**自动降重**

哈 尔 滨 理 工 大 学

毕 业 设 计

题 目： 基于手绘草图的三维模型检索

院、 系： 计算机科学与技术

姓 名： 邱泳锋

指导教师： 高雪瑶

系 主 任： 孙东璞

2021 年 6 月 13 日

基于手绘草图的三维模型检索

摘 要

随着计算机CPU和GPU的计算能力大大提高，三维模型不仅变得越来越复杂，而且细节信息也越来越丰富，在动画、机械、医疗等领域应用越来越广泛，三维模型的数量也越来越多，而三维模型的分类和检索已成为一个重要的研究方向。尽管许多学者对三维模型检索技术进行了各种研究，并提出了许多不同的模型检索算法，但仍有许多问题有待解决。

本文通过对现有三维模型检索技术的研究分析，发现三维模型固有的复杂性和高维数的计算量严重影响了模型的检索。本文对三维模型进行降维处理，将二维视图作为三维模型的检索条件，以降低检索成本。提出了一种基于手绘草图的三维模型检索方法，本文主要研究了以下内容。

为了提高检索的准确性，首先通过适当的空间位置变化来渲染三维模型，然后根据固定投影的方法得到二维视图图集。每个模型选择6个2D视图作为3D模型的特征视图集。然后从手绘草图和二维视图图集中提取特征向量，并以Zernike矩和Fourier描述符作为集成特征描述符，构造加权的全局视图特征和D2描述符集。通过集成特征描述符来检索三维模型，以充当用户手绘草图和模型的二维视图的特征向量，从而执行相似性评估。实验结果表明，该方法能够有效地对三维模型进行分类。

三维模型检索; 手绘草图检索; Zernike矩; 傅里叶描述符; D2形状描述符; 集成特征描述符

3D Model Retrieval Based on Hand-drawn Sketch

Abstract

With the rapid development of CPU and GPU, 3D models are not only becoming more and more complex and rich in details, but also widely used in animation, machinery, medical and other fields. The number of 3D models is also increasing. The classification and retrieval of 3D models has become an important research direction. Although many scholars have done different researches on 3D model retrieval technology and proposed many different model retrieval algorithms, there are still many problems to be solved.

Through the research and analysis of the existing 3D model retrieval technology, this paper finds that the inherent complexity of 3D model and high-dimensional calculation seriously affect model retrieval. In this paper, the three-dimensional model dimension reduction method is used, and the two-dimensional view is used as the retrieval condition of the three-dimensional model to reduce the retrieval cost. This paper presents a 3D model retrieval method based on hand-drawn sketches.

In order to improve the accuracy of retrieval, firstly, the 3D model is rendered according to the appropriate spatial position change, and then the 2D view set is obtained according to the fixed projection method. Each model selects 6 2D views as the feature view set of 3D model. Secondly, feature vectors are extracted from sketch and 2D view sets, and weighted sets of global view features and D2 descriptors are constructed by Zernike moments and Fourier descriptors as integrated feature descriptors. By integrating feature descriptors to serve as feature vectors of user's hand drawn sketches and 2D views of the model, similarity evaluation is performed to retrieve 3D models. Experimental results show that this method can effectively classify 3D models.

Keywords 3D model retrieval, freehand sketch retrieval, Zernike, Fourier descriptor, D2 shape descriptor, integrated feature descriptor

符

目 录

摘要 I

Abstract II

第1章 绪论 1

1.1 课题背景 1

1.2 研究意义 1

1.3 国内外研究现状 2

1.4 课题的主要研究内容 3

1.5 本文文章结构安排 3

第2章 基于手绘草图的三维模型检索的总体框架 5

2.1 三维模型检索的体系架构 5

2.2 三维模型检索的处理流程 6

2.2.1 三维模型投影的处理架构 6

2.2.2 三维模型检索的处理架构 6

2.3 本章小结 7

第3章 三维模型的处理 8

3.1 OFF模型文件格式解析及渲染 8

3.2 固定视角的模型投影技术 9

3.3 三维模型的光照和材质添加 12

3.4 三维模型的空间变化 14

3.5 三维模型投影系统结果 15

3.6 本章小结 16

第4章 草图的绘制处理 17

4.1 直线类图形绘制 17

4.1.1 直线 18

4.1.2 矩形 18

4.1.3 三角形 18

4.2 曲线图形绘制 19

4.3 圆形绘制 20

4.4 铅笔线绘制 21

4.5 草图绘制的结果 21

4.6 本章小结 22

第5章 草图及三维模型的特征提取 23

5.1 全局试图特征描述符 23

5.1.1 Zernike矩 23

5.1.2 Fourier描述符 25

5.2 D2描述符 26

5.3 集成描述符 27

5.4 本章小结 28

第6章 基于手绘草图的三维模型检索 29

6.1 相似性计算 29

6.1.1 欧几里得距离 30

6.1.2 曼哈顿距离 30

6.1.3 切比雪夫距离 30

6.1.4 闵可夫斯基距离 31

6.1.5 马氏距离 31

6.2 基于手绘草图的三维模型检索结果 32

6.3 本章小结 33

结论 34

致谢 35

参考文献 36

附录A 37

附录B 48

附录C 57

。

绪论

课题背景

随着3D建模相关技术的进步，尤其是3D扫描设备的成熟和普及，互联网空间积累了大量可共享的3D模型，模型数量持续急剧增加。例如，用户可以直接从Google的3D仓库下载海量3D模型。著名的3D模型交易平台TurboSquid目前有300,000多个模型。为了充分利用现有的3D模型，有必要开发方便，高效和可靠的3D形状 (模型) 检索引擎。三维形状检索是通过特定的交互操作，从数据库中找到符合用户意图的三维形状。其中，检索到的对象可以是整个模型 (例如椅子) 或模型的一部分 (例如椅子的手柄)。

三维模型不仅在数量上迅速增长，而且应用越来越广泛。三维模型广泛应用于工业产品设计、建筑设计、虚拟现实、计算机仿真、多媒体教学、分子生物学、教育、三维游戏和影视动画等领域，三维数据模型库也越来越普遍。在互联网上，越来越多的三维模型库应运而生。三维模型已经成为MPEG-7标准模型的重要研究内容。因此，三维模型的检索是当今图形检索的一个热点。

研究意义

相关三维建模企业应该优先把握在当前激烈的市场竞争中的商机，生产出符合用户需求的产品，并从中获取利润，就不得不思考降低三维模型设计和三维模型处理的高昂成本。研究和调查表明，在产品设计过程中，只需从头开始制作约20% 个设计。40% 设计可以通过直接重用现有设计来完成; 此外，可以通过对现有设计进行适当的修改来获得40% 设计。如果您在每次生产新产品时从头开始选择材料，绘制图纸并设计加工计划，这不仅会花费时间，而且会增加成本。在生产过程中，经常发生重复设计，其本质原因是没有高效的CAD模型检索工具。快速有效的CAD模型检索无疑将大大缩短开发周期，不仅提高了设计效率，而且为企业带来更多的效益。重用现有的CAD模型也将使企业节省更多的成本。

随着三维CAD模型的数量和类型的急剧增加，CAD模型库的规模也随之增加。据统计，世界上有300亿个三维模型。三维模型的结构复杂多样。如何从模型库中准确找到符合用户设计意图的模型变得越来越重要。因此，三维CAD模型检索技术正受到国内外专家学者的广泛关注。

三维CAD模型检索是计算图形学和计算机辅助设计 (Computer Aided Design，CAD) 领域的重要研究课题，也是近年来这些领域的研究热点。这项研究不仅对三维CAD模型的复用具有很高的实用价值，而且大大加深了人们对计算机辅助设计和人工智能的认识，促进了相关学科的发展。

国内外研究现状

近年来，国内对图形检索的研究也得到了进一步的发展。目前，在国内对三维图形检索的研究中，草图识别方法主要包括以下几类:

(1) 基于收集统计数据和数据的识别方法，对草图中的各种线型进行数理统计，统计得到的信息用于识别。

(2) 基于模糊类的识别方法，运用模糊处理的相关原理和技术，对草图位置、笔画速度等绘图特征参数进行识别。

(3) 基于交互式几何帧识别方法，分析降噪草图的笔画特征，然后通过笔画中反映的几何特征找到相似的因变量，将笔画形成的角度信息与预设阈值进行比较分类，实现草图识别。

(4) 基于滤波器和神经网络的识别方法，即构造滤波器对笔画进行分类。

基于手绘草图的3D模型检索，尽管不同的用户会根据其主观意图对同一模型进行不同的描述，但模型的组成是固定的。例如，显示器由两个主要部分组成，一个是底座，另一个是屏幕。绝大多数屏幕是矩形的，并且屏幕必须在底座上方，并且显示屏比显示器的底座大得多。由此可以得出结论，我们可以根据用户绘制的不同的简单草图，简单地确定用户想要检索的预期目标。Sezgin T M等 [1] 提出了一种可以识别线、圆等几何图形的算法，Li B等 [2] 开发了一种基于监督学习的正能量草图识别器，草图表达的语义可以正确得到。觉飞 [3] 建立了基本的基于草图的三维场景检索基准，并在该基准上评估了14种基于草图的检索方法。Zhu[4] 将三维图形投影到二维空间中，并使用自动编码器对二维图像进行特征学习。传统的局部图像描述符辅以深度学习功能。Konstantinos等 [5] 和周岩等 [6] 利用三维模型的二维全景表示作为卷积神经网络的输入，利用卷积神经网络计算特征。张云峰 [7] 利用图像与三维模型表达信息的互补性，建立图像与模型的关系。在Canny边缘信息的基础上，张一坤等进一步提取形状上下文特征描述的全局信息，并融合ORB特征和形状上下文特征，得到新的特征表示三维模型。李海生等提出了基于模型中二面角分布直方图的特征描述方法。安伯青 [10] 利用深度学习技术，解决了基于手绘草图的三维模型检索问题。

课题的主要研究内容

三维模型检索策略研究在分析二维视图特征定义和三维模型相似度计算的基础上，探索了一套基于sketch的三维模型检索方法。通过从不同角度投影三维模型得到相应的二维视图。计算草图和三维模型的每个二维视图之间的相似性。选择最大相似度值作为草图和三维模型之间的相似度。同时，开发了基于草图的三维模型检索系统。

通过阅读大量文献，深入了解了国内外三维模型检索方法的起源、发展及趋势，并分析总结了目前存在的问题和不足，对三维模型的检索方法进行了认真分析和深入研究。本文的主要研究内容如下:

1) 从不同角度投影3D模型，以获得相应的2D视图，并收集所有2D视图以形成最佳视图集。

2) 综合利用多个特征来描述草图的二维视图和三维模型，并使用欧几里得距离来计算草图与二维视图之间的相似性。基于三维模型的二维图集计算草图与三维模型之间的相似度。

3) 开发三维模型检索系统，从模型库中找到与草图最相似的三维模型，并显示多个相似的三维模型排序列表。

本文文章结构安排

三维模型检索策略研究在分析二维视图特征定义和三维模型相似度计算的基础上，探索了一套基于sketch的三维模型检索方法。通过从不同视角投影三维模型得到相应的二维视图。计算草图的每个二维视图与三维模型之间的相似性。选择最大相似度值作为草图和3D模型之间的相似度。同时，开发了基于草图的三维模型检索系统。论文共6章，各章的内部结构安排如下:

