y学 校 代 码 10459

学号或申请号

密 级



博 士 学 位 论 文

集体行为

作 者 姓 名： 李明原

导 师 姓 名： 徐明亮

学 科 门 类： 工 科

专 业 名 称： 软件工程

培 养 院 系： 信息工程学院

完 成 时 间： 2018.03

A thesis submitted to

Zhengzhou University

for the degree of doctor

**Research on the Recognition and Simulation of**

**Collective Behavior**

Software Engineering

School of Information Engineering

Mar. 2017

**原创性声明**

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

学位论文作者： 日期： 年 月 日

**学位论文使用授权声明**

本人在导师指导下完成的论文及相关的职务作品，知识产权归属郑州大学。根据郑州大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权郑州大学可以将本学位论文的全部或部分编入有关数据库进行检索，可以采用影印、缩印或者其他复制手段保存论文和汇编本学位论文。本人离校后发表、使用学位论文或与该学位论文直接相关的学术论文或成果时，第一署名单位仍然为郑州大学。保密论文在解密后应遵守此规定。

学位论文作者 日期： 年 月 日

摘 要

集体行为是连续、有序的个体呈现出的宏观行为模式，广泛地存在于细菌菌落、动物群、人群、车流等各种群体系统中。

**关键字：** 集体行为，密度估计

Abstract

Collective behavior refers to macroscopic patterns consisted of continuous and ordered agents, which exists widely in diverse crowd systems, such as bacterial colony, animals group, human crowd and traffic flow.

**Keywords:** Collective Behavior, Density Estimation

目录

[摘 要 I](#_Toc505001624)

[Abstract II](#_Toc505001625)

[目录 III](#_Toc505001626)

[图目录 V](#_Toc505001627)

[表目录 VI](#_Toc505001628)

[第 1 章 绪论 1](#_Toc505001629)

[1.1 Test 1](#_Toc505001630)

[1.2 1](#_Toc505001631)

[1.3 研究现状 1](#_Toc505001632)

[1.3.1 交叉学科中的集体行为研究 1](#_Toc505001633)

[1.4 研究内容及意义 2](#_Toc505001634)

[1.5 论文章节安排 2](#_Toc505001635)

[第 2 章 基于多视角交互式建模方法 3](#_Toc505001636)

[2.1 引言 3](#_Toc505001637)

[2.2 背景知识 3](#_Toc505001638)

[2.3 本章小结 3](#_Toc505001639)

[第 3 章 基于图像的机械模型运动优化方法 4](#_Toc505001640)

[第 4 章 基于关节优化的机械模型装箱方法 5](#_Toc505001641)

[4.1 引言 5](#_Toc505001642)

[4.2 本章小结 5](#_Toc505001643)

[第 5 章 实时精确的三角网格半规则化方法 6](#_Toc505001644)

[5.1 引言 6](#_Toc505001645)

[5.2 背景知识 6](#_Toc505001646)

[5.2.1 半规则网格与多分辨率分析 6](#_Toc505001647)

[5.2.2 网格半规则化算法综述 6](#_Toc505001648)

[5.2.3 网格半规则化相关工作 6](#_Toc505001649)

[5.3 基于层次的网格半规则化算法概述 6](#_Toc505001650)

[5.4 基于层次的网格并行简化 6](#_Toc505001651)

[5.4.1 网格分层并行简化 6](#_Toc505001652)

[5.4.2 网格表面参数化 6](#_Toc505001653)

[5.5 6](#_Toc505001654)

[第 6 章 总结和展望 6](#_Toc505001655)

[6.1 工作总结 6](#_Toc505001656)

[6.2 下一步工作 6](#_Toc505001657)

[参考文献 7](#_Toc505001658)

图目录

[图 1.1 自然界中的集体行为。图中场景依次是细菌菌落、蚂蚁群、鱼群、鸟群、斑马群、人群、车流。 1](#_Toc505957934)

[图 5.1半规则网格多分辨率分析。从左到右是高分辨率网格通过简化得到低分辨率网格的过程，反之，从右向左是低分辨率网格通过细分得到高分辨率网格的过程。 7](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957935)

[图 5.2网格半规则化算法流程图。该算法主要分为两步：第一，简化网格得到基网格；第二，细分网格并重采样，得到半规则网格。 9](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957936)

[图 5.3基于层次的网格半规则化算法流程图。输入网格通过15层迭代简化得到最粗糙网格，将作为基网格进行两次细分，得到目标半规则网格。 11](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957937)

[图 5.4网格二次误差度量简化过程半边图。(a)待简化的网格。(b)待简化的边在简化时的独立区域。(c)简化后并重新网格化得到的网格。 13](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957938)

[图 5.5独立域顶点投影平面并重新网格化图。红色点代表简化后新顶点，以该顶点为轴将区域分割为两个环形区域，将两个环形区域的顶点分别投影到各自平面上，删除原顶点的连接关系（图中虚线表示），重新三角网格化（图中实线表示），最终被简化的顶点投影在新生成的三角网格上。 16](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957939)

[图 5.6顶点参数化图。层网格向层网格简化，图中蓝色顶点是层网格简化时删除的顶点，将其投影在层网格的三角面片上。图中红色点是比层更高层简化时被删除的顶点，首先将顶点投影再平在平面上，然后再将其参数化。 18](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957940)

[图 5.7中点1：4细分方法图。图中黑色线是初始网格，三个初始网格三角形经过两层细分，每层在各边中点位置添加一个顶点，三个顶点相互连接，得到下一层网格。红色网格是细分一层后的网格，绿色网格是细分两层后的网格。 20](file:///E:\毕业\PHD_thesis\博士论文.docx#_Toc505957941)

表目录

[表 5.1网格分层并行简化算法 15](#_Toc505957942)

[算法5.1：网格分层并行简化算法 15](#_Toc505957943)

# 绪论

* 1. Test

集体行为作为自然界中最普遍的现象之一，广泛地存在于各种群体系统中。图1.1展示了典型的集体行为场景，包括细菌菌落、动物群、人群、车流等。



图 1.1 自然界中的集体行为。图中场景依次是细菌菌落、蚂蚁群、鱼群、鸟群、斑马群、人群、车流。

* 1. 研究现状
     1. 交叉学科中的集体行为研究

集体行为显著地存在于各种群体系统中，吸引了来自不同科研领域研究者的广泛关注。已有研究在集体行为的产生机理、演化过程、运动建模等方面都取得了阶段性成果。

* 1. 研究内容及意义
  2. 论文章节安排

# 基于多视角交互式建模方法

* 1. 引言

集体行为是最普遍的自然现象之一，

。

* 1. 背景知识

提出的

* 1. 本章小结

# 基于图像的机械模型运动优化方法

# 基于关节优化的机械模型装箱方法

* 1. 引言
  2. 本章小结

# 实时精确的半规则网格化方法

* 1. 引言

近年来，随着计算机辅助设计和计算机图形学方向的迅猛发展，三维模型表面数字化表达已经成为该领域内的热点问题。模型曲面表达的发展，促进了模型制造、数字娱乐、增强现实、虚拟现实、医学图像处理、计算机模拟等众多领域。目前最流行的表面表达方式是使用多边形网格表达三维模型表面几何结构。基于结构简单、扩展性强、便于主流图形硬件设备计算等情况，四边形网格和三角形网格是学术界最普遍的作为多边形网格来表达三维模型表面的两种结构。

随着三维建模越来越自动化、智能化和便捷化，三维模型也来到井喷式发展的时代，所以快速优化网格结构是计算机图形学内亟待解决的重要问题。优化网格结构主要目标是为了加快网格处理和增大网格压缩率。快速网格处理可以提高大规模场景渲染速度、复杂建模速度等。而网格压缩率增加能加快网格传输速度，降低带宽。半规则网格具有极好的可扩展性和可压缩性的特点，能够很好的解决上述网格处理遇到的难题。相比较非规则网格，半规则网格具有层次结构和规律结构，便于实现多分辨率网格分析，网格半规则化在文献[[1](#_ENREF_1)]小波信号处理中首先被关注，之后被广泛应用于建模、模型分析、表面渲染。半规则网格最大的优点是网格数据可以多分辨率表达，这个优点被广泛应用在多个研究方向中。Certain [[2](#_ENREF_2)]利用半规则网格的优点，进行可视化和渲染的时候可以在各级分辨率下高效的切换。Roudet[[3](#_ENREF_3)]利用小波分析对半规则网格进行网格分割。文献[[4-8](#_ENREF_4)]都是利用半规则网格多分辨率的优点，对网格进行高压缩比的压缩。

