

大连理工大学本科毕业设计（论文）

基于手绘草图的快速植物建模研究

Sketch-based Fast Plant Modeling

学院（系）： 计算机科学与技术学院

专业： 计算机科学与技术

学生姓名： 何振邦

学号： 201585043

指导教师： 徐喜荣副教授

评阅教师： _____

完成日期： 2019.6.4

大连理工大学

Dalian University of Technology

原创性声明

本人郑重声明：本人所呈交的毕业设计（论文），是在指导老师的指导下独立进行研究所取得的成果。毕业设计（论文）中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，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。对本文的研究成果做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。

作者签名：

日 期：

关于使用授权的声明

本人在指导老师指导下所完成的毕业设计（论文）及相关的资料（包括图纸、试验记录、原始数据、实物照片、图片、录音带、设计手稿等），知识产权归属大连理工大学。本人完全了解大连理工大学有关保存、使用毕业设计（论文）的规定，本人授权大连理工大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业设计（论文）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为大连理工大学。本人离校后使用毕业设计（论文）或与该论文直接相关的学术论文或成果时，第一署名单位仍然为大连理工大学。

论文作者签名：

日期：

指导老师签名：

日期：

摘要

植物作为现实世界的重要景观，植物建模一直是三维建模研究中的热点课题。随着电子游戏、影视特效和虚拟现实相关产业的发展，人们对具有形态多样、较强真实感的植物模型的需求与日俱增。然而由于植物自身结构的复杂性，使用传统方法对植物进行建模十分困难，如何快速构建出所需的植物模型是目前亟待解决的一大难题。

为了实现快速的植物建模，本文提出一种基于草图的植物建模方法，首先对创作者绘制的草图进行预处理和分析，推测创作者的创作意图；通过观察植物样本和查阅相关的植物学知识，提出了基于植物学知识的深度信息恢复算法，能够从二维草图中恢复缺失的深度信息；最后根据深度信息和植物的特点建立出目标植物模型。

本文实现了一个基于草图的快速植物建模系统，能够让创作者自由绘制欲建模植物的二维草图，并据此构建出具有一定真实感的三维模型。仅仅需要寥寥数笔即可快速生成相应的三维植物模型，本文系统与传统植物建模方法相比具有较高的效率，并能够导出为 OBJ 格式的模型，满足了人们对快速植物建模的需求。

关键词：植物建模；基于草图的建模；三维重建

Sketch-based Fast Plant Modeling

Abstract

As an important landscape in real world, plants have been the focus of three-dimensional modeling. With the development of computer games, film and virtual reality industries, the demand of realistic plant models is constantly increasing. However, it is significantly difficult to construct plant models due to the complex structure of plants. Thus how to construct the required plant models rapidly to meet the application needs is a major problem that needs to be solved urgently.

To achieve rapid plant modeling, we propose a sketch-based plant modeling method. Firstly, we perform pre-processing and analysis on the user's sketch to infer user's creative intention. By observing plant samples and consulting botany knowledge, then we propose a botany-knowledge-based depth recovery algorithm which is able to recover lost depth information. Finally we construct plant model according to depth information and plant characteristic.

We implement a sketch-based rapid plant modeling system which allows user to draw plant sketch freely and construct a realistic 3D model. Plant model can be rapidly generated with only a few strokes. Our system is more efficient than traditional plant modeling methods and can export the model as OBJ format to meet the needs of rapid plant modeling.