第一章，引言。本章首先简要介绍了三维模型检索的研究背景和意义，论述了基于二维视图的三维模型在国内外的研究现状以及存在的问题，然后介绍了本文的来源和主要研究内容，最后陈述了本文的章节安排。

第二章是基于手绘草图的三维模型检索的总体框架。该系统主要包括两部分，一是模型投影的处理，二是检索系统的处理。对于这两部分的过程进行梳理和介绍。

第三章是三维模型的处理。本章主要介绍了本文实验所需的数据采集，分析了所采用的模型格式，具体介绍了各种投影方式在坐标空间上的变化以及灯光效果的添加。对模型进行渲染以获得合适的三维模型，并进行投影以获得二维视图集。

第四章，素描绘图处理。本章主要介绍手绘草图绘图板的设计。在硬渲染的基础上，采用软渲染的算法实现了基本图形的绘制，包括直线、矩形、三角形、圆形、曲线和铅笔线的绘制。

第五章，三维模型的草图和特征提取。本章提出了一种集成描述符，将包括Zernike矩和Fourier描述符的全局视图描述符与二维形状分布相结合，提取手绘草图和二维视图的特征，以解决单个描述符对二维视图特征提取不完全的问题。

第六章是基于手绘草图的三维模型检索。本章主要比较几种距离计算方法，选择最佳距离来计算草图与二维视图的相似度。

最后对全文的工作进行总结并对未来发展进行了展望。

基于手绘草图的三维模型检索的总体框架

三维模型检索的体系架构

本文实现的检索系统是基于手绘草图的三维模型检索。基于手绘草图的三维模型检索主要包括手绘草图、手绘草图的特征提取、三维模型的渲染、三维模型的固定投影、影视图集的特征提取、相似度比较。图2-1显示了系统的总体框架。

1.手绘草图的绘制。用户可以在纸上画出简单的草图并上传到检索系统，也可以在系统中给出的简单画板中画出草图，画板是工具的集合 (直线、矩形、三角形、圆形、曲线、铅笔线) 需要绘制草图的一部分。

2.手绘草图的特征提取。本文提出了一种结合全局视图特征和D2描述符的集成描述符作为新的描述符，全局视图特征是Zernike矩和Fourier描述符的组合，使用该描述符提取手绘草图的特征以检索3D模型。

3.3D模型的渲染本系统的三维模型采用net40。模型库包含40种不同类型的3D模型。并且模型的存储文件格式为OFF file format。因此，有一个简单的渲染器可以渲染所有模型，并且渲染器集成了一些基本操作 (模型旋转，缩放，移动，添加和修改照明材料)。

4.三维模型的固定投影如果直接使用二维草图进行三维模型检索，则三维模型的大小会造成特征的较大差异，本文中的方法是对现有的三维模型进行一定程度的处理，如模型降维操作，也就是说，通过投影将三维模型变成二维视图的表示。

5.影视图集的特征提取提取模型的特征的方法与提取草图特征的方法相同。

6.相似性比较。距离公式用于计算最接近草图的8个检索模型。

图2-1 系统的总体框架

三维模型检索的处理流程

三维模型投影的处理架构

基于手绘草图的三维模块投影过程如图2-2所示。首先，分析OFF格式模型，将模型导入系统，为模型添加材质和各种照明效果，然后对模型进行平移，旋转和缩放。在渲染的三维模型中，以一定角度进行投影。

图2-2 三维模型投影的处理架构

三维模型检索的处理架构

基于手绘草图的三维模型检索过程分为在线检索和离线检索两个阶段，如图2-3所示。

图2-3 三维模型检索的处理架构

在离线检索过程中，首先采用固定投影的方法对三维模型进行降维，得到6个三维模型库的二维视图图集，最后，通过集成描述符提取最优视图图集的全局视图特征和二维形状特征，并对55维的集成描述特征进行加权并存储在数据库中。在在线用户检索阶段，用户首先制作手动手绘草图，然后使用集成描述符提取集成特征。将相似性与存储在数据库中的三维模型的集成特征进行比较。最后，将相似度最高的前八个模型从大到小排序，并在检索界面中显示相似度最高的前八个模型。

本章小结

本章主要介绍了三维模型检索算法的两部分，一部分是三维模型投影的完成过程，包括模型渲染、光照、材料添加和模型修改，包括旋转、平移和缩放。另一部分是完整的检索过程，包括基于草图的三维模型检索过程来分析各部分模块的功能和功能，并给出流程图。

三维模型的处理

OFF模型文件格式解析及渲染

本系统的三维模型采用modelnet40。模型库包含40个不同的3D模型，模型的存储文件格式为OFF file format。要读取和绘制模型，我们首先需要知道模型是如何存储在文件中的。通常模型由网格组成，一般为三角形网格。这是因为其他多边形网格可以很容易地划分成三角形，而三点共面可以确保平面性并可以很容易地定义内部和外部方向，进行插值等操作。因此，您需要先解析OFF文件格式。OFF文件用于表示给定曲面多边形的模型的几何形状。这里的多边形可以有任意数量的顶点。Net40中的OFF文件遵循以下标准。OFF files是所有以OFF关键字开头的ASCII文件。下一行描述顶点、补丁和边的数量。可以安全地省略边的数量。顶点列出每行一个x、y和z坐标。在顶点列表之后，按行列出补丁。对于每个补丁，指定顶点的数量，然后是顶点的索引列表。如图3-1所示，这是浴缸的OFF格式模型。

图3-1 OFF模型

现在，主流存储模型的数据结构有面列表、邻接矩阵、半边结构。人脸列表是一个三重 (Vertex1，Vertex2，Vertex3)，用于存储人脸中的顶点。它的优点是方便紧凑，可以表达非流行的网格。缺点是不能有效地支持点，面之间的邻接查询。邻接矩阵是表示顶点之间相邻关系的矩阵。它的优点是支持顶点之间的邻接信息v-v (顶点到顶点) 的有效查询，并支持非常用网格。它的缺点是没有边显示表达式，V-F (顶点到面)，V-E (顶点到边)，E-V (边到顶点)，F-E (面到边)，E-F (边到面) 的快速查询。半边结构记录了所有的面，边和顶点，包括几何信息，拓扑信息和辅助属性，在大多数集合建模应用中很流行。它的优点是所有查询操作的时间复杂度为O(1)，所有编辑操作的时间复杂度为)(1)。它的缺点是只能表示流行的网格。在这个系统中，我使用人脸列表的数据结构。读取模型的流程图如图3-2所示。

图3-2 模型流程图

固定视角的模型投影技术

该系统基于手绘草图检索。目前主流的研究方法之一是获得一组较为理想的最优视图作为检索数据库，以增加检索的成功率; 另一种是探索一种较为合适的特征描述符作为模型的特征向量，从而增加检索的准确率。本文将输入模型的最优视图定义为6个不同的二维图集。采用固定视角的投影技术。

因为用户绘制的草图是二维的，而模型是三维的，所以需要对三维模型进行降维。避免在后续的相似性计算中由于维度灾难而产生的额外时间开销。此外，当不同的用户绘制二维草图时，他们实际上已经对三维空间中的对象进行了降维过程。所画的是物体以固定视角的投影。因此，为了适应不同的用户，可以在不同的视角下绘制相同的物体。考虑到检索系统的效率，该系统仅获得同一模型的六个不同视角的投影，并形成该模型的最优视图图集。在尽可能包括用户绘制的对象的透视图的同时，减少了由视图集的过度冗余引起的额外时间开销。

为了解决草图和模型之间的尺寸不一致，Su[11] 等人采用了固定透视的方法。这种方法的具体过程是将模型放置在水平面上，使模型垂直向上。对于充当检索数据库的每个三维模型，将一个摄像机设置在其水平面上方30度的角度，并且摄像机指向三维模型的中心，每个摄像机得到模型的二维视图。图3-3。投影方法可以生成12个视图。此外，对于不满足直立向上的模型，将相机放置在模型周围的正二十面体上的二十个顶点上。用这个来获得视图。并形成一个最佳视图地图集。

图3-3 三维模型投影

潘婷 [12] 提出了一种基于球面投影的三维模型检索方法。如图3-4所示，它用于解决域的不匹配问题。对三维模型进行预处理后，在三维模型的外围外接一个球体，并在球体的半弧上以30 ° 的间隔放置一个摄像机，并确保相机镜头垂直于模型质心和相机之间的线。相机获得的每个视图后，将半弧旋转30度，然后重复该步骤，直到半弧返回其原始位置。如图所示之后，利用高斯差分和贝塞尔曲线提取折线图，利用草图和投影图像之间的关系构造分类器，得到模型的最优视图。

图3-4 球体投影

Christopher M. Cyr[13] 等人以指定的间隔 (5度) 对观察球进行采样，如图3-5所示，该图给出了对象视觉球空间被划分成若干个区域，每个区域对应一个二维视图。并且在迭代过程中，使用曲率匹配和视图相似性组来定义形状相似性度量，以将视图组合到各个方面。然后将生成的视图集输入数据库。

图3-5 物体视球空间

在对模型进行投影时，计算机图形学中常用的两种投影是平行投影变化 (正交投影) 和中心投影变换 (透视投影)。如图3-6所示。两者之间的区别在于，正交投影中使用的投影射线彼此平行，而透视投影变换中使用的投影射线是从选定的公共点导出的，射线以椎体的形式投射到三维模型中。该系统采用中心投影，即透视投影，因为中心投影在绘制草图时更符合用户的绘制方法。

图3-6 正交投影和透视投影

本文采用的固定投影的思想是通过一定量的母狗在Su的方法上固定要在中心投影的三维模型，选择固定角度和固定数量的投影进行投影。对于每个三维模型，虚拟摄像机以水平面上方30度的角度放置，指向三维模型的中心，投影方法可以生成6个视图。这6个视图构成了模型的最优图集。

三维模型的光照和材质添加

在该系统中，读取OFF模型后，自动计算模型每个顶点的法向向量。使用法向量，系统可以将照明效果添加到模型中。采用Phong照明模型。Phong照明模型的主要构建块包括3个组件: 环境照明，漫射照明和镜面照明。Phong照明模型的构成如图3-7所示。

图3-7 Phong光照模型

环境光: 光通常不是来自单个光源，而是来自散布在我们周围的许多光源，即使它们不是立即可见的。光的特征之一是它可以在多个方向上散射和反弹，到达不直接可见的点。因此，光线可以在其他表面上反射，并间接影响物体的照明。使用小的常数 (光) 颜色，并将其添加到对象片段的最终颜色中，因此即使没有直接光源，它也总是看起来总会有一些散射光。物体上的一点P的环境光强度可以表示