目前通过建模软件或者三维扫描仪得到的模型网格均为非规则网格，所以需要对模型重新网格化得到半规则网格。本文主要研究课题为交互式三维模型的建模研究，为了不影响用户体验，交互式建模要求模型网格可以实时生成、编辑以及优化。所以本章提出的三角网格半规则化方法对算法时间要求极高，需要算法达到毫秒级别。主流的半规则化三角网格有两个主要步骤：简化网格和精细化网格。为了达到算法效率要求，我们算法需要并行化这两个主要步骤。精细化网格算法具有局部独立性，基本可以直接并行化。网格简化本章采用Garland[[9](#_ENREF_9)]提出的二次误差测度的网格简化方法，该方法优点是在误差度量标准表现突出，但缺点是无法直接并行化该方法。本章提出了一种实时精确的三角网格半规则化方法，致力于实时得到精确的半规则网格。

本章工作主要贡献点总结如下：

1. 提出了一种快速并行的三角网格半规则化算法，该算法处理消费级网格能达到实时处理速度，具体为处理百万个三角面片的网格运算时间为几十毫秒级别。
2. 提出了一种基于二次误差测度的分层并行网格简化算法，通过分层网格简化，每层寻找并行最大独立集，独立集间可以并行化操作。
3. 提出一种新颖的基于二次误差测度简化的网格参数化算法，将精细网格顶点信息存储在粗糙网格表面上。
4. 基于算法中的参数化信息，提出一种并行的鲁棒的精细化网格算法，从而得到半规则化网格。
   1. 背景知识
      1. 半规则网格与多分辨率分析



图 5.1半规则网格多分辨率分析。从左到右是高分辨率网格通过简化得到低分辨率网格的过程，反之，从右向左是低分辨率网格通过细分得到高分辨率网格的过程。

网格是否为规则网格是根据顶点的连接关系定义的，对于一个三角形网格，每个顶点的度为6，这个网格即为规则网格（regular mesh），非规则网格即为顶点的度不为6的三角形网格。半规则网格（semi-regular mesh）是由顶点度为6的规则网格和顶点度不为6的非规则网格组成。半规则网格有着特殊的结构，可以将它规则网格部分以四个三角形合并为一个三角形的方式，简化为一个非规则网格（如图5.1所示），我们将简化后的网格称为低分辨率网格。相反，我们也可以对非规则网格中每一个三角形，以一个三角形细分为四个三角形的方式，得到半规则网格。

半规则网格几何拓扑结构非常适合基于小波的多分辨率分析（如图5.1所示）。通过对低分辨率网格的细分或者对高分辨率网格的简化，可以得到不同分辨率的半规则网格。对于高分辨率网格进行基于小波的多分辨率分析，通过低频过滤器可以得到低分辨率网格，通过高频过滤器可以得到网格的细节信息，高频细节信息保存了相邻两个分辨率网格的几何结构差异。通过这种多分辨率分析方法，网格可以看作是由低分辨率网格和一些细节信息构成。基于小波理论，低分辨率网格代表低频信号，一个细节信息代表一个频带。所以一个网格通过细分策略，可以得到高分辨率网格，其细节信息由高频信号提供。

基于上述分析，可以得到半规则网格的两个主要特性。其一，半规则网格的拓扑连接关系保存在低分辨率网格中，高频的网格拓扑关系可以计算得到，不需要保存。其二，半规则网格由于其多层结构可以进行多分辨率分析，方便得到多分辨率网格。

* + 1. 网格半规则化算法综述

本章提出的半规则网格化算法是将输入的非规则三角网格转换成半规则三角网格的方法，得到的半规则网格近似于输入的非规则网格。Payan 等人将[[10](#_ENREF_10)]主流的网格半规则化算法概括为两步（如图5.2所示）。首先，简化网格得到基网格。基网格是由非规则网格不断简化得到的低分辨率网格，它近似输入网格。在简化过程中，不断参数化因简化所丢失的顶点信息，将这些参数化信息保存在基网格表面上。其次，以三角网格一分四策略（如图5.1红色三角形），迭代细分基网格。每次迭代，根据基网格表面存储的参数化信息，重新计算新细分出的顶点空间位置，得到高分辨率半规则网格。

半规则网格化算法关键核心在于如何在简化网格的同时如何参数化网格。网格参数化方法很多，全局网格参数化方法（例如文献[[11](#_ENREF_11), [12](#_ENREF_12)]）都可以应用到半规则网格化算法，但是时间效率比较差。本文是要求实时生成半规则网格，考虑到算法时间效率，本文参考了文献[[13](#_ENREF_13), [14](#_ENREF_14)]，基于二次误差测度简化方法，采用局部参数化方法，将半规则网格顶点映射到非规则网格的表面。该方法很容易并行化，并且具有很好的运行效率。

* + 1. 网格半规则化相关工作

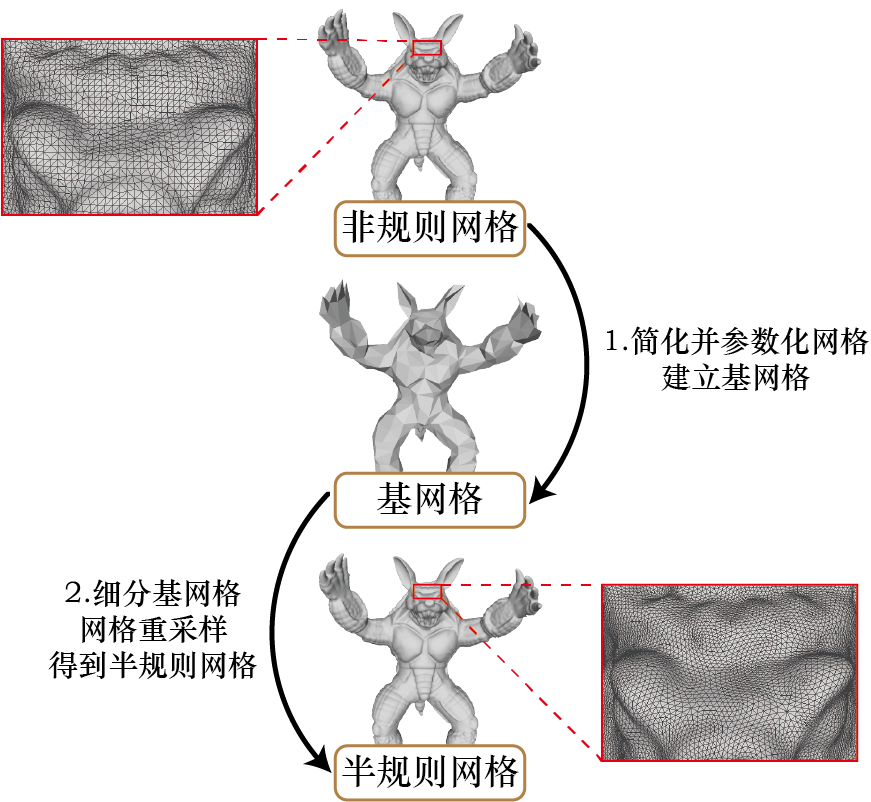


图 5.2网格半规则化算法流程图。该算法主要分为两步：第一，简化网格得到基网格；第二，细分网格并重采样，得到半规则网格。

对于半规则网格的研究最早出现在十九世纪中期。Lounsbery等人[[14](#_ENREF_14), [15](#_ENREF_15)]首先对于表面网格提出了多分辨率分析，通过简单的基网格和线性的细分策略，对半规则网格进行小波分析。也是利用小波变换，将非规则网格转变为半规则网格。之后主要研究关注于怎样建立基网格和建立基网格和非规则网格的对应关系。文献[[14](#_ENREF_14), [16](#_ENREF_16), [17](#_ENREF_17)]致力于处理复杂的网格布线情况、有边界的情况和复杂的几何表面情况。文献[[18](#_ENREF_18), [19](#_ENREF_19)]通过局部参数化或局部投影，半规则网格顶点对应到非规则网格表面。二十一世纪前的研究都是分块参数化的，缺乏对参数化平滑，之后的大部分研究（例如文献[[20-23](#_ENREF_20)]）主要在平滑参数化得到低畸变的参数信息。最近几年的研究主要致力于得到高精度的半规则网格，仅仅对参数化进行平滑是不够的，需要得到高精度的采样信息，文献[[24-26](#_ENREF_24)]采取的方法是在非规则网格上直接重新采样。本章网格半规则化算法应用在交互系统中，需要一种自适应实时的半规则化算法，所以本章算法思路类似[[14](#_ENREF_14)]，局部的参数化有利于算法的并行化，从而大幅度提高算法效率。