Key Words: Plant Modeling; Sketch-based Modeling; Three-dimensional Reconstruction

目 录

摘要.....	I
Abstract	II
1 绪论.....	1
1.1 选题背景与意义.....	1
1.2 植物建模研究的历史与现状.....	2
1.2.1 基于规则的过程式建模方法.....	2
1.2.2 基于真实数据的建模方法.....	3
1.2.3 基于草图的建模方法.....	4
1.3 课题研究内容与目标.....	5
1.4 论文结构与内容.....	6
2 草图的获取、预处理.....	7
2.1 草图获取.....	7
2.2 草图预处理.....	7
2.2.1 草图重采样.....	8
2.2.2 草图平滑.....	8
3 草图分析.....	10
3.1 笔划分类.....	10
3.2 笔划特征提取.....	10
3.3 花瓣聚类.....	11
3.4 枝干层级分析.....	12
3.5 轮廓填充.....	14
3.5.1 枝干轮廓填充.....	14
3.5.2 叶片轮廓填充.....	16
4 深度信息获取.....	17
4.1 花朵深度信息获取.....	17
4.2 枝干深度信息获取.....	19
4.3 叶片深度信息获取.....	21
5 模型构建与优化.....	22
5.1 花朵模型构建.....	22
5.1.1 曲面细分.....	22
5.1.2 花瓣弯曲模拟.....	23

5.1.3 花瓣内凹模拟.....	24
5.1.4 Delaunay 三角剖分	26
5.2 叶片模型构建.....	30
5.3 枝干模型构建.....	30
5.3.1 各级枝干半径确定.....	30
5.3.2 枝干半径衰减.....	30
5.3.3 枝干几何形体建模.....	31
6 模型的渲染与导出.....	33
6.1 模型渲染.....	33
6.1.1 纹理与材质库.....	33
6.1.2 渲染模块.....	34
6.2 模型导出.....	35
6.2.1 OBJ 格式简述	35
6.2.2 导出 OBJ 模型	36
7 结果与分析.....	38
7.1 建模效果.....	38
7.1.1 系统建模效果.....	38
7.1.2 模型可视化效果.....	40
7.2 建模性能与质量.....	42
结 论.....	44
参 考 文 献.....	45
修改记录.....	47
致 谢.....	48

1 绪论

1.1 选题背景与意义

植物作为现实世界中的常见的自然景观，植物模型在计算机影视、游戏和虚拟现实应用中具有广泛的应用，然而植物模型的建立却十分困难和繁琐，这是因为植物具有较为复杂的结构和多变的外形。传统的植物建模方法是艺术家使用例如 3ds max、Maya 等三维建模软件根据真实的植物图片或者实物作为参考进行细致地建模，然而这种方法非常繁琐和低效；当艺术家想要设计出不存在的植物模型时，这个任务会变得更加艰巨。因此高效、快速的植物建模方法一直是计算机图形学研究中一个十分重要且具有挑战性的课题。



图 1.1 各种形态多样的植物

过去几十年里，研究人员和学者们发展出了许多有效的植物建模方法，如基于规则的过程式建模方法、基于视频图像和三维点云的重建方法、基于手绘草图的建模方法^[1]。

其中基于手绘草图的建模作为一种新颖、高效的建模方法，近来得到了许多学者的广泛关注。基于手绘草图的三维建模方法快速、简便、灵活，创作者可以根据自己的想法绘制草图并快速生成所期望的模型而无需经过特殊训练，相比其他方法整个建模过程大大简化了，建模效率也有着极大的提高。另外其他建模方法如基于规则的建模方法对于植物的外形控制较为困难，需要使用者具有丰富的植物学专业知识且熟悉各种参数的调整才能得到理想的目标模型，此外模型生成之后也不易进行修改；而基于手绘草图的建模方法能给与创作者较大的自由度去生成各种复杂外形的植物模型而无需了解模型生成的相关细节。因此基于手绘草图的建模方法具有很高的实用价值、具有十分重要的研究意义。