表示来自周围环境的入射光强，为材质的环境反射率。

漫射反射光: 模拟光物体对物体的方向性影响。这是照明模型中最重要的视觉部分。物体面对光源的部分越多，它就会变得越亮。兰伯特余弦定律总结了漫反射光的强度与光入射方向与物体表面法向量之间的角度之间的相关性。当 = 0时，物体表面正好垂直于光线的方向，是得到的最大光强; 当 = 90时，物体表面平行于光线的方向，此时光线无法照亮物体，光的强度是最弱的; 最后，物体的表面转向光的背面，物体的相应表面无法接收光。物体上的一点P的漫反射光强表示为:

它是光源发出的入射光的强度，物质的漫反射比，以及入射光与物体表面法向量之间的夹角，称为入射角。

镜面反射光: 镜面光分量模拟物体表面光滑时反射的亮光。镜面光通常反射光线的颜色，而不是物体的颜色。物体上的一点P的镜面反射光的光强度可以表示为:

为入射光光强，为材质的镜面反射率，镜面反射光光强与成正比。

在计算漫反射和镜面光分量时，应考虑矢量L，法矢量N，反射方向R和矢量V在观察者和顶点位置之间的关系。光的减弱。因此，可以定义Phong照明模型:

f(d)表示光照的衰弱，可以定义为：

为恒定衰减因子，为线性衰减因子，为二次衰减因子，d为光源位置到物体上点P的距离。

基于Phong光照模型的材质。通过多次实验，得到数据如表3-1 所示。

表3-1 Phong光照模型的材质

三维模型的空间变化

在渲染模型中，一个非常重要的功能就是对模型进行空间上的修改变化。把变换看成是一组顶点移动到一个新的位置的过程，这一组顶点描述成一个或者多个几何对象。尽管把一个顶点移动到另一个顶点位置有许多方式。但是把一组对象从一个位置移动到另一个位置，且他们之间的空间关系保持不变，极狐总是只有一种变换方式。因此，尽管可以找到多个矩阵，把渲染后的模型的某个顶点从一个位置移动到另一个位置，单是只存在一个矩阵，把它应用到该模型的全部的顶点后，该模型的大小仍然保持不变。

翻译操作。将平移定义为将同一方向上的所有点移动相同距离的操作。因此，只需要确定一个位置向量就可以确定平移变换。平移是在原向量上增加另一个向量，以在不同位置获得新向量，从而在位移向量的基础上移动原向量的过程。具体公式为:

规模转型。缩放是一种非刚性仿射变换，可以适当选择缩放、平移和旋转的组合顺序，以获得任何仿射变换。定义缩放变换是为了使几何对象变大或变小。缩放分为两种情况。一种是各个方向的均匀缩放变换。另一种是单一方向上的缩放变换。

缩放转换有一个固定点，因此要定义缩放更改，您必须首先确定一个固定点，缩放方向和缩放因子。如果，对象沿缩放方向变大; 如果在 (0,1) 之间，对象沿缩放方向变小; 如果为负，则指示沿以固定点为中心的缩放方向的反射变换。由于确定了固定位置和3个独立的缩放因子，因此缩放具有6个自由度。具体公式为:

旋转变换。旋转是一个很难定义的概念，它需要更多的参数。它与框架无关，具有普遍意义。需要3个参数。固定点，旋转角度以及旋转轴所在的线或矢量。给定某个固定点，旋转具有3个自由度。使用三角函数，给定一个角度，可以将一个向量转换为旋转的新向量。这通常是使用一系列正弦和余弦函数的巧妙组合 (通常称为sin和cos) 得到的。旋转矩阵对3D空间中的每个单元轴有不同的定义，旋转角用 θ 表示，则具体公式为:

绕X轴旋转：

绕Y轴旋转：

绕Z轴旋转：

三维模型投影系统结果

本文三维模型的投影系统采用OpenGL和OpenCV作为几何建模平台，ImGui作为UI界面，采用C ++ 语言。实现了一个简单的渲染器，主要功能有: 读取和逼真的模型; 旋转、缩放和移动模型; 为模型添加不同的照明效果 (平行光、点光源、聚光灯); 不同的材质效果; 3D模型的投影。渲染器的帧接口如图3-8所示。

图3-8 渲染器的框架界面图

用户步骤包括如下几步：

步骤1.选择需要渲染的三维模型。

步骤2.使用鼠标来旋转，缩放，移动三维模型。

步骤3.调整照明效果。您可以选择平行光，点光源和聚光灯。您可以分别调节这些灯的环境光，漫射光和镜面光。

步骤4.调节三维模型的材质。

第5步。按下键盘 “a” 和 “s” 以60度的固定水平旋转3D模型。

步骤6.按下保存按钮，保存该模型的一张二维视图。

第7步。按下自动截图按钮，批量处理3D模型，获得不同3D模型的2D视图集。

本文采用ModelNet40作为模型数据库。对数据库的部分模型进行渲染，渲染结果如图3-9所示。

图3-9 部分模型

本章小结

本章主要介绍了检索系统中使用的数据集ModelNet-40，解释了如何解析OFF模型文件格式，渲染和添加平行光，点光和聚光灯以及基于Phong照明模型的材料效果。提出了一种基于草图检索算法的视图集选择方法。在草图检索算法中，采用固定投影法，将得到的6个投影作为一个模型的二维图集。

草图的绘制处理

设计了一个简单的二维画板，以解决不同用户在线素描的功能需求。程序接口的大小默认为800x600，以避免不同的操作系统和不同的分辨率。屏幕显示黑色背景，您可以在其上绘制。单击鼠标右键，弹出绘图板的可选菜单栏，该菜单栏具有不同的绘图功能和操作功能。基本图形的绘图功能 (包括点、直线、矩形、三角形、圆形、曲线、铅笔线); 实现橡皮筋技术; 基本图形修改功能 (包括橡皮擦技术、草图保存技术)。

直线类图形绘制

Bresenham算法用于绘制直线。Bresenham算法在主位移方向上每次摇动增加一个单位。另一个方向的增量为0或1，取决于像素与理想直线之间的距离，该距离定义为d。

如图4-1 Bresenham线算法所示，线的斜率在间隔0-1中，因此X方向是主要的位移方向。假设当前像素，它是理想直线和下一个垂直网格的交点。并且假设线的起点是和位于网格点上，则可以定义的初始值为0。

图4-1 Bresenham直线算法

沿X方向增加一个单位是可用的。下一个候选点是或。根据点q的位置选择或点q的位置由直线的斜率决定。q点和像素点之间的误差项为。那时，像素更接近点q，选择，反之亦然。如果从点q到这两个像素点的距离相等，则可以选择任何像素。惯例选择。

因此，可以得到一个简单的的递推公式：

其中，递归公式的关键在于计算误差项。递归一个沿X方向的单位，有。一旦你在Y方向上走了一步，就把它减去1。由于只需要检查错误项的符号。因此，定义以消除小数的影响。重写上面的递推公式可以得到:

采取。对于沿X方向更新的每个单元，都有。当不小于0时，更新下一个像素。同时，它将更新为-1。相反，下一个像素是更新的。

直线

点击直线功能的按钮后，在绘图面板上，实现绘制直线的功能，在屏幕上鼠标的当前点绘制一个黑点，移动鼠标，按下的第一个点与此时鼠标位置上的点直线相连。当鼠标按钮被抬起时，一条直线就会产生，不再改变。当鼠标移动时，会产生一条直线，但会根据当前鼠标的位置而变化。达到橡皮筋的效果。

矩形

单击矩形功能的按钮后，在绘图面板上，实现了绘制矩形的功能，在屏幕上鼠标的当前点处绘制了一个黑点，并移动了鼠标，第一个按下的点与此时鼠标位置上的点相连，形成一个矩形。当鼠标按钮被抬起时，矩形被生成并且不再改变。当鼠标移动时，矩形会生成，但会根据当前鼠标位置而变化。达到橡皮筋的效果。鼠标两点之间的位置关系如图4-2所示。

图4-2 矩形绘制的位置关系

三角形

单击三角形功能的按钮后，在绘图面板上，实现了绘制三角形的功能，在屏幕上鼠标的当前点处绘制一个黑点，并移动鼠标，第一个按下点此时与鼠标位置上的点相连，形成一个三角形。当鼠标按钮被抬起时，矩形被生成并且不再改变。当鼠标移动时，会生成三角形，但会根据当前鼠标位置而变化。达到橡皮筋的效果。鼠标两点之间的位置关系如图4-3所示。

图4-3 三角形绘制的位置关系

曲线图形绘制

选择曲线按钮，通过单击屏幕上的任意四个点来绘制三次贝塞尔曲线。

给定n+1个控制点，则n次贝塞尔曲线定位为：

其中，为贝塞尔曲线的基函数，其表达式可以定义为：

当n = 3时，贝塞尔曲线的控制多边形有4个控制点，这些控制点将分别设置为，贝塞尔曲线是如图4-4所示的三次多项式。

图4-4 三次贝塞尔曲线

将该类贝塞尔曲线称其为三次贝塞尔曲线，将其定义为：

拆分开，既可以得到：

写成矩阵形式为：

圆形绘制

对于圆形绘制，有三种常用的方法: 第一种是使用Bresenham中点绘制圆来绘制圆。具体想法类似于Bresenham绘制直线。第二种方法是将正多边形细分为大量的细分。当多边形中有足够多的变量，并且多边形的边长足够小时，可以得到一个近似圆。第三种方法是用贝塞尔曲线画一段弧线，将多个弧线段拼接在一起，得到一个完整的圆。

该系统采用第三种方法，可以通过使用三次贝塞尔曲线来模拟1/4单位圆。如图4-5所示。假定坐标是 (0,1)，坐标是 (m，1)，坐标是 (1，m)，坐标是 (1,0)。

图4-5 贝塞尔曲线模拟1/4圆弧

对于一段三次贝塞尔钱，其参数表达式为：

将代入，对于圆弧的中点，取t=0.5，则有：

引入控制点的坐标，可以得到m的近似0.5523，称为魔术常数。

单击圆圈功能的按钮后，在绘图面板上，实现了绘制圆圈/椭圆的功能，在屏幕上鼠标的当前点绘制黑点，并移动鼠标，第一个按下点此时与鼠标位置上的点相连，形成一个圆。当鼠标按钮被抬起时，圆圈就会产生并且不再改变。当鼠标移动时，会产生一个圆圈，但会根据当前鼠标的位置而变化。达到橡皮筋的效果。根据鼠标的位置，它显示为圆形或椭圆形。鼠标两点之间的位置关系如图4-6所示。

