miss 23,17,18,27,28

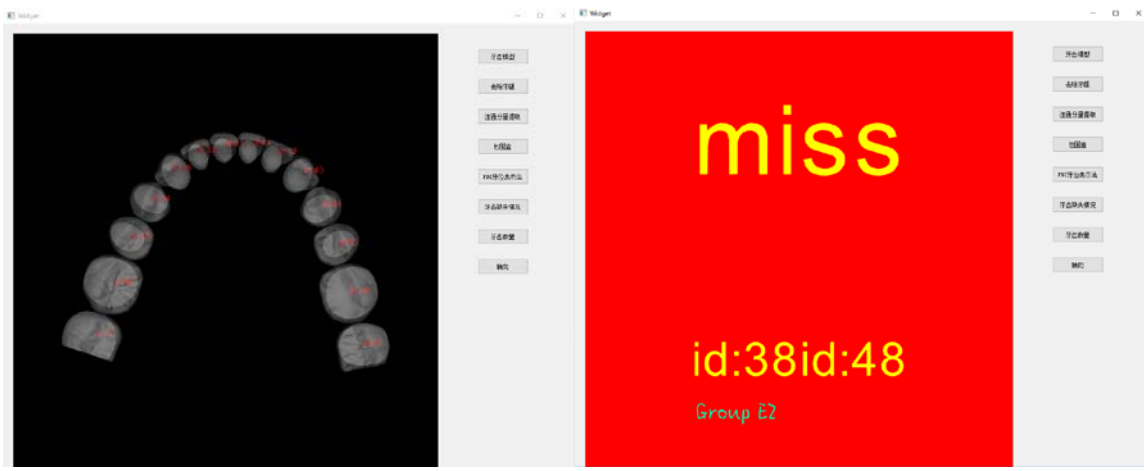


图 8　下颌 FDI 牙位标记法+缺失情况显示，缺失 38,48

Fig.8 FDI dental position labeling method + Absense display，miss 38,48

4.7　牙齿计数

点击**牙齿数量**，会显示牙齿的总个数，如图 9 所示；

图 9：牙齿个数显示

Fig.9 Number of teeth

4.8 牙齿轴向绘制

点击**轴向**，会显示添加了牙齿轴向的画面，如图 10 和 11 所示；

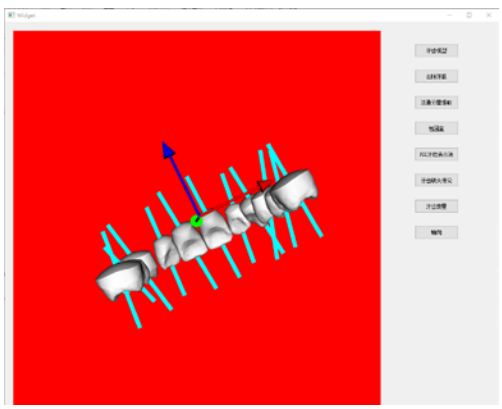


图 10 上颌牙齿轴向绘制

Fig.10 Axial drawing of maxillary teeth

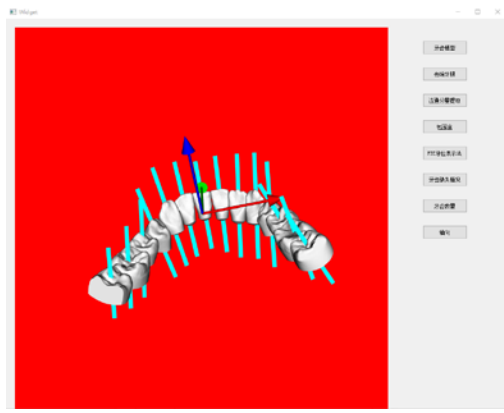


图 11 下颌牙齿轴向绘制

Fig.11 Axial drawing of mandibular teeth

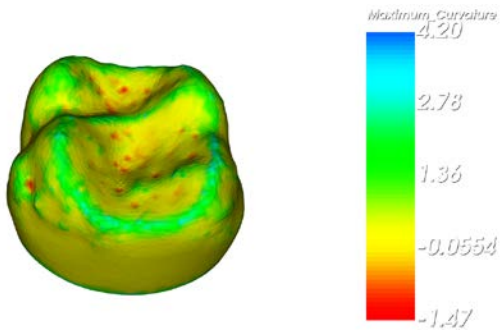


图 12 牙齿曲率的分析

Fig.12 Analysis of tooth curvature

5 特色与创新/ Distinctive or Innovation Points

5.1 特色与创新

1. 本小组实现的程序能对任意的 STL 牙齿模型，进行牙龈的去除、牙齿的分割、编号、缺失牙齿的识别、以及牙齿轴向的显示，程序的鲁棒性较高，即便是缺失了门牙或者最边缘的磨牙，本小组的程序依然能对牙齿进行正确的编号和缺失情况的正确识别；

2. 本小组的程序对牙齿进行编号的根据是通过牙齿坐标，而不是牙齿的具体形态，算法更简约高效，运行快速，也能保证很高的准确性；

3. 此外，本小组专门利用 Qt 设计了可交互的用户界面，使得程序功能的可实用性大大提高了，对用户更加友好。

5.2 总结与展望

该系统可以很好的帮助医生对牙齿模型进行数量统计、编号、牙轴方向显示和缺失牙齿的识别，但还存在需要改进的地方。

首先是程序对连续缺失两颗以上牙齿的情况的处理不周，当有两颗以上牙齿连续的缺失时，算法只会补上一颗很大的“空牙齿”，而这会导致缺失部位周围牙齿的编号错误，因为两侧的牙齿的编号依赖于相对中间的牙齿的编号，因此会导致连续的错误。而事实上，解决这个问题并不难，只需要在缺失识别算法中再加上一种情况判断：当两颗牙的中心的间距大于半径之和的二倍时，添加连续两颗“空牙齿”。连续缺失三颗乃至以上的情况同理。但是这样处理的问题是，系统的计算是直接计算两颗牙齿中心点之间的直线距离，当连续缺失的牙齿个数较多时，由于牙齿排列是有弧度的，依赖直线距离进行牙齿缺失的判断的误差会变大，导致编号和缺失识别不准确，因此对连续多颗牙齿缺失情况的识别，该系统还是需要进一步改善，改善的思路可以考虑根据牙模弧度来矫正牙齿中心点之间的距离的计算。