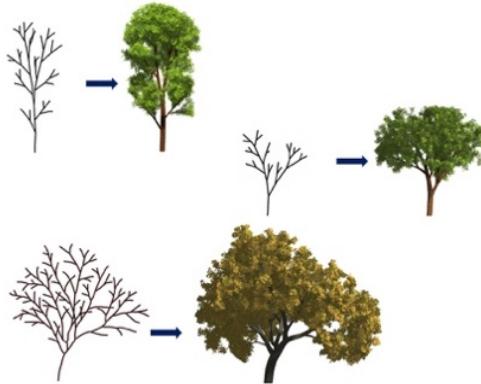


图 1.2 基于手绘草图的植物建模

基于手绘草图的建模方法首先需要获取创作者的草图，从二维草图中提取各种笔画特征，分析草图表达的含义并理解创作者的表达意图，以此为导向进行后续的处理；然后根据创作者绘制出的二维图形进行三维重建，这个过程不可避免地要面临深度信息缺失的问题，正确的深度信息对生成具有较强真实感的植物模型具有至关重要的作用。另外如何从草图轮廓中识别植物类型并结合植物学专业知识进行相应的细节的补充和润色，使得最终产生的模型符合自然规律、具有一定的真实感，这也是基于手绘草图的建模方法需要解决的问题。由此可知，基于手绘草图的建模方法具有其他方法不可比拟的优点，但同时也存在着许多亟待解决的问题，因此基于手绘草图的植物建模的相关工作是具有很高的应用价值与研究意义的。

1.2 植物建模研究的历史与现状

植物建模一直是计算机图形学领域的一个重要研究课题，还发展出了树木建模、花卉建模等子课题；在过去几十年里，学者和研究人员们发展出了许多有效的植物建模方法，下面对各种植物建模方法的历史与现状进行简要介绍。

1.2.1 基于规则的过程式建模方法

基于规则的过程式建模方法是最早被提出和应用的植物建模方法。早期的过程式建模方法主要依据植物枝干的生长特性所提出的几何规则，如树木枝干的角度、长度比例等^[2]，然而这种方法过于繁琐，建模过程中需要定义大量参数；1968 年学者 Lindenmayer^[3]通过观察植物枝干的分形结构提出了 L-System 文法系统^[4]；L-System 是一个基于字符串的替换和重写系统，通过反复使用不同的规则来生成字符串，模拟植物

的生长的模式来构造分形结构，学者 Prusinkiewicz^[5]提出这些字符串可以赋予一定的图形意义来表达植物模型结构，一个字符串对应了一种植物模型。

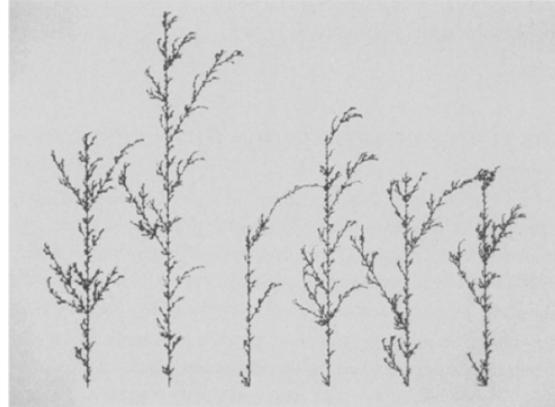


图 1.3 通过 L-System 生成的植物模型

L-System 具有很强的表达能力，能够描述各种复杂的植物结构，然而 L-System 对于普通用户来说并不友好，使用该系统进行植物建模需要一定的数学和植物学专业背景知识；另外使用 L-System 难以控制植物的整体轮廓外形，亦有学者通过改进 L-System 来产生外形可控的植物模型^[6]。

现实世界中的植物还存在着空间竞争、趋光性等特点，单单使用 L-System 是无法模拟这种行为的；学者 Runions^[7]提出了一种空间殖民地算法来模拟这种树木生长的竞争过程；Palubicki^[8]提出将叶序等植物学因素加以考虑的自组织算法，使得生成的模型更为逼真、自然。

1.2.2 基于真实数据的建模方法

要想建立真实的植物模型，最好的方法还是参考真实植物的情况。随着硬件技术的发展，高清相机和高精度三维扫描仪越来越普及，我们能够获得较高精度的植物图像和植物点云数据集，使得从植物图像和点云数据中重建植物模型成为可能。