图4-6 圆形绘制的位置关系

铅笔线绘制

点击铅笔功能的按钮后，在绘图面板上，实现现实生活中的铅笔功能，在屏幕上的鼠标当前点画一个黑点，移动鼠标，这些黑点将连接成任意一条线，可以是直线或曲线。

草图绘制的结果

本文三维模型的投影系统采用OpenGL和OpenCV作为几何建模平台，MFC作为UI接口，采用C ++ 语言。实现了一个简单的草图绘图板。主要功能有: 绘制基本图形 (点、直线、矩形、三角形、圆、曲线、铅笔线) 和基本图形修改 (橡皮擦技术、草图保存技术) 简单草图绘图板的框架界面如图4-7所示。用户步骤包括以下步骤:

步骤1.点击绘制草图按钮。

步骤2.调节弹出窗口的大小，准备绘制。

步骤3.点击鼠标右键，弹出绘制菜单，并选择要绘制的图形。

步骤4.绘制图形

第5步。可选步骤，进入上述菜单，选择 “修改图形” 或 “清除画板”，然后重新绘制。

步骤6.按下保存按钮，保存所绘制的草图。

图4-7 简易草图画板的框架

本章小结

本章主要介绍了检索系统中使用的简单草图绘图板的设计，其中增加了一些基本的绘图功能，包括直线、矩形、三角形、圆形、曲线，导出了用于Bresenham算法和三次Bessel de-curve算法的铅笔线。同时应用了这两种算法，如使用Bresenham算法绘制矩形和三角形等直线图形，也应用了贝塞尔曲线算法，如使用多段贝塞尔曲线绘制圆。

切勿删除行尾的分节。“结论” 的所有先前正文内容都应该写在这一行之前。

草图及三维模型的特征提取

在基于手绘草图的三维模型检索问题中，需要计算所有源模型和目标模型之间的相似度，以便找到与目标模型最相似的源模型。随着三维模型的不断增加，这种方法显然效率很低。在实践中，我们发现直接将草图与3D模型或3D模型的2D视图集进行比较是没有意义的。因此，需要一个能够表达二维视图和草图特征的描述符。为了能够解决二维视图和草图之间的旋转和比例大小的差异，本文所需的特征描述符应具有平移、比例和旋转不变性的特征。在计算机图形学领域中已经使用了大量的描述符，包括诸如HOG，SIFT，Zernike和shape context的描述符。本章主要研究Zernike描述符、傅立叶描述符、二维形状分布和集成描述符，本章中的集成描述符可以解决单个描述符的不完全特征提取问题。

全局试图特征描述符

全局特征是指图像的整体属性。常见的全局特征包括颜色特征，纹理特征和形状特征，例如强度直方图。由于是像素级低层视觉特征，全局特征具有良好的不变性、计算简单、直观表示等特点，但高特征维数和大量计算量是其致命弱点。此外，全局特征描述不适用于图像混叠和遮挡。

本文提出的全局视图特征包含Zernike矩和傅立叶描述符。首先，本文提取了二维视图的Zernike矩，并使用标准矩的方法将其归一化到 (0,1) 范围。然后，获得二维视图的等高线视图，提取其一维傅里叶算子，除以DC分量并标准化为 (0,1) 的范围。基于草图的3D模型检索不应受到草图绘制位置，比例大小和旋转角度的影响。

Zernike矩

图像的矩通常描述修改后的图像的全局特征，并提供有关不同类型图像的大量几何特征信息，例如大小，位置，方向和形状。

在基于手绘草图的三维模型检索摘要中，一个核心问题是图像的特征提取。一个简单的描述就是用一组相对简单的数据来描述整个图像，这组数据越简单越有代表性越好。良好的特征矩不会观察到光，噪声和几何变形的干扰。

Zernike矩是一个正交矩，是一个基于Zernike多项式正交化的函数。Zernike矩具有以下特征: 完备性，正交性和无变形的旋转。泽尼克时刻是一个复杂的时刻。通常，Zernike矩的模块用作描述物体形状的特征。目标对象的特征矩可以用一小组Zernike矩特征向量表示。低阶矩阵和特征向量描述图像目标的整体形状，高阶矩特征向量描述图像目标的细节信息。

定义在单位圆上的一组复值函数具有完备性和正交性，使得它可以表示任何定义在单位圆上的平方可积函数，定义:

其中，表示从点到 (x，y) 的向量长度，并表示向量与x轴逆时针方向的夹角。是一个实值径向多项式:

称其为为Zernike多项式。

Zernike多项式满足正交性：

其中，共轭多项式。而由于Zernike多项式具有正交完备性，单位圆中的任意图像f(x，y) 都可以用唯一的公式展开:

式子中就是Zernike矩:

Zernike矩的平移和缩放变换给出了具有平移，缩放和旋转不变性的图像g(x，y)。g(x，y) 的泽尼克矩定义为:

其中，; 表示视图轮廓中所有黑色像素的横坐标之和，表示视图轮廓中所有黑色像素的纵坐标之和。表示投影视图中所有白色像素的总和。() 表示手绘草图的中心。

由于Zernike矩定义在单位圆中，因此映射在单位圆之外的像素点不参与计算。这就是泽尼克矩的固有几何误差。Zernike矩计算时间补码长度，因此需要选择快速算法来减少时间。

徐丹华等人提出了Zernike矩的快速算法。利用Zernike多项式的迭代性质，找到了Zernike正交矩之间的内在关系，从而可以在Chan[15] 等人提出的一维几何矩有效算法的基础上，从低价的Zernike矩中获得高阶的Zernike矩，获得了一种快速算法。与现有方法相比，该算法大大减少了求解过程中的乘法次数，降低了计算复杂度，从而提高了运算速度和效率。

Fourier描述符

傅立叶描述符的作用是描述图像的轮廓信息，具有平移、旋转和尺度不变性的特点。对于图像，通过傅立叶描述符获得图像轮廓信息，这本质上是一个空间和频域变换问题。通过对图像中的像素进行傅立叶变换来获得图像的轮廓信息。

如果对傅立叶描述符进行低通滤波，则傅立叶描述符的低频分量捕获物体的一般形状特征，高频分量捕获更精细的细节，而傅立叶描述符不考虑空间位置，因此，傅立叶描述符的作用与矩非常相似: 低阶项/矩给出一个近似形状，添加附加项可以细化形状。

傅立叶形状描述符的基本思想是利用对象边界的傅立叶变换作为形状描述，利用区域边界的闭合性和周期性将二维问题转化为一维问题。从边界点导出三种形状表达式，即曲率函数、质心距离和复坐标函数。

假设特定形状的边界具有从0到n-1的n个像素。第k个像素沿轮廓的位置定义为 ()。对于曲线上的一点，可以用复数表示:

对S(t)进行傅立叶变换可以得到：

其中a(k) 是傅立叶描述符，需要对其进行归一化，以便具有平移，缩放和旋转不变性

归一化后的模为：

该描述符具有平移、比例和旋转不变特性。其中平移不变性和旋转不变性是其自身的特点，归一化和绝对值运算保证了尺度不变性。

D2描述符

二维形状分布基于三维形状分布。三维形状分布的主要思想是利用形状函数表示三维模型表面点对的距离关系、模型的内角关系和区域的面积，构造表示形状特征的统计分布直方图。类似地，二维形状分布可以构造二维视图的统计分布直方图。二维形状主要包括五个形状函数: A3，D1，D2，D3和d4。

A3: 在模型的表面上随机选择三个点，由三个点形成的三角形内角的概率分布构成A3形状特征。

D1: 连接模型表面上的任意点，从模型中心生成的距离概率分布构成D1形状特征。

D2: 选择模型曲面上连接的任意两点，两点之间距离的概率分布构成D2形状特征。

D3: 选择模型曲面上的任意三个点，三个点形成的三角形区域的平方根的概率分布构成D3形状特征。

D4: 选择模型曲面上的任意四个点，由四个点形成的立方体体积的概率分布构成D4形状特征。不同的形状分布描述符如图5-1所示。

图5-1 不同的形状分布描述符

本文采用较好的D2形状描述符来描述二维视图的形状区域特征。本文在每个2D视图上采用随机抽样，并输入3D模型的草图。所使用的样本点并不全部在外边界上，每个样本点可以位于给定视图的任何边缘线上。二维形状分布特征提取的算法描述如下:

1.在素描过程中，记录草图中绘制的所有像素，并将这些像素放入集合中。N是所有像素的总数。对像素进行随机抽样，1024本文选取的样本数。

2.对于1024随机采样点，可以知道二维形状描述符生成的所有点对的公式，如下所示:

3.计算所有点对之间的欧几里得距离。

4.计算随机点对之间的距离，构造二维形状分布直方图。统计分布直方图的横坐标应从0到点对之间的最远距离。在计算两个直方图之间的相似度时，应确保直方图间隔的数量相同。对于间隔较少的直方图，应使用归零方法填充间隔。

集成描述符

经过大量实验，表明傅立叶描述符中的较小值适用于表示全局视图特征。Zernike描述符中较大的值适用于表示全局视图特征。本文中的傅立叶描述符和Zernike描述符都选择经验值 [16]。其中，傅立叶描述符选择前10个较小的值; 泽尼克矩描述符使用前35个较大的矩。对于草图和三维模型的二维视图，本文使用向量的L1范式来测量傅立叶描述符、Zernike描述符和D2描述符之间的距离，具体如下:

其中F(p) 和Z(q) 分别表示草图和二维视图的傅立叶变换特征和Zernike矩特征。，分别表示草图和二维视图的二维分布特征。，和分别是使用傅立叶描述符、泽尼克矩描述符和D2形状描述符的二维草图和投影视图之间的相似性结果。其中，傅里叶描述符和泽尼克矩描述符构成算法的全局视图特征描述符，公式为:

为了解决单一描述符对二维视图特征提取不完全的问题，本文在全局视图描述符和二维形状分布的基础上集成一定权重，得到一个更好地代表二维视图特征的描述符。定义了集成描述符D:

是全局视图描述符的权重，该权重与二维形状分布的权重相加为1。草图和二维视图之间的相似性越高，D越小。不同的用户有不同的绘画风格，绘制的草图可能差异很大，集成的描述子特征也会不一致。因此，本文有以下方法来避免检索结果不准确。首先，本文提出的集成描述符具有平移、尺度和旋转的特点，避免了旋转和大小对检索结果的影响。因此，只有绘画风格对素描有影响，但是，绘画风格存在两个问题。1) 画家对模型观察角度的差异影响了集成描述符的特性，因此本文采用多投影的方法，尽可能考虑不同用户观察角度的差异。2) 本文中的草图检索对绘图有一定的要求。绘图越准确，检索效率越高，检索的准确性也越高。

Imin [17] 用不同的值进行了实验。[0-1] 之间具有每0.1的增长率。在不同的值下，记录3D模型的召回率 (召回率是检索时检索到的所有模型与所有模型的比例) 是0.1时的精确比 (精确比是检索到的此类模型总数的比例)。在 [0.3-0.6] 的范围内，集成描述符具有更好的检索性能。对于0.4来说，检索效果最好。在后续验证了本文提出的算法和其他算法的优缺陷，值0.4。不同的搜索结果如图5-2所示。

