三维可视化图形工具OpenGL概述

OpenGL是用于渲染2D、3D矢量图形的跨语言、跨平台的应用程序编程接口（API），基于OpenGL并根据不同实际需要产生了功能众多的可视化程序。大量工程中实际使用的工程辅助软件均使用OpenGL作为图形接口实现可视化功能，包括模型显示、云图渲染、物理仿真等。使用OpenGL来完成这些工作能够方便地在各种平台间移植，也能与C++紧密关联，保证算法的正确和高效。在拓扑优化领域，利用OpenGL图形接口可对ANSYS等软件进行二次开发，拓展现有工业软件在拓扑优化可视化方面的功能，也可按需要自行开发拓扑优化可视化程序，其接口的便捷性使得开发图形可视化程序变得简单。

本文选择在Win32应用程序环境中建立OpenGL应用程序框架进行可视化系统的开发，将三维悬臂梁的等几何拓扑优化结果（即密度值）以图像的方式显示，为研究者对于悬臂梁的拓扑优化提供了帮助，也为等几何拓扑优化的直观表达提供了图形化的理解，具有一定的研究意义。

4.1 引言

本章基于前文已经累积的理论和工作，自主进行基于OpenGL的三维悬臂梁等几何拓扑优化可视化系统的搭建。本系统以Windows应用程序接口（即WinAPI）作为窗体系统框架，配置OpenGL环境以进行图形显示及颜色渲染，基本实现了一般使用可视化系统的功能。

4.2 需求分析

系统需求分析是系统开发的一个重要环节，基于OpenGL的轻量化三维拓扑优化可视化应用可以满足传统应用软件无法满足的灵活性、良好的可移植性以及可拓展性。实际应用方面已经涌现出了大量基于OpenGL的三维可视化应用，许多工业软件中的可视化模块也对OpenGL的功能进行了深入的发掘。通过对OpenGL的理解之后，将其应用于等几何拓扑优化可视化程序中，研发一套基于拓扑优化数据的三维可视化系统。需求分析模块图见下图 所示。

图

4.2.1 功能性分析

由3.1节所述，传统的等几何拓扑优化可视化方式仍使用的是与有限元拓扑优化可视化中相同的方法，显示结果与实际模型几何位置不匹配、无法精确地从可视化结果中得到等几何拓扑优化结果的准确形态。对拓扑优化结果进行三维模型构建时需基于B样条和NURBS等理论，准确对模型进行显示。、

由于本系统的初衷是便于研究人员进行数据结果的直观观察，故所要实现的功能主要包括：拓扑优化结果展示、迭代过程结果展示。拓扑优化结果展示实现三维模型的旋转、拖拽、缩放展示，并通过用户控件界面在程序运行时对可视化结果进行调节；迭代过程结果展示可在程序界面中显示出拓扑优化时每一次迭代得到的结果，并对迭代的整个结果进行快速显示，研究人员可对整个过程进行观察、记录，提高可视化的交互性。

4.2.2 非功能性需求

系统的非功能性需求包括系统可靠性、响应时间等内容，指的是用户使用系统时和使用体验相关的特性。这些需求虽不像功能性需求那样对系统的实际运行产生影响，但它们和用户的使用体验有关。本文根据系统的具体使用场景要求，所需保证的非功能性需求主要有灵活性、可移植性、响应效率等。

（1）灵活性

目前大多数拓扑优化可视化效果都被集成在大型工业软件或是集成开发环境中，仅仅为了对可视化结果进行查看，就需要安装这些大型软件，过程较为繁琐。由于本系统底层所使用的开发语言为C++和OpenGL，因此可以在保证轻量化的同时在绝大部分的计算机上运行。

（2）可移植性

可移植性需求是指在系统的使用场景、使用对象发生变化时，只需少量修改代码即可保证新系统的正常运行。本文所开发的等几何拓扑优化可视化程序在不同细分密度、以及不同形状的结构情况下对程序中的代码进行少量的修改仍可正常使用。