并行简化网格也是计算机图形学方向的热点研究问题，同时也是网格半规则化重要步骤。边界冲突是并行简化网格的主要问题，解决这一问题主要有两种方法：顶点删除法和边折叠法。顶点删除法（例如文献[[27](#_ENREF_27)]）是选择独立顶点，删除这些顶点，通过重新三角网格化来填补因删除顶点产生的网格空洞，迭代上述操作，直到得到满意的简化网格。为了提高每次迭代删除顶点的数量，Rossignac等人[[28](#_ENREF_28)]提出的算法基于几何最近邻提取顶点簇。Lindstrom等人[[29](#_ENREF_29)]使用Rossignac的提取顶点簇的方法，简化超出内存容量的大型网格模型。Schaefer等人[[30](#_ENREF_30)]使用八叉树整理网格拓扑结构，从而自适应得到顶点簇。边折叠法是迭代的将一条边缩减为一个顶点，通过评估算法确定该顶点位置，并重新网格三角化。早期，Guéziec等人[[31](#_ENREF_31)]提出基于边折叠的网格简化算法，主要致力于寻找一种挑选待折叠的边的策略。Garland等人[[9](#_ENREF_9)]提出了一个简单的二次误差度量，有效地测量了顶点变化引起的表面畸变，并提出了一种基于度量的高效简化算法。为了提高简化效率，许多简化方法寻求并行算法。文献[[32](#_ENREF_32), [33](#_ENREF_33)]先将网格分割为多个子网格，然后再进行简化，这种简化算法并行粒度很难控制，无法应用在GPU上。Tatarchuk 等人[[34](#_ENREF_34)]对网格顶点建立八叉树结构，利用该结构对顶点聚簇，对多个簇进行并行简化。并行的粒度取决于八叉树的分辨率，虽然效率很高，但灵活性不足。Shontz等人[[35](#_ENREF_35)]提出了一种混合算法，将分割网格和简化网格分别放在CPU和GPU上处理。文献[[36](#_ENREF_36), [37](#_ENREF_37)]通过建立顶点的独立集得到简化时的独立区域，在并行简化时，每个独立区域之间不会影响。本章采用的简化算法是基于层次的迭代简化，与文献[[36](#_ENREF_36), [37](#_ENREF_37)]类似，不过本章每层建立被选中的边的最大独立集，基于二次误差度量简化方法[[9](#_ENREF_9)]，简化每一条选中的边，每个独立区域都是独立的，独立区域之间可以并行简化操作，迭代上述操作，直到得到满意的网格。

* 1. 基于层次的网格半规则化算法概述

首先用数学符号描述网格结构，方便本章节算法阐述。一个网格由拓扑信息和几何信息构成，用表示，其中代表个顶点的坐标，其中包含了网格所有的拓扑结构，具体表示为三种类型：顶点，边，面 。如果，则顶点和相邻。顶点周围所有相邻顶点集合表示为，顶点周围所有相邻面集合表示为，顶点周围所有相邻边集合表示为。

主流的网格半规则化算法概括为两步，简化网格和细分网格（如图5.2所示）。本章简化网格策略采取的是分层简化（如图5.3所示），算法输入的初始网格为非规则网格表示为，非规则网格通过每层简化得到每层对应的近似网格，其中，简化直到得到基网格。每层简化方法是基于二次误差度量简化方法[[9](#_ENREF_9)]，为了使算法可并行化，我们首先需要寻找网格独立的边，这些边组成该层网格最大独立集，选中的独立的边压缩成点的所影响到的网格拓扑结构称为独立区域，所谓边独立，就是在每层简化时，任意两个独立区域相互不干扰。简化后，层网格中的两个顶点由层网格中的一个顶点所代替，之后重新三角化该独立区域，并将层网格中简化掉的顶点用参数化的方式投影到层网格中三角面片上，用来保存该顶点信息。

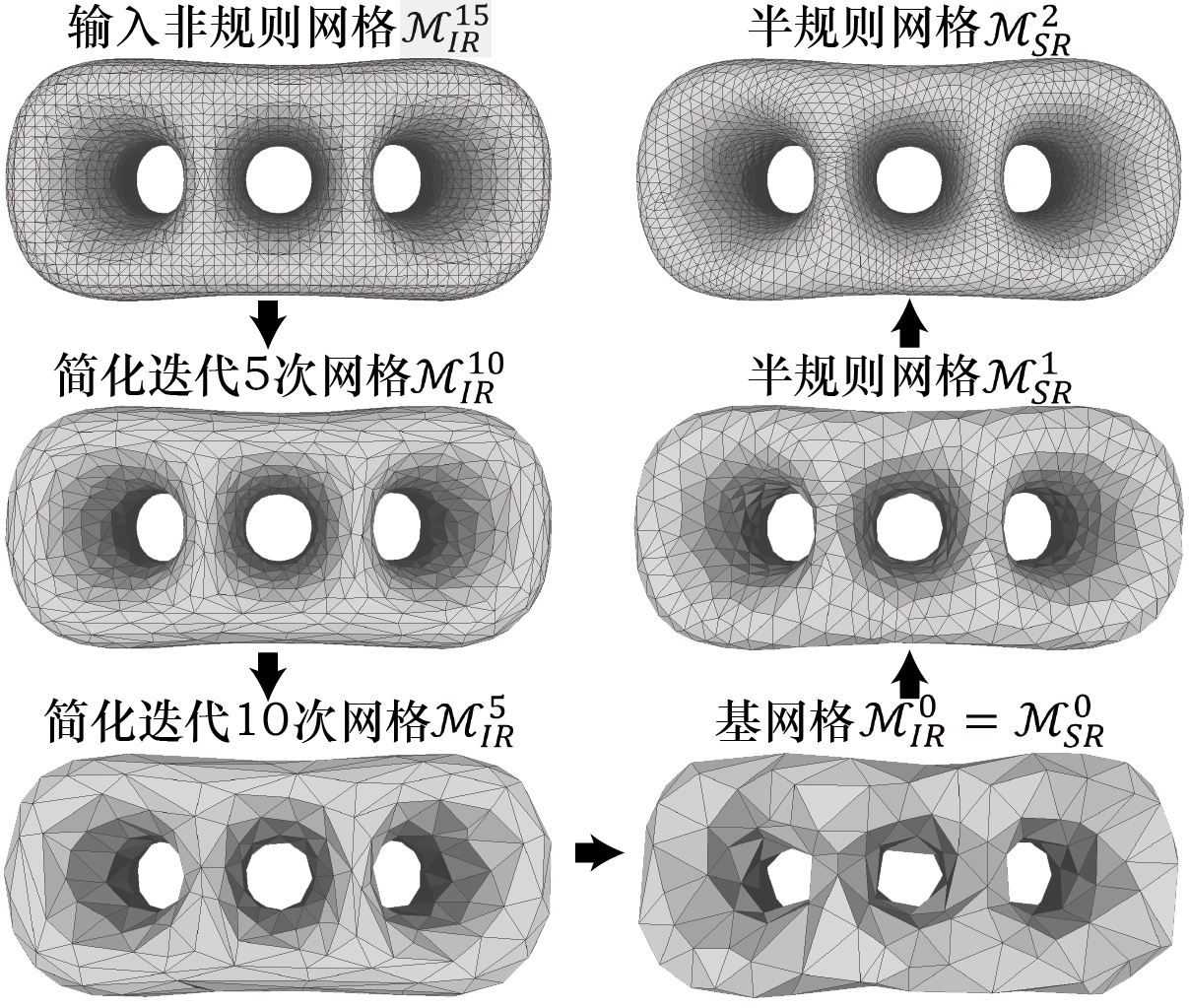


图 5.3基于层次的网格半规则化算法流程图。输入网格通过15层迭代简化得到最粗糙网格，将作为基网格进行两次细分，得到目标半规则网格。

细分网格也采用分层细化，对网格中的三角形以一分四的方式并行细分，再通过保存在基网格表面的参数化信息优化细分出来的新顶点坐标，得到下一层网格。值得注意的是，该细分方法可以直接并行操作，并且为了使网格空间坐标分布均匀，本章采用Loop策略平滑参数化信息。通过多层细分，最终得到满足要求的半规则精细网格。

* 1. 基于层次的网格并行简化

本节提出一种基于二次误差度量简化方法的分层并行网格简化方法，并参数化上层网格损失的顶点信息，将信息保存在该层网格表面，最终得到基网格，该方法可以减小因简化而网格表面的畸变，同时因为并行操作，可以大幅度提高算法效率，从而达到算法的实时性要求。

* + 1. 网格分层并行简化

本章采用的网格简化过程是分层简化，每层简化分为两步（如图5.4所示）：第一，选取若干待简化边，采用二次误差度量简化方法简化，每条边设定一个独立区域，多个独立区域之间可以并行简化。第二，采用德洛内三角化（Delaunay triangulation）算法对独立区域重新三角化。如图5.4为例，该简化过程中独立区域内减少了一个顶点和三个边。