此外，在牙齿特征点的识别方面，比如牙尖的识别，虽然也进行了曲率特征的可视化分析（如图 12 所示）但主要采取的方法还是利用 OBB 包围盒的轴向来近似牙齿的牙尖方向，这个估计的误差还是挺大的。本项目将来的工作打算深入研究利用曲率来识别牙尖的方法，通过曲率计算，精确的标记出各个牙齿的牙尖方向。

参考文献:

[1] 褚玉伟. 数字化口腔中三维模型处理技术的研究与系统开发[D]. 2019.

[2] 汤德衍. 基于数字几何处理的数字化口腔技术研究与系统开发[D]. 2018.

[3] 宁楠. 数字口腔可视化技术及系统开发[D]. 2017.

[4] 周杭民. 三维模型处理算法在骨科与牙科中的应用[D]. 2015.

[5] 张翔, 廖文和, 俞青, et al. 基于 OpenGL 的复杂多面体模型间距离计算及碰撞检测[J]. 东南大学学报(自然科学版)(2):54-58.1

[6] 褚玉伟, 罗晓博, 屈珂, et al. DBSCAN 和 K-Means 混合聚类的牙齿特征自动识别[J]. 计算机辅助设计与图形学学报, v.30(7):102-109.

[7] 宁小娟. 牙齿数字模型的分割方法研究[D].西安科技大学,2007.

[8] 郝国栋,程筱胜,戴宁,俞青. 基于形态学的牙齿模型交互分割[J]. 中国制造业信息化：学术版(1 期):36-39

[9] 王启超, 宁楠, 褚玉伟, et al. 一种基于局部坐标的牙齿模型特征点自动识别算法:.

[10] 陈小岗, 孙全平, ChenXiaogang,等. 基于生长原理的磨牙修复体(牙合)面特征区域识别方法[J]. 计算机应用与软件, 2010, 27(3):87-89.

[11] 公茂亮. 基于 CT 体数据的牙齿分割研究[D]. 东南大学, 2012.

[12] 刘伟. 基于结构点与轮廓特征的牙齿影像处理和配准技术研究[D]. 山东大学, 2014

[13] 马亚奇, 李忠科, 王先泽. 基于测地路径的牙齿模型交互分割算法研究[J]. 中国图象图形学报, 2011, 16(4):554-558.

[14] 郝国栋, 程筱胜, 戴宁, et al. 基于形态学的牙齿模型交互分割[J]. 机械设计与制造工程, 2008, 37(1):36-39.

[15] 马勇, 柯永振, 杨帅. 基于网格抽取的牙齿模型快速分割算法[J]. 计算机应用与软件, v.35(5):253-258.

[16]汪葛. 基于水平集的 CBCT 牙齿图像分割算法研究[D]. 上海理工大学.

[17]Liepa P. Filling holes in meshes[C]// Eurographics/acm SIGGRAPH Symposium on Geometry Processing. Eurographics Association,2003:200-205

[18]Pernot J P,Moraru G.Filling holes in meshes using a mechanical model to simulate the curvature variation minimization[M].Pergamon Press,Inc,2006:892-902

[19]袁天然，廖文和，程筱胜，等. 三维牙颌模型的牙齿形状建模方法[J]. 计算机辅助设计与图形学学报，2010，22(4):000703-710

[20]范然，金小刚. 模版特征线匹配的牙齿形状修复[J]. 计算机辅助设计与图形学学报,2014,26(2):280-286

[21]Xia G,Chen L.3D dental mesh repairing using template-based deformation[C]//International Conference on Biomedical Engineering and Informatics. IEEE,2015:410-414

[22]汪利飞. 基于线结构光的三维牙模扫描技术研究[D].西北农林科技大学,2019

时间安排与分工统计表

<table>
<tr><td colspan="4">组员信息（含组长）杨国峰（组长），张宏（组员）</td></tr>
<tr><td>学生姓名</td><td>杨国峰</td><td>学　　号</td><td>517021910736</td></tr>
<tr><td>项目分工</td><td colspan="3">资料搜集，计划安排，代码编写和调试，报告撰写</td></tr>
<tr><td>学生姓名</td><td>张宏</td><td>学　　号</td><td>517051910016</td></tr>
<tr><td>项目分工</td><td colspan="3">资料搜集，报告撰写，代码编写，答辩</td></tr>
<tr><td>学生姓名</td><td></td><td>学　　号</td><td></td></tr>
<tr><td>项目分工</td><td colspan="3"></td></tr>
<tr><td>时间安排/
Schedule</td><td colspan="3">（如选题、方案制定、试验研究、数据处理、研制开发、撰写总结报告等）(Such as topic selection, program formulation, experimental research, data processing, research and development, writing summary reports, etc.)
第七周选题；
第八周搜集和阅读相关论文；
第十周确定实现方案和代码编写计划；
第十一周完成代码的编写；
第十二周完成代码的调试、优化和报告的初步撰写；
第十三周准备答辩和程序的进一步优化；
第十四~十六周完成报告的撰写和润色。</td></tr>
</table>