（3）响应效率

当用户使用可视化程序对数据进行显示时，程序的加载效率和相应时间决定了用户的实际使用体验。因此本文采取冗余单元消隐算法对程序可视化效率进行提升，以此来提升用户的使用体验。

4.3 系统设计

4.3.1 总体流程设计

对三维等几何拓扑优化可视化系统完成需求分析后，在此基础之上，对可视化系统的总体运行流程进行设计。系统总体流程可划分为四个部分：前期初始化阶段、数据与状态更新阶段、OpenGL渲染阶段和图形程序退出阶段，分别对应可视化程序设计中的四个主要部分：

（1）InitScene：在该函数中实现整个应用程序的初始化，包括窗体初始化、OpenGL初始化、主要数据初始化等工作，并在该阶段中进行首次NURBS细分曲面的计算，将计算着色器得到的数据与相应的OpenGL缓冲区绑定，在之后的渲染过程中进行数据在屏幕视口的显示。

（2）Update：在渲染步骤前调用，主要进行摄像机的更新以及显示图形形状、状态、显示模式等属性的更新，并根据实际情况确定是否需要重新进行细分曲面计算。

（3）Render：渲染过程，是利用OpenGL进行图形显示所需的关键函数，再整个应用程序正常运行的时间内，都会根据需要不断进行调用，并在调用函数的过程中给出所有关于图形显示的命令。

（4）UninitScene：在图形程序退出时进行调用，断开程序与OpenGL的连接，清理所有C++生成的指针等信息，防止内存泄漏等问题的出现。

在获取等几何拓扑优化后可视化的算法程序框图如图所示。算法程序框图中程序的整体运行被分为上述四个部分，程序在初始化结束后进入Update和Render部分的循环中，直到程序结束并退出。