网格拓扑信息存储的数据结构决定了网格的表达和计算方式，适合的数据结构可以提高网格的运算效率。本章对网格的优化算法绝大多数都是对三角网格局部连续访问，所以采用Openmesh[[38](#_ENREF_38)]中的半边（half-edge）网格结构作为拓扑信息存储的数据结构。半边结构是由三元组构成：顶点（vertex）、半边（half-edge）、面（face），这样存储结构的优势可以迭代快速访问每个顶点相邻的顶点集，同时也可得到相邻的边和面的信息。

二次误差度量简化算法是常用的网格简化算法，简化过程后，可以得到高质量与原网格近似的精简网格。简化过程是基于二次误差度量法，选择误差最小的边，将该边收缩为一个顶点，并优化顶点位置。我们定义网格中顶点的二次误差是该顶点到相邻三角面片表面的二次距离总和，计算如下：

|  |  |  |
| --- | --- | --- |
|  |  | (5.1) |

其中表示平面参数方程中的参数。定义基本二次误差矩阵为：

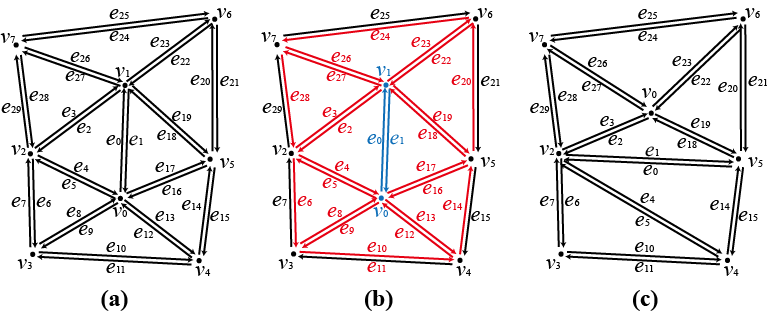
|  |  |  |
| --- | --- | --- |
|  |  | (5.2) |

将公式(5.2)带入公式(5.1)可得到：

|  |  |  |
| --- | --- | --- |
|  |  | (5.3) |

通过公式(5.3)可以看出可以表达任意点到平面的二次距离。通过累加基本二次误差矩阵，我们可以用单个矩阵表示一个平面集合。一条边的误差矩阵可以表示为两个顶点的误差矩阵的累加。

图 5.4网格二次误差度量简化过程半边图。(a)待简化的网格。(b)待简化的边在简化时的独立区域。(c)简化后并重新网格化得到的网格。



如何确定收缩边后得到的新顶点坐标？如何挑选待简化的边，使简化后网格与原网格误差最小？这两个问题是线性简化的核心问题，如何定义误差和怎样优化减小误差是解决问题的关键。首先我们定义顶点误差二次型为，由于为的对称矩阵，将顶点表示为。将一条边收缩为一个顶点，用近似表达顶点的误差，误差矩阵用待简化的边两端顶点误差矩阵累加表示。利用简单的策略，将新顶点坐标设置为或者，不过这样的精度无法得到满意的结果。为了得到更精确的简化网格，我们需要最小化顶点误差。因为误差函数是二次方程，所以这是一个线性优化问题。在梯度为零，即,这一位置，顶点误差最小。如果矩阵为可逆矩阵，可以求得顶点坐标为：

|  |  |  |
| --- | --- | --- |
|  |  | (5.4) |

若矩阵为不可逆矩阵，则采用简单策略将新顶点坐标设为。将新顶点坐标带入误差函数中，我们即可以得到每条边被简化的代价值。

根据采用Openmesh[[38](#_ENREF_38)]中的半边（half-edge）网格结构，网格拓扑结构由三元组点、半边、面组成。一个三角面片包含一个面，三个半边，三个顶点；一个半边包含两个半边。每个顶点存储以该顶点作为起点的半边，每个半边存储信息为：该半边的起始顶点；该半边所属的三角形内部，以该半边终点为起始顶点的半边；该半边所属的面。每个面存储包含的三条半边中任意一条。每条半边和它的反向半边组成一条完整的边，为了节省空间和提高访问速度，两条半边存储时相邻存储，不需要额外开辟空间进行存储。

为了使简化算法能够并行，我们采用了分层的简化方法。算法在层与层之间，是线性运行，在每一层上，可同时选中多个边进行简化，所以在层内，算法是并行运行的。为了做到层内边简化并行，首先定义边简化独立区域，独立区域作用是在独立区域内部简化一条边时，不会更改区域外部的网格信息，所以独立区域之间可以做到并行计算。

为了找到适合二次误差度量简化算法的独立区域，我们需要知道在简化时，哪些网格信息被修改。将一条边简化为一个顶点，简化后，该边两个顶点的一周相邻三角形（如图5.4-(b)红色区域）包含的顶点和半边的信息都需要修改，所以该区域为一个独立区域。如图5.4-(b)中，蓝色的边将简化为顶点，为了使内存重用和访问连续，新顶点信息保存在原顶点内存中，新顶点坐标采用上述二次误差度量简化算法得到。得到新顶点后对独立区域重新三角化，得到新网格拓扑结构，所以顶点和的临边也会被新生成的半边所覆盖。独立区域内最外侧的半边（如等），其结构信息中下一条半边信息会改变，所以也在独立区域内部。

在每层简化过程中，独立区域之间可以并行做简化，为了达到最大的并行粒度，需要寻找最大独立集，即每层简化时可以并发执行个数最多的独立区域。根据二次误差度量的简化原则，先选取误差最小的边进行简化，所以算法首先需要计算当前网格中每条边的误差值，然后根据误差值由小到大对边进行排序放入待简化边队列。每次从队列中选择误差最小的边作为待简化的边，边被选中后，将该边所在的独立区域内的边从队列中删除。重复上述操作，直到队列为空，则得到最大并行简化的独立集。

本章介绍的算法是网格分层并行简化算，该算法既保证简化后的网格有较小误差，同时可以高粒度的并行计算，从而达到实时的运行速度，该算法概括为表5.1所示。

表 5.1网格分层并行简化算法

|  |  |
| --- | --- |
| 算法5.2：网格分层并行简化算法 | |
| 输入：初始网格，被简化次数*L* | |
| 输出：基网格 | |
| 1: | **for** **do** |
| 2: | 对每个顶点计算误差矩阵 |
| 3: | 计算网格中每条边若被压缩所生成的新顶点坐标 |
| 4: | 计算每条边的误差值 |
| 5: | 根据误差值从小到大排序边，将边放入队列 |
| 6: | **while** 不为空 **do** |
| 7: | **。。** |
| 8: | 将插入待简化独立集，并将边所在独立集内的边从中删除 |
| 9: | **end while** |
| 10: | 并行删除独立集中的每条边，生成新顶点， |
| 11: | 重新三角化独立区域，得到该层简化后的网格 |
| 12: | **end for** |

* + 1. 网格表面参数化

分层简化可以将输入的不规则网格简化成基网格，本节介绍如何将简化掉的网格信息保存在基网格上。参数化表面网格是将网格表面信息存储在网格上，在每层简化过程中做参数化，最终能使基网格累加参数化信息得到输入网格的信息。