图5-2 不同的检索结果

本章小结

本章主要介绍了集成描述符，并详细分析了集成描述符的特征构造，它是通过将Zernike矩和Fourier描述符与D2形状特征组合的全局视图特征按一定比例加权得到的。集成描述符作为特征提取的描述符，可以解决单个描述符对二维草图信息描述不完整的问题。

千万。

基于手绘草图的三维模型检索

相似性度量，即综合评估两个事物之间相似程度的度量。两个事物越接近，它们的相似性度量就越大，而两个事物越遥远，它们的相似性度量就越小。相似度通常以数值表示: 当数据样本更相似时，相似度度量越高。它通常通过转换表示为0和1之间的数字: 0表示相似度低 (数据对象不同)。1表示高度相似 (数据对象非常相似)。相似性度量的方法有很多种，一般是根据实际问题选择的。常用的相似度有: 相关系数、角度相似度。本文使用距离来衡量样本之间的相似程度。

相似性计算

距离测度是数学的基本变化。对于定义在两个向量X和Y上的任何函数d(X，Y) 只要满足以下四个性质，就可以称为距离度量。

1)非负性: d(X,Y) >= 0；

2)对称性: d(X,Y) = d(Y,X)；

3)自反性: d(X,Y) = 0，当且仅当 X= Y；

4) 三角形不等式: d(X，Y) d(Y，Z) >= d(X，Z)。

常见的几种距离度量如图6-1所示。

图6-1 常见的距离度量

欧几里得距离

欧几里得距离也称为欧式距离，它是最常见的距离测量方法:

对欧几里得距离的直观理解是特征空间中X和Y两点之间的直线距离。距离度量与矢量度量的长度密切相关。欧几里得距离也可以视为差向量x-y的长度。向量的长度在数学中也称为范数。欧几里得距离对应于向量范数，也可以表示:

曼哈顿距离

曼哈顿距离也称为块距离，是两点在欧氏空间的固定直角坐标系上形成的线段投影的距离之和。定义从a点到B点的曼哈顿距离是两点坐标之差的绝对值之和。曼哈顿军力对应向量的范数可以表示为:

切比雪夫距离

两个n-d观测值或向量之间的切比雪夫距离等于数据样本的坐标变化的最大绝对值。在二维世界中，数据点之间的切比雪夫距离可以确定为其二维坐标的绝对差之和。因此，切比雪夫距离被定义为:

例如，在国际象棋中，国王和王后所行的两点之间的最小平方数可以用切比雪夫距离来衡量。数学上，切比雪夫距离对应于向量的范数:

闵可夫斯基距离

闵可夫斯基距离是对之前的距离测量的推广: 欧几里得距离、曼哈顿距离和切比雪夫距离。定义为在n-d空间中两个观测值之间的距离，公式为:

闵可夫斯基距离对应于向量的范数。不同的p可以得到不同的距离度量。显然，欧几里得距离和曼哈顿距离都是闵可夫斯基距离的特例，分别对应于p = 1和p = 2的特例。事实上，切比雪夫距离也是闵可夫斯基距离的特例，对应于p趋向正无穷大。

坐标原点距离为1的点在欧氏距离度量下的轨迹是如图6-2所示的单位圆，并成为曼哈顿距离度量下单位圆的内切正方形，随着闵可夫斯基距离中的p不断增加，单位圆继续向外扩展，直到p趋近正无穷，演化为外接正方形。

图6-2 不同距离度量下的单位圆

马氏距离

尽管上述距离相对简单，但它们也有明显的缺点。对样本不同属性之间的差异 (即各指标或变量) 一视同仁，这一点有时不符合实际要求。马氏距离表示数据的协方差距离。这是计算两个未知样本集相似度的有效方法。与欧几里得距离的区别在于，它考虑了各种特性之间的联系，并且与尺度无关，即与测量尺度无关。马氏距离也可以定义为服从相同分布且协方差矩阵为 Σ 的两个随机变量之间的差异程度。如果协方差矩阵是单位矩阵，则马氏距离简化为欧几里得距离。如果协方差矩阵是对角矩阵，也可以称为归一化欧几里得距离。

对于均值为且协方差矩阵为 Σ 的多元向量x，马氏距离为:

基于手绘草图的三维模型检索结果

在分析了上述距离计算方法后，本文利用最简单的欧氏距离公式计算了草图与二维视图图集的相似度。基于该相似性算法，设计了基于手绘草图的三维模型检索。

本文三维模型的检索系统采用OpenGL和OpenCV作为几何建模平台，MFC作为UI接口，采用C ++ 语言。实现了一种检索程序系统。主要功能是: 导入和绘制草图，绘制草图功能的具体实现是指第4章简单的草图绘图板的实现; 可以选择不同的数据库; 显示绘制的草图; 图形检索功能; 显示8个最优结果图和一个全局最优结果。寻回犬的框架界面如图6-3所示。

图6-3 检索器的框架界面

用户步骤包括如下几步:

步骤1。用户选择绘制在线草图或导入已经绘制的草图的图片。如果按下绘制草图，则会弹出一个简单的草图绘图板，供用户绘制草图。具体功能请参见第4章。

步骤 2.在程序的左侧会显示出用户导入或者绘制好的草图。

步骤3.选择用于检索3D模型库的路径。默认情况下，有一个较小的数据库。用户可以通过简单渲染器的程序获得不同数量级的模型数据库。有关简单渲染器的具体功能，请参阅第3章。

步骤 4.点击检索，进行草图检索。

步骤 5.输出结果，返回8张结果图。

第6步。输出结果并返回最接近的结果图。以显示器为例，进行基于手绘草图的搜索，搜索结果如图6-4所示。

图6-4 草图检索结果

本章小结

本章主要介绍了基于不同距离公式的相似性度量，并推导出不同距离公式的计算方法和范数。本文利用欧式距离计算了二维视图与草图的相似度，并对手绘草图的检索结果进行了从小到大的排序和显示。本章还重点介绍了基于手绘草图的三维模型检索系统的应用。草图搜索过程的第一步是绘制草图。为了降低对用户的要求，提供了在线绘图或离线绘图功能。只需要简单的草图就可以完成搜索，实验结果也表明本文的方法可以有效地检索相关的3D模型。

千万不。

结论

随着互联网技术和计算机辅助设计的飞速发展，三维CAD模型的数量急剧增加。如何从海量模型数据中快速准确地检索出所需的模型变得尤为重要。对现有工业3D模型进行有效检索，可以促进新模型的设计，并节省大量时间，人力和物力。计算机辅助设计 (Computer Aided Design，CAD) 利用计算机及其辅助设备帮助设计人员进行计算、信息存储和图形设计，从而大大减少了设计人员的工作量，缩短了产品设计周期，提高了产品设计质量。

本文叙述了三维模型检索技术的国内外研究现状，综述了三维模型检索技术的发展及主要方法。本文主要研究了基于手绘草图的三维模型检索。在基于草图的检索过程中，首先使用OpenGL对模型进行渲染，并使用固定投影获得模型点的6个二维视图图集。为用户提供了不同的绘制方法，以便用户可以在线或离线绘制草图，以提高识别精度。使用了一个新的描述符，即集合描述符，该描述符有效地利用了视图的区域特征和边界轮廓特征，最后使用距离公式来完成检索。实验表明，本文方法效果较好。基于草图的检索算法相对于其他算法的优点是，它方便普通用户检索，并且可以在不需要大量三维模型的详细信息的情况下检索到更准确的三维模型。

本文以OpenGL和OpenCV为几何建模平台，以MFC和ImGui为UI接口，以C ++ 为编程语言开发三维模型检索。该系统集成了本文提出的所有算法，包括简单的模型渲染，简单的草图绘制和3D模型检索。运用ModelNet-40完成模型库，验证了本文提出的算法的优良机能和检索体系的人性化开辟。

上述研究取得了一些阶段性成果，但仍有许多内容需要进一步研究，主要是:

1.由于一些三维模型结构复杂，不同的用户在绘制草图时容易带来一定的误差因素，如行程闭合等，因此需要降噪，手绘草图降噪的目的是消除因聚点而产生的毛刺，封闭处的缝隙，用户不小心或无意识地绘制造成的提笔或收笔，以防止手绘草图过于粗糙，手绘草图与用户的初衷大相径庭。

2.由于当前三维模型的复杂性，相似度计算为欧几里得距离。因此，当模型数量急剧增加时，检索的准确性会在一定程度上下降。

致谢

时光飞逝，时光飞逝。四年的大学生活即将结束，收获太多，不舍。春梦秋云，聚散无常。随着离校日期的临近，毕业论文的完成也告一段落。在这短短的四年的学习之旅中，我不仅学到了很多知识和技能，将会在今后的工作中使用，而且学到了一些人生的经验，收获了与同学们宝贵的友谊。

谢谢你，我非常尊敬的导师高雪瑶先生。这篇论文是在高老师的悉心指导下完成的。在学术研究中，高先生坚持专注、负责、精益求精的学术态度，孜孜不倦的高尚师德。我不仅树立了崇高的学习目标，而且懂得了学会与人打交道的道理。在校期间，在高老师的指导下，我也逐渐做了几个项目，收获很大。从宣帝到完成，这篇论文离不开老师的良苦用心，倾注了老师的大量心血。在此，我谨向高先生表示衷心的感谢。感谢大学里其他老师对我学习和生活的关心和帮助。感谢余林森老师和李莎莎老师，感谢我听完我的问题，课后耐心解答。感谢各位老师提供了一个学习氛围浓厚的校园环境，

在四年前的一次选择中，我独自来到哈尔滨学习，跨越了半个中国。很荣幸能完成我在哈尔滨理工大学的本科学习。在我的学校生涯中，我最想感谢的是我的父母。我学习已经16年了。是他们默默的努力激励着我去学习。感谢母亲为了让我在一个更好的学习环境中，去家乡城市的所有学校旅行。多亏了我父亲，在我的初中和高中，我每周都会开车送我去石狮和厦门。在哈尔滨工业大学的四年里，我感谢我的五个室友。你的公司让我明白了友谊的价值。

感谢答辩组的各位老师，感谢老师们在百忙之中抽出时间参加硕士生毕业答辩，并祝愿老师们身体健康，心想事成，在今后的工作和生活中万事如意。

参考文献

[1] Sezgin T M,Stahovich T,Davis R. Sketch-based interface: early processing of sketch understanding [C]. Seminar on Perceived User Interface. ACM,2001:1-8

[2] Li B, Lu Y,Fares R. 3D model retrieval based on semantic sketch [C]. IEEE multimedia and expo symposium international conference. IEEE,2013:555-558

[3] yuan juefei, Abdul Rashid Hameed,Li Boet al. Comparison of three-dimensional scene shape retrieval methods [J] computer vision and image understanding, 2020,201:103-120

[4] Zhu Z, Wang X, Bai S, et al. Depth Learning Representation for 3D Shape Retrieval Using Automatic Encoder [J]. Neural Computing, 2016,204:41-50.