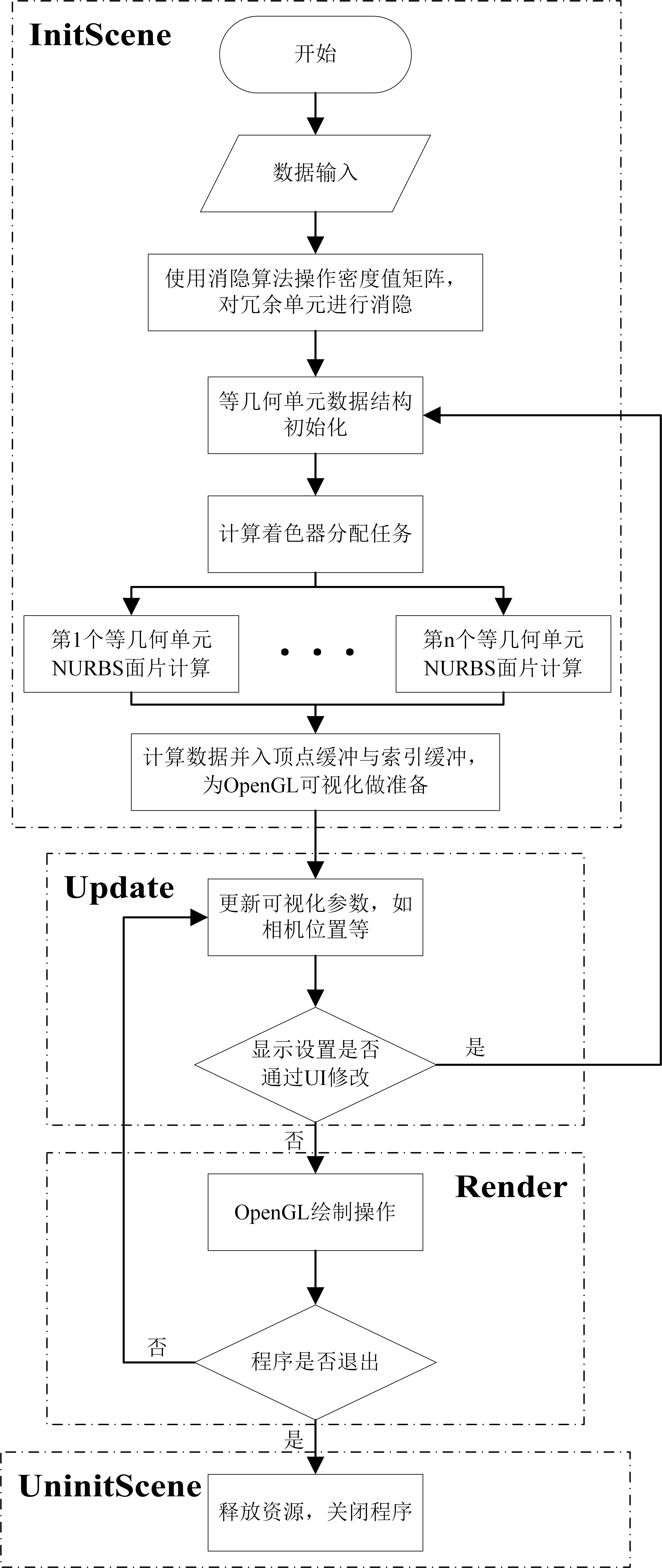


图 拓扑优化可视化程序框图

4.2.2 数据交互设计

本文的数据生成和数据读取均使用二进制文件进行，其中二进制文件相较于字符串形式存储的文件更省空间，在使用C++进行读取时也具有更快的反应速度，以保证数据的实时高效性传输。本可视化系统将数据交互分为三层：

表现层是用户使用的可视化交互界面，实时展示等几何拓扑优化可视化结果。该层向逻辑层发送数据请求，对接收到的数据进行渲染，以实现程序运行时拓扑优化结果的模型信息更新。

逻辑层对表现层发送而来的请求进行响应，并为请求提供相应的接口，然后通过C++代码底层逻辑向数据层请求对应数据，实时进行计算后返回相应的数据。

数据层是进行数据交互的数据来源，拓扑优化得到的数据通过实时计算存储在程序根目录的对应文件夹下。逻辑层通过C++中的二进制读取接口执行与数据层相关的读取操作，并返回对应的数据到逻辑层。

总体数据交互设计示意图如下图 所示

图

数据层中拓扑优化数据的生成与采集主要基于已有的等几何拓扑优化模块实现，逻辑层的二进制读取接口分别读取、处理了数据层内拓扑优化结果中的控制点索引矩阵、权重矩阵、节点向量和单元密度矩阵这四个主要的数据。

4.2.3 功能模块设计

等几何拓扑优化可视化系统采用模块化的开发理念，主要分为曲面细分模块和拓扑优化可视化模块两部分如图 所示。

图

（1）曲面细分

（2）拓扑优化可视化

拓扑优化三维可视化主要指的是用户进入系统后能直观看到的拓扑优化三维模型，该模块核心在于提高用户的交互体验，是本文所主要实现的重点部分。

用户在拓扑优化的三维展示界面可对模型进行缩放、旋转等操作并可通过控件组件对拓扑优化的显示内容的属性和状态进行修改：单元密度阈值、是否网格化显示、是否使用冗余单元消隐算法等。用户通过鼠标点击事件调控控件界面的各项参数，在此过程中系统将用户请求传到逻辑层，逻辑层根据相应的请求修改显示参数，应用到程序运行的Update阶段，进行参数的更新，并将相关数据传递到前端视图层用户界面指定位置。此过程流程图如图 所示

图

在拓扑优化的实践工作中，除了显示迭代得到的最终结果之外，对迭代过程的整体观察也是可视化需要解决的重要问题。对于各个迭代过程进行动态显示能让用户对拓扑优化过程有更好的整体把握，所以将迭代中各个阶段单元密度值的动态显示集成于系统中让用户可实时查看。可通过用户控件界面对动态显示进行操作。

4.3 系统功能实现