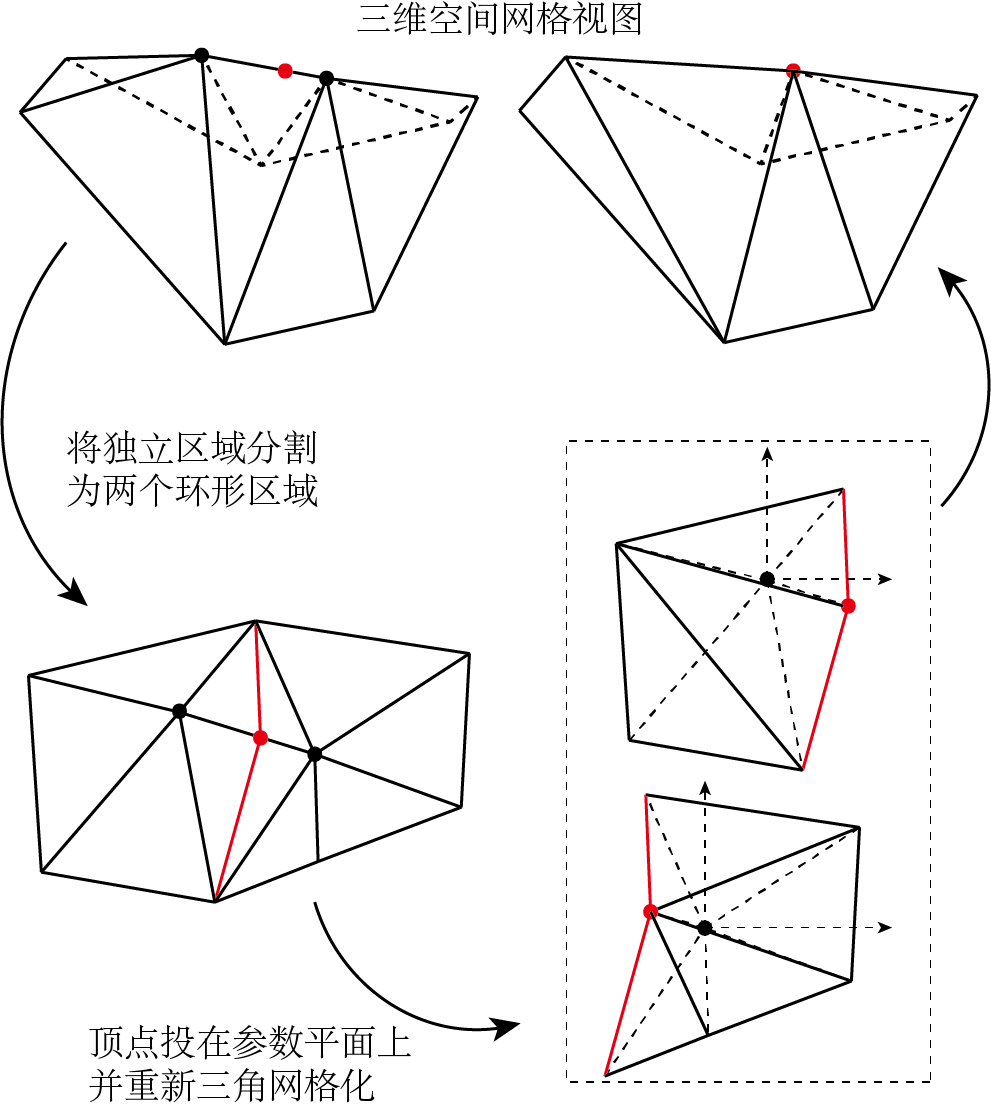
为了得到网格的拓扑结构，5.4.1已经介绍了网格简化的过程，将网格分割为若干个独立域并行简化，下面介绍在每个独立域内如何参数化和如何重新三角网格化。为了参数化每个独立域内的网格信息，我们需要找到一个平面，将独立域内的顶点投影到平面上。

图 5.5独立域顶点投影平面并重新网格化图。红色点代表简化后新顶点，以该顶点为轴将区域分割为两个环形区域，将两个环形区域的顶点分别投影到各自平面上，删除原顶点的连接关系（图中虚线表示），重新三角网格化（图中实线表示），最终被简化的顶点投影在新生成的三角网格上。

独立域由一条边和它们环形邻域顶点、构成。我们使用文献[[39](#_ENREF_39)]保角映射的方法，将环形邻域的顶点投影到平面上。为了在顶点投影时得到最小畸变，需要将每个独立域分割为两个环形邻域。分割方法如图5.5所示，图中红色顶点代表独立区域内简化后生成的新顶点，环形区域1、2分别包含顶点、，并将新顶点插入环形顶点内，新顶点拓扑结构直接与顶点相连接。则简化边的两个环形邻域分别由，构成。最终算法将被删除顶点的环形邻域顶点投影到平面上，通过重新三角化多边形，得到区域内新的网格拓扑结构。

将环形邻域顶点用保角映射方法投影到平面上，用函数表示。首先环形邻域内的顶点作为平面的中心坐标。该定点由个顶点组成的环形邻域，其中三角形属于中的拓扑结构，因为是环形首个顶点也是末尾顶点，即。环形邻域中的顶点投影到平面的坐标表示为：

|  |  |  |
| --- | --- | --- |
|  |  | (5.5) |

其中，

|  |  |  |
| --- | --- | --- |
|  |  | (5.6) |

并且。使用保角映射方法好处在于：首先这种方法使用于任何网格结构；并且这样方便计算，能够使算法达到实时的效果；而且可以最小化因投影产生的畸变；最后是这种映射方法是双向映射，不会将两个三角形投影到同一个区域而产生网格重叠。映射后，环形邻域的顶点被投影到同一个平面，我们可以采用固定边界的Delaunay三角化（CDT）重新三角化环形区域，得到每个独立区域简化后的网格拓扑结构，从而得到整个网格简化后的网格拓扑结构。

上述介绍了将简化后删除的顶点投影到平面上，接下来将删除的顶点参数化在简化后的网格平面上，我们首先定义映射函数：

|  |  |  |
| --- | --- | --- |
|  |  | (5.7) |
|  |  | (5.8) |

其中函数是一个双向映射函数，可以将初始网格上的顶点投影到基网格表面，也可以将基网格上的顶点投影到初始网格表面。将一顶点通过映射函数投影在基网格上一点，该顶点参数化方程为：

|  |  |  |
| --- | --- | --- |
|  |  | (5.9) |

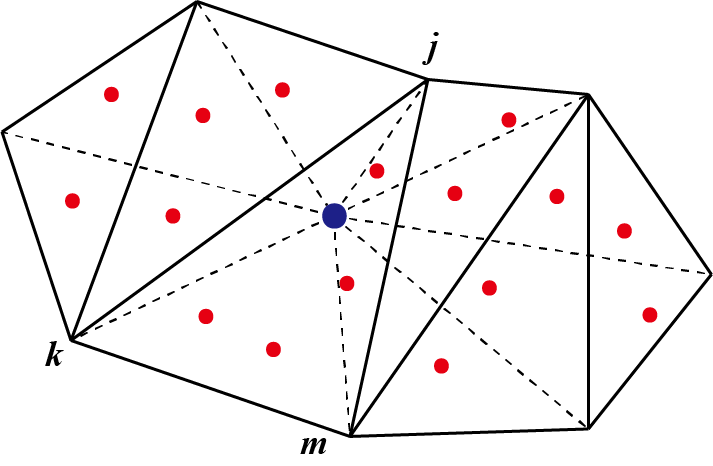
其中是基网格中的一个三角面片，是重心坐标，所以。

图 5.6顶点参数化图。层网格向层网格简化，图中蓝色顶点是层网格简化时删除的顶点，将其投影在层网格的三角面片上。图中红色点是比层更高层简化时被删除的顶点，首先将顶点投影再平在平面上，然后再将其参数化。

本文采用的是层次的网格简化方法，所以需要建立层次的参数化投影方法。本文使用投影函数表示顶点和顶点间的投影关系，则层次投影函数由开始，直至0层结束。假设已知*l*层投影函数，我们需要求出层投影函数，这样递归计算下去，最终可以求出0层投影函数。本文的投影函数仅需要计算中的顶点在*l*层所属三角面片的参数化重心坐标，所以在层次简化过程后，*L*层中的每个顶点会有三种可能：

1. ：在层网格简化过程中，顶点没有被删除，仍然保留在层网格表面。这种情况不需要对该顶点做参数化，投影过后不改变顶点坐标，。
2. ：层网格向层网格简化后，顶点被删除。这种情况需要根据上述介绍的保角映射方法（如图5.5所示），将该顶点的环形邻域顶点投影在一个平面上，通过重新三角化环形邻域，被删除的顶点投影在新生成的三角面片上得到平面上坐标，用重心坐标参数化，如公式。所以投影公式表示为。
3. ：顶点在比层更高层简化时被删除，这种情况投影公式表示为，顶点投影所在的三角面片为。如果，也就是说在层网格简化时的三个顶点均未被删除掉，则顶点参数化信息不需要做改变，否则顶点（图5.6红色顶点）将被投影到新的三角形上面重新做参数化。如图5.6所示，顶点被重新投影在新的平面上，其平面坐标为。重新三角网格化后，所在的三角面片为，其重心坐标为，则投影函数表示为。

算法在每层简化时都需要遍历输入网格的所有顶点，每个顶点根据它的类型做以上操作，所以本算法的线性时间复杂度为。参数化算法在每个独立域内都是独立的，所以在每层简化过程中，参数化过程也是可以并行计算的，从而大大提高算法效率。

* 1. 网格细分重建

本章将简化后的基网格通过多层细分得到半规则网格，每层细分改变网格的拓扑结构，再根据参数化信息确定每个顶点三维坐标。本章首先介绍一种简单的基于参数化信息的网格细分，再对参数化信息做平滑，得到优化后的半规则网格。