[5] Konstantinos Sfikas,Ioannis Pratikakis,Theoharis Theoharis. Panorama-based Convolutional Neural Network Integrated 3D Model Classification and Retrieval [J] Computer and Graphics. 2018:208-218

[6] 周岩，曾梵志，吴晨，罗玥，刘子钦.基于深度学习的三维形状特征提取方法 [J].计算机科学。2019(09):47-58

[7] 张云峰.图像与三维模型匹配方法的研究与应用 [D].南京: 南京大学，2019.[8] 张一坤，唐艳，陈强.基于多特征融合的三维模型检索 [J].郑州大学学报 (工学版).2019(01):1-6

[9] 李海生，孙莉，吴晓群，蔡强，杜俊平.基于模型中二面角分布直方图的非刚性三维模型检索 [J].计算机辅助设计与图形学杂志。2017(06): 1128-1134

[10] 安伯青。基于手绘草图的三维模型检索的研究与实现 [D].西北大学，2017。

[11] suh, maji, Kalogerakis, etc. Multi-view convolutional neural network for 3D shape recognition [C]// Pro-ceedingsoof ICCV2015,2015

[12] c. M. Cyr and B.B. Kimia. A three-dimensional object recognition aspect graph method based on similarity. 57(1). 2004. 3

[13] 潘婷。基于草图的三维模型检索方法研究与应用 [D].中北大学2020

[14] 徐丹华，顾佳，李松宜，舒华中.Zernike矩的快速算法 [J] 东南大学学报2002(03):32:2

[15] Chan FHY,Lam FK. A full-adder contraction structure for fast calculation of moments [J]. JVLSI signal processing, 196,12:159 175.

[16] DALAL N,TRIGGS B. Histogram of directional gradients for human detection [C]. Computer Vision and Pattern Recognition, 2005.CVPR 2005.IEEE Computer Society Conference, 2005,1: 886-893.

[17] 一鸣。基于内容和草图的三维模型检索方法 [D].哈尔滨理工大学.2020

[18] 强慧英，李玉红，王洪深，等.基于HMM和仿射不变矩三维模型的分类与线缆检测算法 [J].制造自动化，2019 41(3): 40-46.

附录A

附录A主要为三维模型投影程序的部分主要代码的实现

1.模型的光照 CLight.h

#pragma once

#include "ggl.h"

class CLight

{

public:

CLight();

~CLight();

public:

GLenum m\_LightIdentifier;

public:

void SetAmbientColor(float r,float g,float B ,float a);

void SetDiffuseColor(float r,float g,float B ,float a);

void SetSpecularColor(float r,float g,float B ,float a);

void Enable();

void Disable();

};

Class Direction Light: Public Lighting

{

public:

CDirectionLight(GLenum light);

void SetPosition (float x, float y, float z);

};

Class Light: Public Lighting

{

public:

CPointLight(GLenum light);

void SetPosition (float x, float y, float z);

void setconst decay (floating v);

void setlinear attenuation (float v);

void setquadric attenuation (floating v);

};

CSpotLight class: public CPointLight

{

public:

CSpotLight(GLenum light);

void SetDirection(float x,float y,float z);

void SetExponent(float v);

void SetCutoff(float v);

};

2.模型的光照 CLight.cpp

#include "CLight.h"

void CLight::SetAmbientColor (floating point r, floating point g, floating point B, floating point a)

{

Floating environment color [] = {r,g, B, a };

glLightfv(m\_LightIdentifier,gl\_ambientcolor);

}

void CLight::SetDiffuseColor (float r, float g, float B, float a)

{

Floating diffuse color [] = {r,g, B, a };

glLightfv(m\_LightIdentifier,gl\_diffusecolor);

}

void CLight::SetSpecularColor (float r, float g, float B, float a)

{

Floating sepcularColor[] = {r,g, B, a };

glLightfv(m\_LightIdentifier,GL\_SPECULAR,sepcularColor);

}

void CLight::Enable()

{

glEnable(GL\_LIGHTING);

glEnable(m\_LightIdentifier);

}

void CLight::Disable()

{

glDisable(GL\_LIGHTING);

glDisable(m\_LightIdentifier);

}

CDirectionLight::CDirectionLight(GLenum light)

{

m\_LightIdentifier = light;

}

void CDirectionLight:: set position (floating x, floating y, floating z)

{

Floating position [] = {x,y,z,0.0f };

glLightfv(m\_LightIdentifier,GL\_POSITION,pos);

}

CPointLight::CPointLight(GLenum light)

{

m\_LightIdentifier = light;

}

void CPointLight:: set the position (floating point x, floating point y, floating point z)

{

Floating position [] = {x,y,z,1.0f };

glLightfv(m\_LightIdentifier,GL\_POSITION,pos);

}

void CPointLight:: setconst attenuation (floating v)

{

glLightf(m\_LightIdentifier,gl\_constant\_defailation,v);

}

void CPointLight:: set linear attenuation (floating v)

{

glLightf(m\_LightIdentifier,gl\_linear\_defier,v);

}

void CPointLight:: set secondary attenuation (floating v)

{

glLightf(m\_LightIdentifier,gl\_quadratic\_defier,v);

}

CSpotLight::CSpotLight(GLenum light) : CPointLight(light)

"{}"

void CSpotLight:: set the direction (floating point x, floating point y, floating point z)

{

Floating directory [] = {x,y,z,1.0f };

glLightfv(m\_LightIdentifier,GL\_SPOT\_DIRECTION,dir);

}

void CSpotLight:: setexclusive (floating v)

{

glLightf(m\_LightIdentifier,GL\_SPOT\_EXPONENT,v);

}

void CSpotLight::SetCutoff (floating v)

{

glLightf(m\_LightIdentifier, GL\_SPOT\_CUTOFF, v);

}

3. 模型的加载 CMyModle.h

#pragma once

#include "ggl.h"

typedef struct Vertex

{

float x;

float y;

float z;

} Vertex;

typedef struct Face

{

Face (void): vert\_number(0),verts(0) {};

int vert\_number;

Vertex\*\* verts;

float normal[3];

} Face;

typedef struct MyMesh

{

MyMesh(void): vert\_number(0),verts(0),face\_number(0),faces(0) {};

int vert\_number;

Vertex\* verts;

int face\_number;

Face\* faces;

vector<Vertex> point;

} MyMesh;

class CMyModel

{

public:

CMyModel();

Cmmodel (constant character \* file name);

~CMyModel();

public:

MyMesh\* ReadOffFile();

MyMesh \* reread (constant character \* new path);

void get\_normal(Face& face);

void draw\_faces();

void draw\_points();

void draw\_lines();

void SetAmbientMaterial(float r,float g,float B ,float a);

Void set diffusion material (float r, float g, float B, float a);

void SetSpecularMaterial (float r, float g, float B, float a);

void SetShininessMaterial (floating s);

void SetMaterial();

public:

const char\* m\_filename;

MyMesh\* m\_mesh;

float m\_AmbientMaterial[4];

float m\_DiffuseMaterial[4];

float m\_SpecularMaterial[4];

float m\_ShininessMaterial[1];

};

4.CMyModel.cpp

#include "CMyModel.h"

Cmmodel:: cmmodel (constant character \* file name)

{

m\_filename = filename;

m\_mesh = NULL;

memset(m\_AmbientMaterial,0,sizeof(m\_AmbientMaterial));

memset(m\_DiffuseMaterial,0,sizeof(m\_DiffuseMaterial));

memset(m\_SpecularMaterial,0,sizeof(m\_SpecularMaterial));

memset(m\_ShininessMaterial,0,sizeof(m\_ShininessMaterial));

}

Invalid cmenodel::get\_normal (face and face)

{

Face. Normal [0] = face. Normal [1] = face. Normal [2] = 0;

Vertex \* v1 = face.verts[face.vert\_number - 1];

For (int I = 0; I <face.vert\_number; I ++)

{

Vertex\* v2 = face.verts[i];

Face. Normal [0] + = (v1->y \* v2->z) - (v1->z \* v2->y);

Face. Normal [1] + = (v1->z \* v2->x) - (v1->x \* v2->z);

Face. Normal [2] + = (v1->x \* v2->y) - (v1->y \* v2->x);

v1 = v2;

}

float squared\_normal\_length = 0.0;

Square\_normal\_length + = face. Normal [0] \* face. Normal [0];

Square\_normal\_length + = face. Normal [1] \* face. Normal [1];

Square\_normal\_length + = face. Normal [2] \* face. Normal [2];

float normal\_length = sqrt(squared\_normal\_length);

if (normal\_length > 1.0E-6)

{

Face. Normal [0] /= Normal\_Length;

Face. Normal [1] /= Normal\_Length;

Face. Normal [2] /= Normal\_Length;

}

}

MyMesh \* cmmodel:: read file ()

{

FILE\* fp = NULL;

errno\_t err;

If (! (fp = fopen(m\_filename,“r”)

{

cout << "无法打开文件" << endl;

return 0;

}

m\_mesh = new MyMesh();

int vert\_number = 0;

int face\_number = 0;

int line\_number = 0;

int line\_count = 0;

char buffer[1024];

while (fgets (buffer, 1023,fp))

{

line\_count++;

char\* bufferp = buffer;

And (isspace(\* bufferp)) bufferp ++;

If (\* bufferp = = '#') continues;

If (\* bufferp = = '\ 0') continues;

if (vert\_number == 0)

{

if (!strstr(bufferp, "OFF"))

{

If ((sscanf(bufferp,"% d",& vert\_number,& face\_number,& line\_number) ! = 3) | | (vert\_number = = 0)) {

Coout << "ERROR: 有语法错误!"<< endl;

fclose(fp);

return NULL;

}

m\_mesh->verts = new vertex [vert\_number];

assert(m\_mesh->verts);

m\_mesh->faces = new face [face\_number];

assert(m\_mesh->faces);

}

}

Otherwise, if (m\_mesh->vert\_number < vert\_number)

{

Vertex & vert = m\_mesh->verts[m\_mesh->vert\_number ++];

If (sscanf(bufferp, "% f", &(vert.x),&(vert.y),&(vert.z)) ! = 3)

{

Coout <<“错误: 在点的信息中，数据量不足 (3)” << endl;

fclose(fp);

return NULL;

}

}

Otherwise, if (m\_mesh->face\_number < face\_number)

{

char\* pTmp = NULL;

Face & face = m\_mesh->faces[m\_mesh->face\_number ++];

bufferp = strtok(bufferp,"\ t");

if (bufferp)

Face. vert\_number = atoi(bufferp);

else

{

fclose(fp);

return NULL;

}

face.verts = new vertex \* [face.vert\_number];

assert(face.verts);

For (int I = 0; I <face.vert\_number; I ++)

{

char\* pTmp = NULL;

bufferp = strtok(NULL,"\ t");

if (bufferp)

face.verts[ I] = &(m\_mesh->verts[atoi(bufferp)]);

else

{

fprintf(stderr, "error: face syntax error for% d line in file% s \n", line\_count,m\_filename);

fclose(fp);

return NULL;

}

}

get\_normal(face);