* + 1. 基于参数化信息的网格细分

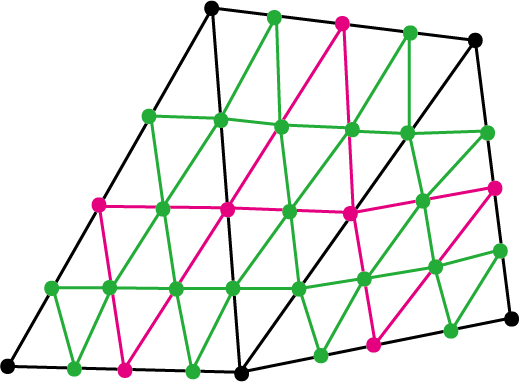
因为是一个双向投影函数，所以可以将细分出的顶点根据反向投影函数投影到初始网格上。参考文献[[16](#_ENREF_16)]的工作，对基网格上的三角形1：4的方法细分（如图5.7所示），图中可以看出，新增的顶点是度都为6的顶点，细分出来的结构为规则网格结构，这样细分后可以得到半规则网格的拓扑结构。首先定义层次的规则化网格，其中代表细分出的新顶点集合，代表网格的拓扑结构，上标代表该网格经过层中点细分，该细分的初始网格是基网格，则。中点细分得到的结果是，细分网格所有顶点在基网格表面，即，并且基网格的拓扑结构仍然保存在细分网格内，新增加的拓扑结构上的顶点的度都为6，属于规则网格。所以将基网格进行层细分后所得到的网格可以表示为。

图 5.7中点1：4细分方法图。图中黑色线是初始网格，三个初始网格三角形经过两层细分，每层在各边中点位置添加一个顶点，三个顶点相互连接，得到下一层网格。红色网格是细分一层后的网格，绿色网格是细分两层后的网格。

上述细分过程已经得到了半规则网格的拓扑结构，接下来需要基于基网格表面的重心坐标参数化信息计算每个新增顶点坐标。由于初始网格顶点和细分网格时细分顶点均在基网格表面，所以我们只需要确定顶点投影在哪个三角形内。这属于标准的非规则网格点定位问题，我们采用Brown[[40](#_ENREF_40)]的方法寻找目标网格，是为了避免遇到非德劳内三角形[[41](#_ENREF_41), [42](#_ENREF_42)]，而产生搜索闭环问题。当找到*q*所在的三角面片，顶点*q*投影在基网格平面上表示为：

|  |  |  |
| --- | --- | --- |
|  |  | (5.10) |

所以，顶点*q*在细分网格上的坐标为：

|  |  |  |
| --- | --- | --- |
|  |  | (5.10) |

* + 1. 平滑参数化信息

上述网格细分策略能够得到半规则网格，并且该网格与初始网格有着近似的网格表面，但该策略最大的问题是，以中点1：4细分方法细分基网格表面三角形得到的新三角面片，其形状近似于对应的基网格上三角形的四分之一，由于简化算法无法保证基网格表面三角形形状相同，所以属于同一个基网格上三角形细分出的新网格形状近似，而属于基网格上不同三角形细分出的新网格形状相差较大，所以基网格的边界仍然十分清晰。为了解决网格不均匀问题，需要对网格进行全局优化。传统的方法是建立从到的映射函数，全局优化这个映射函数，这样就需要迭代求解偏微分方程。

我们介绍一种操作简单、算法效率高的参数化平滑的方法。我们采用Loop subdivision中顶点平滑权重函数，平滑投影在基网格平面上的顶点，即平滑网格结构，而不是。我们定义函数表示基网格表面的顶点对应到通过Loop平滑函数平滑后的新顶点，每个细分的顶点根据其映射的初始网格三角形，将顶点分为如下三类：

1. 若初始网格三角形的三个顶点全部简化在基网格的同一个三角面片内，我们仅需要根据Loop平滑权重平滑，新生成的顶点也在同一个平面内。
2. 若初始网格三角形的三个顶点简化后位于基网格相邻的两个三角面片内（如图5.8a所示），则依据其公共边，将两个三角形摊开成一个平面，这样三个顶点位于统一个平面内，再进行平滑。
3. 若初始网格三角形的三个顶点简化后位于基网格相邻的多个三角面片内（如图5.8b所示），则采用上述的保角投影法，依据公共顶点，将邻域三角形摊成一个平面，再进行平滑。

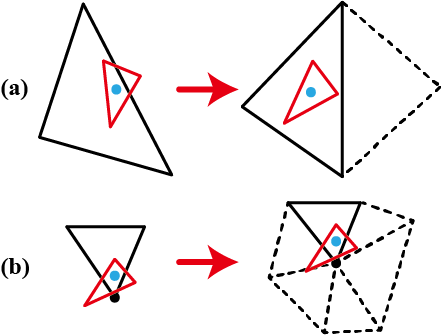


图 5.8平滑参数化示意图。若初始网格三角形的三个顶点没有在基网格同一个三角形上，则依据不同情况，将基网格三角形摊开为一个平面。

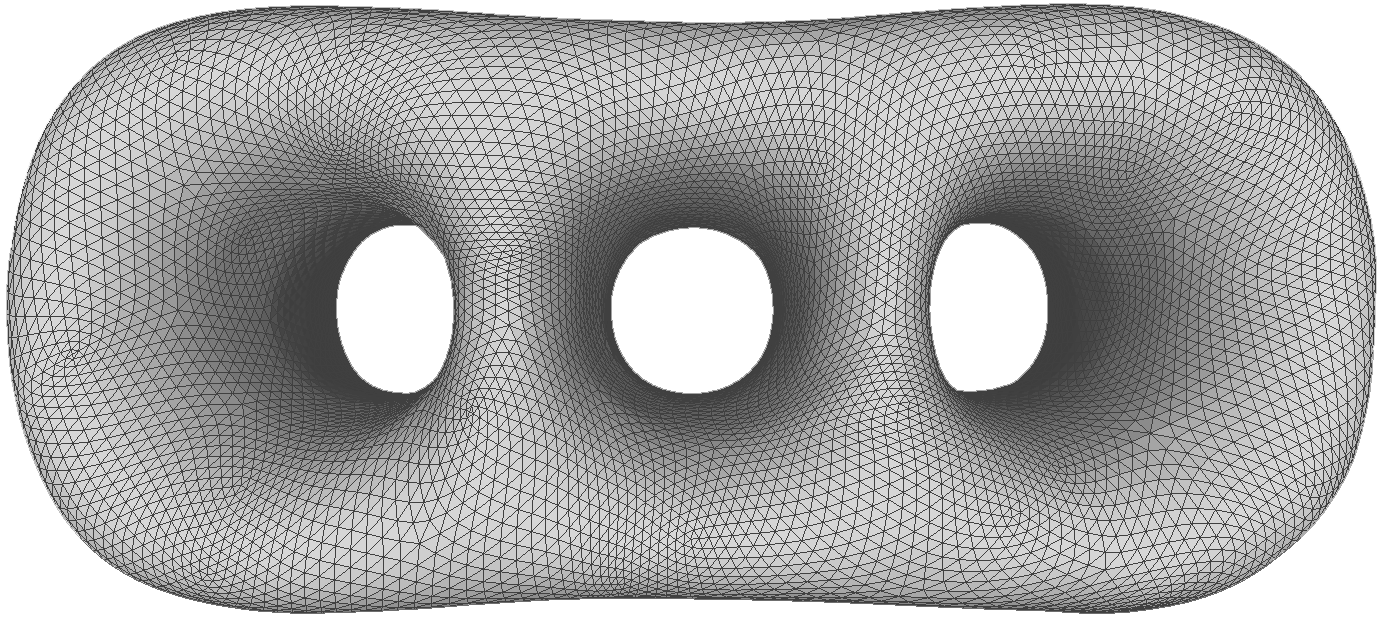
将细分网格映射到初始网格，平滑其映射参数，该过程用表示，则第*m*层细分后的网格表示为，其表示基网格表面的顶点，表示使用Loop算法平滑后的顶点。至此，我们得到了算法最终的半规则化网格。如图5.9所示，是经过3层细分得到的半规则网格。

图 5.9参数化平滑后结果图。

在网格细分过程中，无论在细分，反映射，参数平滑，我们均采用了效率较高的算法。为了提高算法效率而达到实时性，细分过程仍需要并行计算。在细分过程中，每个基网格都是独立的1：4的细分，所以可以直接并行计算。而在平滑参数化时，初始网格投影在基网格上的三角形的三个顶点会跨越基网格的多个三角形（如图5.8a和图5.8b），并行粒度无法以基网格表面来并行化。解决的方法是，以图5.8b中，一个保角映射的环形邻域作为一个独立单位进行并行，基网格每个顶点对应一个线程，每个线程首先通过保角映射方法将其顶点邻域三角形投影到一个平面上，然后对环形邻域内所有三角形内的点进行参数化平滑和反投影。该并行化方法不仅有高并行粒度，并且也具有很高的运行效率。