}

else

{

Coout << "ERROR: format ERROR! "<< endl;

break;

}

}

If (face\_number! = m\_mesh->face\_number)

{

Coout <<“错误: 面数与实际” << endl不匹配;

}

fclose(fp);

return m\_mesh;

}

MyMesh \* cmmodel:: reread (constant character \* new path)

{

memset(m\_mesh,0,sizeof(m\_mesh));

m\_filename = newpath;

m\_mesh = ReadOffFile();

return m\_mesh;

}

void CMyModel::draw\_faces()

{

//if (m\_mesh != NULL)

{

For (int I = 0; I <m\_mesh->face\_number; I ++)

{

Face & face = m\_mesh->faces[ I];

glColor3f(1.0f, 1.0f, 1.0f);

glBegin(GL\_POLYGON);

glNormal3fv(face.normal);

For (int j = 0; j < face.vert\_number; j ++)

{

Vertex\* vert = face.verts[j];

glVertex3f (vertical->x, vertical->y, vertical->z);

}

glEnd();

}

}

}

void CMyModel::draw\_points()

{

glColor3f(1.0f, 1.0f, 1.0f);

glPointSize(3);

glBegin(GL\_POINTS);

For (int I = 0; I <m\_mesh->vert\_number; I ++)

{

glVertex3f(m\_mesh->verts[ I ].x

m\_mesh->verts[i].y,

m\_mesh->verts[i].z);

}

glEnd();

}

void CMyModel::draw\_lines()

{

Double temp\_x,temp\_y,temp\_z;

For (int I = 0; I <m\_mesh->face\_number; I ++)

{

Face & face = m\_mesh->faces[ I];

glColor3f(1.0f, 1.0f, 1.0f);

glBegin(GL\_LINES);

For (int j = 0; j < face.vert\_number; j ++)

{

Vertex\* vert = face.verts[j];

if (j == 0) {

temp\_x = vert->x;

temp\_y = vert->y;

temp\_z = vert->z;

continue;

}

glVertex3f(temp\_x,temp\_y,temp\_z);

glVertex3f (vertical->x, vertical->y, vertical->z);

temp\_x = vert->x;

temp\_y = vert->y;

temp\_z = vert->z;

}

glEnd();

}

}

void cmenodel::SetAmbientMaterial (floating point r, floating point g, floating point B, floating point a)

{

m\_AmbientMaterial[0] = r;

m\_AmbientMaterial[1] = g;

m\_AmbientMaterial[2] = b;

m\_AmbientMaterial[3] = a;

}

void cmenodel::SetDiffuseMaterial (float r, float g, float B, float a)

{

m\_DiffuseMaterial[0] = r;

m\_DiffuseMaterial[1] = g;

m\_DiffuseMaterial[2] = b;

m\_DiffuseMaterial[3] = a;

}

void cmenodel::SetSpecularMaterial (float r, float g, float B, float a)

{

m\_SpecularMaterial[0] = r;

m\_SpecularMaterial[1] = g;

m\_SpecularMaterial[2] = b;

m\_SpecularMaterial[3] = a;

}

Invalid cmenodel::SetShininessMaterial (floating s)

{

m\_ShininessMaterial[0] = s;

}

void CMyModel::SetMaterial()

{

glMaterialfv(GL\_FRONT,gl\_ambientmaterial,m\_AmbientMaterial);

glMaterialfv(GL\_FRONT,gl\_diffusematerial,m\_DiffuseMaterial);

glMaterialfv(GL\_FRONT,GL\_SPECULAR,m\_specialmaterial);

glMaterialfv (gl\_front\_and \_back,GL\_SHININESS,m\_ShininessMaterial);

}

附录B

附录B主要为简易绘制绘图板程序的部分主要代码的实现

1. CMyDraw.h

#pragma once

#include "ggl.h"

typedef struct MYPOS

{

GLfloat x;

GLfloat y;

}MYPOS;

class CMyDraw

{

public:

CMyDraw() {}

~CMyDraw() {};

public:

void Draw\_MouseDown(int hh,int x,int y)

{

m\_MouseBegin.x = x;

m\_MouseBegin.y = hh - y;

m\_MouseEnd.x = m\_MouseBegin.x;

m\_MouseEnd.y = m\_MouseBegin.y;

}

Virtual void Draw\_MouseUp(int hh,int x,int y) = 0;

Virtual void Draw\_MouseMove(int hh,int x,int y) = 0;

public:

MYPOS m\_MouseBegin;

MYPOS m\_MouseEnd;

MYPOS m\_MouseMove;

int hh;

};

class CLine : public CMyDraw

{

public:

CLine() {}

~CLine() {}

public:

void Draw\_MouseUp(int hh,int x,int y);

void Draw\_MouseMove(int hh,int x,int y);

};

class CPoint2 : public CLine

{

public:

CPoint2() {}

~CPoint2() {}

public:

void Draw\_MouseDown(int hh,int x,int y);

void Draw\_MouseUp(int hh,int x,int y);

public:

float x, y;

void setxy (floating x2, floating y2)

{

x = x2;

y = y2;

}

CPoint2 operator &(const CPoint2 & rPoint)

{

x = rPoint.x;

y = rPoint.y;

return \*this;

}

};

class CPencil : public CLine

{

public:

CPencil() {}

~CPencil() {}

public:

void Draw\_MouseMove(int hh,int x,int y);

};

Class loop: public CMyDraw

{

public:

CCircle()

"{}"

~CCircle()

"{}"

public:

void Draw\_MouseUp(int hh,int x,int y);

void Draw\_MouseMove(int hh,int x,int y);

public:

float m\_radius;

};

CRectangle class: public CMyDraw

{

public:

CRectangle() {}

~CRectangle() {}

public:

void Draw\_MouseUp(int hh,int x,int y);

void Draw\_MouseMove(int hh,int x,int y);

};

CTriangle class: public CMyDraw

{

public:

CTriangle() {}

~CTriangle() {}

public:

void Draw\_MouseUp(int hh,int x,int y);

void Draw\_MouseMove(int hh,int x,int y);

};

class CCurve : public CMyDraw

{

public:

CCurve()

{

nNumPoints = 4;

currentNum = 0;

}

~CCurve() {}

public:

int nNumPoints;

int currentNum;

float ctrlPoint[][3];

public:

void Draw\_MouseDown(int hh,int x,int y);

void Draw\_MouseUp(int hh,int x,int y);

void Draw\_MouseMove(int hh,int x,int y);

};

2. CMyDraw.cpp

#include "CMyDraw.h"

Invalid CLine::Draw\_MouseUp(int hh,int x,int y)

{

m\_MouseEnd.x = x;

m\_MouseEnd.y = hh - y;

glDisable(GL\_COLOR\_LOGIC\_OP);

glBegin(GL\_LINES);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glEnd();

glFlush();

}

Void CLine::Draw\_MouseMove(int hh,int x,int y)

{

glEnable(GL\_COLOR\_LOGIC\_OP);

glLogicOp(GL\_XOR);//用异或的方式来画

glBegin(GL\_LINES);//两点确定一条直线

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glEnd();

m\_MouseMove.x = x;

m\_MouseMove.y = y;

m\_MouseEnd.x = m\_MouseMove.x;

m\_MouseEnd.y = hh - m\_MouseMove.y;

glBegin(GL\_LINES);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glEnd();

glFlush();

}

void CCircle::Draw\_MouseUp(int hh,int x,int y)

{

m\_MouseEnd.x = x;

m\_MouseEnd.y = hh - y;

glDisable(GL\_COLOR\_LOGIC\_OP);

m\_radius = sqrt((m\_MouseBegin.x - m\_MouseEnd.x) \* (m\_MouseBegin.x - m\_MouseEnd.x) + (m\_MouseBegin.y) \* (m\_MouseBegin.y - m\_MouseEnd.y) \* 1.0) / 2;

glBegin(GL\_LINE\_LOOP);

for(int i=0; i < 360; i++)

{

glVertex2f((m\_MouseBegin.x + m\_MouseEnd.x) / 2 + m\_radius \* GLfloat(sin(2 \* PI \* I /360))

(m\_MouseBegin.y + m\_MouseEnd.y) / 2 + m\_radius \* GLfloat(cos(2 \* PI \* I /360);

}

glEnd();

glFlush();

}

void CCircle::Draw\_MouseMove(int hh,int x,int y)

{

glEnable(GL\_COLOR\_LOGIC\_OP);

glLogicOp(GL\_XOR);

m\_radius = sqrt((m\_MouseBegin.x - m\_MouseEnd.x) \* (m\_MouseBegin.x - m\_MouseEnd.x) + (m\_MouseBegin.y) \* (m\_MouseBegin.y - m\_MouseEnd.y) \* 1.0) / 2;

glBegin(GL\_LINE\_LOOP);

for(int i=0; i < 360; i++)

{

glVertex2f((m\_MouseBegin.x + m\_MouseEnd.x) / 2 + m\_radius \* GLfloat(sin(2 \* PI \* I /360))

(m\_MouseBegin.y + m\_MouseEnd.y) / 2 + m\_radius \* GLfloat(cos(2 \* PI \* I /360);

}

glEnd();

m\_MouseMove.x = x;

m\_MouseMove.y = y;

m\_MouseEnd.x = m\_MouseMove.x;

m\_MouseEnd.y = hh - m\_MouseMove.y;

m\_radius = sqrt((m\_MouseBegin.x - m\_MouseEnd.x) \* (m\_MouseBegin.x - m\_MouseEnd.x) + (m\_MouseBegin.y) \* (m\_MouseBegin.y - m\_MouseEnd.y) \* 1.0) / 2;

glBegin(GL\_LINE\_LOOP);

for(int i=0; i < 360; i++)

{

glVertex2f((m\_MouseBegin.x + m\_MouseEnd.x) / 2 + m\_radius \* GLfloat(sin(2 \* PI \* I /360))

(m\_MouseBegin.y + m\_MouseEnd.y) / 2 + m\_radius \* GLfloat(cos(2 \* PI \* I /360);

}

glEnd();

glFlush();

}

void CRectangle::Draw\_MouseUp(int hh,int x,int y)

{

m\_MouseEnd.x = x;

m\_MouseEnd.y = hh - y;

glDisable(GL\_COLOR\_LOGIC\_OP);

glBegin(GL\_LINE\_LOOP);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseBegin.x,m\_MouseEnd.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glVertex2i(m\_MouseEnd.x,m\_MouseBegin.y);

glEnd();

glFlush();

}

void CRectangle::Draw\_MouseMove(int hh,int x,int y)

{

glEnable(GL\_COLOR\_LOGIC\_OP);

glLogicOp(GL\_XOR);

glBegin(GL\_LINE\_LOOP);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseBegin.x,m\_MouseEnd.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glVertex2i(m\_MouseEnd.x,m\_MouseBegin.y);

glEnd();

m\_MouseMove.x = x;

m\_MouseMove.y = y;

m\_MouseEnd.x = m\_MouseMove.x;

m\_MouseEnd.y = hh - m\_MouseMove.y;

glBegin(GL\_LINE\_LOOP);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseBegin.x,m\_MouseEnd.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glVertex2i(m\_MouseEnd.x,m\_MouseBegin.y);

glEnd();

glFlush();

}

void CTriangle::Draw\_MouseUp(int hh,int x,int y)

{

m\_MouseEnd.x = x;

m\_MouseEnd.y = hh - y;

glDisable(GL\_COLOR\_LOGIC\_OP);

glBegin(GL\_LINE\_LOOP);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glVertex2i(abs (m\_mousebegin.x-abs (m\_mouseend.x-m\_mousebegin.x)),m\_MouseEnd.y);

glEnd();

glFlush();

}

void CTriangle::Draw\_MouseMove(int hh,int x,int y)

{

glEnable(GL\_COLOR\_LOGIC\_OP);

glLogicOp(GL\_XOR);

glBegin(GL\_LINE\_LOOP);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glVertex2i(abs (m\_mousebegin.x-abs (m\_mouseend.x-m\_mousebegin.x)),m\_MouseEnd.y);

glEnd();

m\_MouseMove.x = x;

m\_MouseMove.y = y;

m\_MouseEnd.x = m\_MouseMove.x;

m\_MouseEnd.y = hh - m\_MouseMove.y;

glBegin(GL\_LINE\_LOOP);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glVertex2i(abs (m\_mousebegin.x-abs (m\_mouseend.x-m\_mousebegin.x)),m\_MouseEnd.y);

glEnd();

glFlush();

}

Invalid CPoint2::Draw\_MouseDown(int hh,int x,int y)

{

m\_MouseBegin.x = x;

m\_MouseBegin.y = hh - y;

glPointSize( 10 );

glBegin(GL\_POINTS);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glEnd();

glFlush();

}

Invalid CPoint2::Draw\_MouseUp(int hh,int x,int y)

{

glBegin(GL\_POINT);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glEnd();

glFlush();

}

Invalid pencil::Draw\_MouseMove(int hh,int x,int y)

{

m\_MouseMove.x = x;

m\_MouseMove.y = hh - y;

m\_MouseBegin.x = m\_MouseEnd.x;

m\_MouseBegin.y = m\_MouseEnd.y;

m\_MouseEnd.x = m\_MouseMove.x;

m\_MouseEnd.y = m\_MouseMove.y;

glPointSize( 1 );

glBegin(GL\_LINES);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glVertex2i(m\_MouseEnd.x,m\_MouseEnd.y);

glEnd();

glFlush();

}

void CCurve::Draw\_MouseDown(int hh,int x,int y)

{

m\_MouseBegin.x = x;

m\_MouseBegin.y = hh - y;

glPointSize( 10 );

glBegin(GL\_POINTS);

glVertex2i(m\_MouseBegin.x,m\_MouseBegin.y);

glEnd();

if(currentNum < nNumPoints)

{

ctrlPoint [current number][0] = m\_MouseBegin.x;

ctrlPoint[currentNum][1] = m\_MouseBegin.y;

ctrlPoint [current number][2] = 0.0f;

currentNum++;

}

if(currentNum == 4)

{

currentNum = 0;

glMap1f(GL\_MAP1\_VERTEX\_3,0.0f,100.0f,3,nNumPoints,& ctrlPoint[0]);

glEnable(GL\_MAP1\_VERTEX\_3);

glBegin(GL\_LINE\_STRIP);

for(int i = 0; i <= 100; i++)

{

glEvalCoord1f((GLfloat)i);

}

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_POINTS);

for(int i = 0; i < 4; i++)

glVertex3f(ctrlPoint[ I ][0] 、ctrlPoint[ I ][1] 、ctrlPoint[ I ][2]);

glEnd();

glColor3f(1.0f, 1.0f, 1.0f);

currentNum = 0;

}

glFlush();

}

附录C

附录C主要为三维模型检索程序的部分主要代码的实现

#pragma region Zernike

-------------------------

Floating point \* CModel\_RetrievalDlg:: select legend (character \* src)

{

float\* z\_src = new float[35];

CModel\_RetrievalDlg \* z\_m = new CModel\_RetrievalDlg();

int index = 0;

for(int i=2;i<=9;i++)

{

// z\_m->grayImg = cvLoadImage(src,1);

if(i%2==0)

{

for(int j=0;j<=i;j+=2)

{

z\_src [index ++] = z\_m->getZernike(src, I, j);

}

}

else

{

for(int j=1;j<=i;j+=2)

{

if(i==9&&j==5)

break;

z\_src [index ++] = z\_m->getZernike(src, I, j);

}

}

}

if(z\_m!=NULL)

delete z\_m;

return z\_src;

}

float \* cmodel\_returevaldlg::normalize\_Zernike(float \* z\_modes)// Zernike归一化

{

float num=0;

float \* z\_image = new float[35];

for(int i=0;i<35;i++){

//cout<<z\_modes[i]<<" ";

num+=z\_modes[i];

}

for(int i=0;i<35;i++){

z\_image[i] = z\_modes[i]/num;

}

return z\_image;

}

Floating CModel\_RetrievalDlg::getZernike(char \* src,int n,int m)

{

int pass = Img2Gray(src);

if(!pass)

return -1;

int depth = 0;

int nChannels = 0;

nChannels = grayImg->nChannels;

depth = grayImg->depth;

switch(depth)

{

Case IPL\_DEPTH\_8U: Caculate\_8\_Zernike(n,m); Break;

Case IPL\_DEPTH\_32F: Caculate\_32\_Zernike(n,m); Break;

default: break;

}

ClearOpenCV();

return Z\_mode;

}

Floating CModel\_RetrievalDlg::get\_8\_XYValue(int x,int y)

{

int height = gray img-> height;

int widthStep = grayImg->widthStep;

char \* Data = grayImg->imageData;

uchar c\_value = ((uchar \*)(data + x \* width step))[y];

Floating point value = (float)c\_value;

return value;

}

Floating CModel\_RetrievalDlg::get\_32\_XYValue(int x,int y)

{

int height = gray img-> height;

int widthStep = grayImg->widthStep;

char \* Data = grayImg->imageData;

Floating point value = ((floating point \*)(data + x \* width step))[y];

return value;

}

int CModel\_RetrievalDlg::Img2Gray(char \* src)

{

If ((grayImg = cvLoadImage(src,1)) ! = 0)

{

return 1;

}

return 0;

}

void CModel\_RetrievalDlg::ClearOpenCV(void)

{

if(oriImg!=NULL){

cvReleaseImage( &oriImg );

oriImg = NULL;

}

if(grayImg!=NULL){

cvReleaseImage( &grayImg );

grayImg = NULL;

}

}

Invalid cmodel\_returevaldlg::Caculate\_8\_Zernike(int n,int m)

{

int height = gray img-> height;

int widthStep = grayImg->widthStep;

Floating N = MinP (height, width);

float N2 = N/2;

float Rpqr\_C =0;

float Rpqr\_S =0;

for(float r=1;r<N2;r++)

{

float temp\_C = 0;

float temp\_S = 0;

for(float s=1;s<=8\*r;s++)

{

float xy\_v = get\_8\_XYValue(r,s);

temp\_C = temp\_C + cos((PI \* m \* s)/(4 \* r))\* xy\_v;

temp\_S = temp\_S + sin((PI \* m \* s)/(4 \* r))\* xy\_v;

}

Floating Rpqr = zernikeR(n,m,(2 \* r)/N);

Rpqr\_C = Rpqr\_C + temp\_C \* Rpqr;

Rpqr\_S = Rpqr\_S + temp\_S \* Rpqr;

}

Cnm = Rpqr\_C \*(2 \* n + 2)/pow(N，2);// Z的实部

Snm = Rpqr\_S \*(2 \* n + 2)/pow(N，2);// Z的虚部

float l\_c = pow(Cnm,2);

float l\_s = pow(Cnm,2);

float l\_p = l\_c + l\_s;

Z\_mode = pow((float)l\_p,(float)0.5);

}

void CModel\_RetrievalDlg::Caculate\_32\_Zernike(int n,int m) //

{

int height = gray img-> height;

int widthStep = grayImg->widthStep;

Floating N = MinP (height, width);

float N2 = N/2;

float Rpqr\_C =0;

float Rpqr\_S =0;

for(float r=1;r<N2;r++)

{

float temp\_C = 0;

float temp\_S = 0;

for(float s=1;s<=8\*r;s++)

{

float xy\_v = get\_32\_XYValue(r,s);

temp\_C + = cos((PI \* m \* s)/(4 \* r))\* xy\_v;

temp\_S + = sin((PI \* m \* s)/(4 \* r))\* xy\_v;

}

Floating Rpqr = getRpqr(n,m,(2 \* r)/N);

Rpqr\_C = Rpqr\_C + temp\_C \* Rpqr;

Rpqr\_S = Rpqr\_S + temp\_S \* Rpqr;

}

Cnm = Rpqr\_C \*(2 \* n + 2)/pow(N,2);

Snm = Rpqr\_S \*(2 \* n + 2)/pow(N,2);

Z\_mode = pow((float)pow(Cnm,2)+ pow(Cnm,2),(float)0.5);

}

Floating-point model\_retrieving dlg::getRpqr (floating-point p, floating-point q, floating-point r)

{

float Rpqr = 0;

float Bppp=1;

int times = (p-q)/2;

int numbers = times+1;

Floating Bpqp = pow((p + q)/(p-q +2), times);

Float \* Bpqk = new float [number];

Bpqk[0] = Bpqp;

int k=(int)p;

int t=(int)p;

For (int I = 0; I <(number -1); I ++)

{

Floating coefficient = ((k + q)\*(k-q)) / ((k + p)\*(p-k +2));

Bpqk[ I + 1] = (-1)\*(Bpqk[ I ])\* coeff;

k=k-2;

}

int temp = numbers-1;

for(k=(int)q;k<=t;k=k+2)

{

Rpqr = Rpqr + (Bpqk[temp])\*(pow(r,k));

temp--;

}

delete[] Bpqk;

float a = Rpqr;

float b = Rpqr;

return Rpqr;

}

Double CModel\_RetrievalDlg:: factorial (long n)

{

if(n < 0)

return(0.0) ;

if(n == 0)

return(1.0) ;

else

return(n \* factorial(n-1)) ;

}

float CModel\_RetrievalDlg::zernikeR(int n,int l,float r)

{

int m ;

float sum = 0.0 ;

if( ((n-l) % 2) != 0 )

{

Coout <<"zernikeR(): incorrect value of n,l \n";

return(0) ;

}

for(m = 0; m <= (n-l)/2; m++)

{

sum + = (pow((float)-1.0,(float)m)) \* (factorial (n-m) ) /

(Factorial (m) \* (factorial ((n - 2 \* m + l) / 2)) \*

(Factorial ((n - 2 \* m - l) / 2) \*

(pow((floating) r,(floating)(n-2 \* m);

}

return(sum) ;

}