* 1. 实验结构及分析

本章提出了并行的半规则网格化方法应用在四个经典三角网格模型来进行实验，他们分别是3-holes，Armadillo，Bunny，Dragon，如图5.10。实验将在两个方面进行验证，算法效率和模型精度。

* + 1. 算法效率

本章算法在PC机上进行测试验证，PC机配置如下，Intel I7处理器，主频为3.4GHz，内存为8GB。本章算法在CPU上，4线程并行计算，如表5.3和表5.4所示，本章算法在包含几十万级别三角面片的网格上做半规则网格化，运行时间在30ms以内，相当于每秒至少处理30三角网格，可以达到实时的效果。

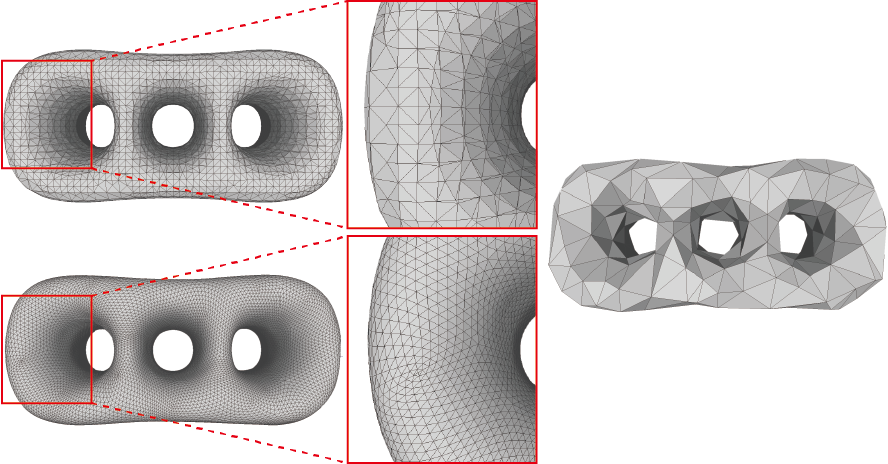
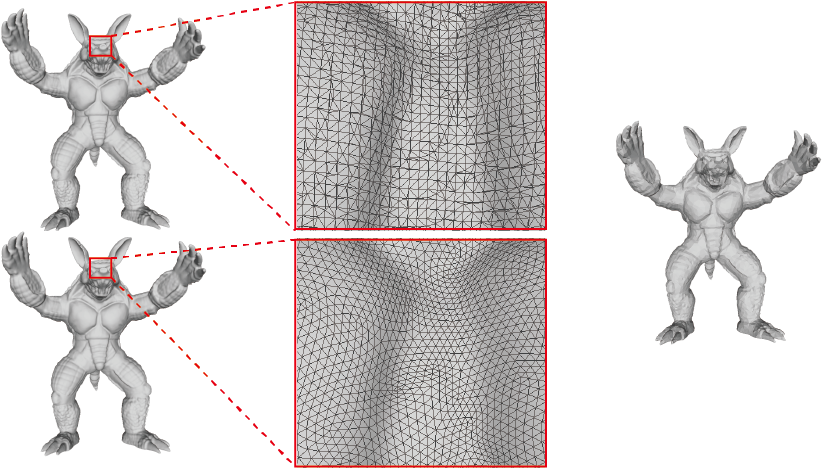
本章算法运行时间依赖于网格简化的层数和细分的层数，根据实验表面，每层简化网格中顶点个数和三角面片个数减少六分之一。实验的四组网格数据，简化率从1.2%至10%不等，简化的层数越多，半规则网格中规则的顶点个数越多，即顶点度为6。从测试的四组网格中可以看出，简化率低于10%，细化网格层数高于2层，能达到很好的半规则网格结构，结果如图5.10所示。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表 5.3网格简化运行时间表 | | | | | | | | | | | |
| 网格 | 顶点数 （初始网格） | | 三角形数 （初始网格） | | 简化层数 | | 顶点数 （基网格） | | 三角形数（基网格） | 运行时间 | |
| 3-holes | 5388 | | 10784 | | 17 | | 250 | | 508 | 5ms | |
| Armadillo | 534712 | | 1069420 | | 24 | | 6943 | | 13883 | 148ms | |
| Bunny | 47426 | | 94848 | | 12 | | 5489 | | 10974 | 11ms | |
| Dragon | 72570 | | 145140 | | 14 | | 5239 | | 10478 | 15ms | |
| 表 5.4网格细分运行时间表 | | | | | | | | | | | |
| 网格 | 顶点数 （基网格） | 三角形数 （基网格） | | 细分层数 | | 顶点数 （细分网格） | | 三角形数 （细分网格） | | | 运行时间 |
| 3-holes | 250 | 508 | | 3 | | 16252 | | 32512 | | | 2ms |
| Armadillo | 6943 | 13883 | | 3 | | 444258 | | 888512 | | | 43ms |
| Bunny | 5489 | 10974 | | 2 | | 87794 | | 175584 | | | 8ms |
| Dragon | 5239 | 10478 | | 3 | | 335328 | | 670656 | | | 17ms |

* + 1. 网格精度

为了度量算法生成的半规则网格与初始网格间的误差，我们采用均方误差（）来度量两个网格间距离。本章算法也与两种经典半规则网格化文献{Lee, 1998 #16}（Maps算法）和文献{Guskov, 2000 #20}（Normal Mesh算法）进行对比。通过表5.5实验表明，本章算法与Maps算法和Normal Mesh算法具有近似的网格精度，但本章算法可以进行并行计算，大大提高算法的效率，可以实时得到半规则化网格。

|  |  |  |  |
| --- | --- | --- | --- |
| 表 5.5与经典半规则网格化精度对比 | | | |
| 网格 | Maps算法 | Normal Mesh算法 | 本章算法 |
| 3-holes | 0.073 | 0.098 | 0.082 |
| Armadillo | 0.135 | 0.182 | 0.157 |
| Bunny | 0.081 | 0.103 | 0.095 |
| Dragon | 0.116 | 0.136 | 0.124 |

最后给出实验的四组实验结果图（如图5.10），每组结果图分别包含：初始网格、简化后网格和半规则网格。

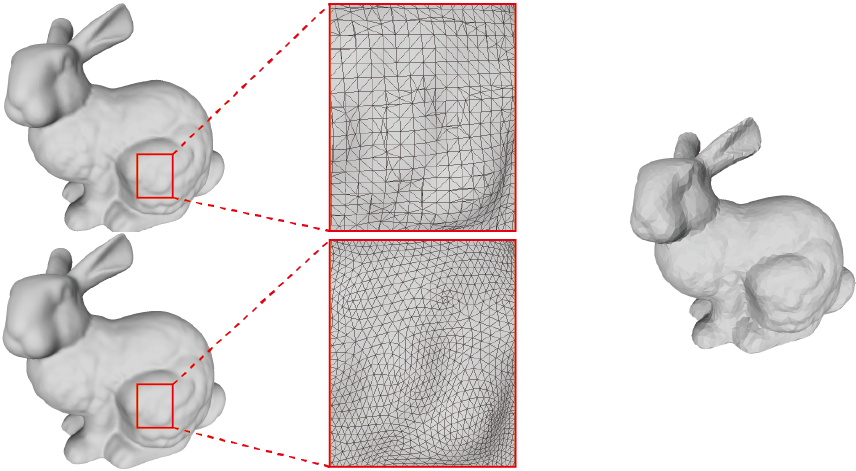
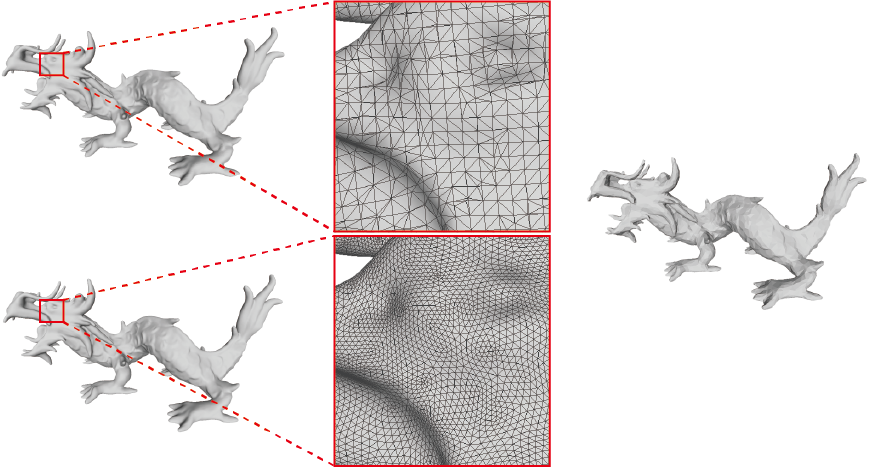
* 1. 本章小结

图 5.10半规则网格结果图。四组实验结果其网格分别为3-holes，Armadillo，Bunny和Dragon，每组结果左上图为输入的非规则网格，左下图为半规则网格，右图为简化后的基网格。

本章给出了一种实时的半规则网格化的算法，该算法可以实时地将一个非规则网格转化为一个半规则网格，其半规则网格可以广泛应用于网格压缩，模型渲染等领域。本章的半规则网格化主要分为两个步骤：首先将输入的非规则网格分层简化为基网格，然后将基网格细分重建为半规则网格。简化过程基于二次误差度量简化方法，分层进行简化，并将简化时删除的顶点参数化投影在基网格的三角形表面。为了使简化算法能够并行运算而提高效率，在简化时定义独立区域，任意两个独立区域间可以并行简化。细分重建网格过程，将基网格细分为半规则网格，其细分出的新网格均为规则网格，细分出的新顶点根据基网格表面的参数化信息反映射到三维空间，得到与初始网格相近似的半规则网格。本章最后通过实验证明本章并行算法可以做到实时半规则化目标网格，同时可以得到与初始网格误差较小的半规则化网格。

# 总结和展望

* 1. 工作总结
  2. 下一步工作

参考文献

[1] S. G. Mallat*,* A Theory for Multiresolution Signal Decomposition: The Wavelet Representation[M]: IEEE Computer Society, 1989.

[2] A. Certain. Interactive multiresolution surface viewing[C]. Conference on Computer Graphics and Interactive Techniques, 1996, 91-98.

[3] C. Roudet, F. Dupont, A. Baskurt. Multiresolution mesh segmentation based on surface roughness and wavelet analysis[C]. Electronic Imaging, 2007

[4] A. Khodakovsky, I. Guskov*,* Compression of Normal Meshes[M]: Springer Berlin Heidelberg, 2004.

[5] S. Lavu, H. Choi, R. Baraniuk. Geometry compression of normal meshes using rate-distortion algorithms[C]. Eurographics/acm SIGGRAPH Symposium on Geometry Processing, 2003, 52-61.

[6] M. Antonini*,* An efficient bit allocation for compressing normal meshes with an error-driven quantization[M]: Elsevier Science Publishers B. V., 2005.

[7] J. Y. Sim, C. S. Kim, C. C. J. Kuo, S. U. Lee. Rate-distortion optimized compression and view-dependent transmission of 3-D normal meshes[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2005, 15(7), 854-868.

[8] L. Denis, S. M. Satti, A. Munteanu, J. Cornelis, P. Schelkens. Scalable Intraband and Composite Wavelet-Based Coding of Semiregular Meshes[J]. IEEE Transactions on Multimedia, 2010, 12(8), 773-789.

[9] M. Garland. Surface simplification using quadric error metrics[J]. Acm Siggraph Computer Graphics, 1997, 1997(209-216.

[10] F. Payan, C. Roudet, B. Sauvage*,* Semi-Regular Triangle Remeshing: A Comprehensive Study[M]: The Eurographs Association & John Wiley & Sons, Ltd., 2015.

[11] P. Alliez, M. Meyer, M. Desbrun. Interactive geometry remeshing[J]. Acm Transactions on Graphics, 2002, 21(3), 347-354.

[12] J. Schreiner, A. Asirvatham, E. Praun, H. Hoppe. Inter-surface mapping[J]. Acm Transactions on Graphics, 2004, 23(3), 870-877.

[13] M. S. Floater, H. Kai*,* Surface Parameterization: a Tutorial and Survey[M]: Springer Berlin Heidelberg, 2005.

[14] A. W. F. Lee. MAPS: multiresolution adaptive parameterization of surfaces[J]. ACM Computer Graphics (Proc. SIGGRAPH, 1998, 40(2), 189-218.

[15] M. Lounsbery, T. D. Derose, J. Warren. Multiresolution Analysis for Surfaces of Arbitrary Topological Type[J]. Acm Transactions on Graphics, 1993, 16(1), 3--4.

[16] M. Eck. Multiresolution analysis of arbitrary meshes[C]. 1995, 173-182.

[17] P. Gioia. Reducing the number of wavelet coefficients by geometric partitioning ☆[J]. Computational Geometry, 1999, 14(1–3), 25-48.

[18] I. Guskov, W. Sweldens. Normal meshes[C]. Conference on Computer Graphics and Interactive Techniques, 2000, 95-102.

[19] A. Lee, H. Moreton, H. Hoppe. Displaced subdivision surfaces[C]. Conference on Computer Graphics and Interactive Techniques, 2000, 85-94.

[20] K. Hormann, U. Labsik, G. Greiner. Remeshing triangulated surfaces with optimal parameterizations[J]. Computer-Aided Design, 2001, 33(11), 779-788.

[21] A. Khodakovsky, N. Litke, P. Schröder. Globally Smooth Parameterizations with Low Distortion[J]. Acm Transactions on Graphics, 2003, 22(3), 350-357.

[22] I. Friedel, P. Der, A. Khodakovsky. Variational normal meshes[J]. Acm Transactions on Graphics, 2004, 23(4), 1061-1073.

[23] I. Guskov. Manifold-based approach to semi-regular remeshing[J]. Graphical Models, 2007, 69(1), 1-18.

[24] L. Denis, A. Munteanu, P. Schelkens. Semi-regular remeshing with reduced remeshing error[C]. IEEE International Conference on Image Processing, 2010, 365-368.

[25] A. Kammoun, F. Payan, M. Antonini. Adaptive semi-regular remeshing: A Voronoi-based approach[C]. IEEE International Workshop on Multimedia Signal Processing, 2010, 350-355.

[26] C. H. Chiang, B. S. Jong, T. W. Lin. A robust feature-preserving semi-regular remeshing method for triangular meshes[J]. Visual Computer, 2011, 27(9), 811-825.

[27] W. J. Schroeder, J. A. Zarge, W. E. Lorensen. Decimation of triangle meshes[J]. Acm Siggraph Computer Graphics, 1992, 26(2), 65-70.

[28] J. Rossignac, P. Borrel*,* Multi-resolution 3D approximations for rendering complex scenes[M]: Springer Berlin Heidelberg, 1993.

[29] P. Lindstrom. Out-of-core simplification of large polygonal models[C]. ACM SIGGRAPH, 2000, 259-262.

[30] S. Schaefer, J. Warren. Adaptive Vertex Clustering Using Octrees[C]. SIAM Geometric Design and Computing, 2003, 491-500.

[31] A. Gueziec. Surface simplification with variable tolerance[J]. Proc.int.symp.on Medical Robotics & Computer Assisted Surgery, 1995,

[32] D. Cabiddu, M. Attene. Large mesh simplification for distributed environments[J]. Computers & Graphics, 2015, 51(81-89.

[33] H. Xiong, X. Jiang, Y. Zhang, J. Shi. Parallel simplification of large meshes on PC clusters[C]. Eurographics Conference on Parallel Graphics and Visualization, 2008, 33-40.

[34] N. Tatarchuk, N. Tatarchuk. Real-time mesh simplification using the GPU[C]. Symposium on Interactive 3d Graphics and Games, 2007, 161-166.

[35] S. M. Shontz, D. M. Nistor*,* CPU-GPU Algorithms for Triangular Surface Mesh Simplification[M]: Springer Berlin Heidelberg, 2013.

[36] A. Papageorgiou, N. Platis. Triangular mesh simplification on the GPU[J]. Visual Computer, 2015, 31(2), 235-244.

[37] F. Cellier, P. M. Gandoin, S. Akkouche. Simplification and streaming of GIS terrain for web clients[C]. International Conference on 3d Web Technology, 2012, 73-81.

[38] M. Botsch, S. Steinberg, S. Bischoff, L. Kobbelt, R. Aachen. OpenMesh: A generic and efficient polygon mesh data structure[J]. Opensg Symposium, 2002,

[39] T. Duchamp, A. Certain, A. Derose. Hierarchical Computation Of Pl Harmonic Embeddings[J]. Siam Journal on Matrix Analysis & Applications, 1997,

[40] P. J. C. Brown, C. T. Faigle. A robust efficient algorithm for point location in triangulations[J]. University of Cambridge Computer Laboratory, 1997,

[41] M. Garl, P. S. Heckbert. Fast Polygonal Approximation of Terrains and Height Fields[J]. Submitted for Publication, 1997,

[42] L. J. Guibas, J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams[C]. ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, Usa, 1983, 221-234.