Shlyakhter^[9]提出一种基于照片的树木三维重建方法，通过抽取照片内树木轮廓为参考生成树木主干，再结合 L-System 形成枝干分支来重建照片中的树木模型。Tan^[10]等人提出一种单张照片恢复树木模型的方法，通过对追踪图片中可见的树木主干为基础，结合树木轮廓，提出一种数据驱动的吸引子生长方法来补充被树叶遮挡的树枝部分。Yan^[11]结合一定的先验知识来对花卉从单幅照片进行三维重建，首先提取出花心以及各个花瓣的顶点以及中轴，计算花瓣顶点围成的椭圆，假定花瓣围成一个圆锥且这椭圆是三维圆锥在图像平面的投影，从而可以还原各个花瓣的大致三维位置，再进行网格化和微调重建出花卉的三维模型。

随着激光扫描仪的普及，我们可以很容易获取植物的点云数据并进行三维重建。实际上由于现实场景中各种物体遮挡和测量角度限制，加之植物自身结构存在不可避免的自身遮挡现象导致获取的点云数据是不完整、有噪声的，如何从这些点云数据中重建出三维植物模型是不小的挑战。

Xu^[12]提出对点云数据进行带权图的构造，并使用最小路径算法进行骨架结构的提取，结合植物学中的先验知识对初始骨架进行细分，直到细化生成所有枝干。

Livny^[13]提出了一种轻量级的树木表达方式，从杂乱的点云数据中提取出较为清晰的主干骨架，而对于树叶部分噪声较多的部分采用一种称为 Lobe 的结构表达，Lobe 拟合点云数据区域空间，Lobe 内使用预储存的样本枝干进行填充，这比直接从点云数据构建树木模型更快速、占用的空间更小，还能通过在 Lobe 内使用不同填充率完成不同层次的渲染。

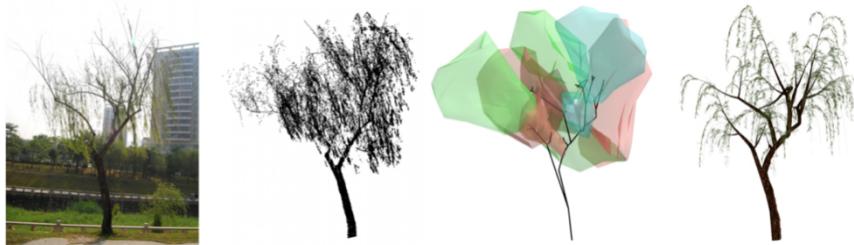


图 1.4 通过 Lobe 表达方式建模产生的树木模型

1.2.3 基于草图的建模方法

从现实场景重建现实世界的植物有时不能满足人们的需求，因此基于手绘草图的植物建模方法被提出，该方法允许人们以直观的方式来表达其建模意图，从而能够自由创建生活中不存在的植物模型，同时还大大降低了建模的门槛、提高了建模的效率，是一种新颖的植物建模方法。

国外在手绘草图建模方面的研究工作开展得比较早，手绘建模最早是在 1999 年由 Igarashi^[14]等人在开发基于手绘草图的三维建模软件 Teddy 时提出的，该系统通过根据草图轮廓进行膨胀来产生低复杂度的三维模型。随后草图三维建模开始逐渐流行并应用在各个行业中；Okabe^[15]等人最早提出将手绘草图建模应用在树木建模上，并设计了一个交互式系统来帮助用户从草图生成真实的树木模型，该系统可以在 10 分钟内从草图中构建一棵树的三维模型；

Anastacio^[16]等人提出使用基于构造线的概念草图方式去定义植物器官的排列顺序全局结构，而局部使用 L-System 进行细节创建。Ijiri^[17]等人也提出使用草图进行树木整

体结构与轮廓的控制，并结合 L-System 进行局部枝干生成的树木建模算法；Longay 等人^[18]在平板上实现一个手绘树木建模系统，该系统结合了程式式建模和自组织建模方法，可以根据手绘草图模拟树木生长产生真实的树木模型；Ijiri^[19]等人还将手绘草图建模应用在花卉建模上，开发出了根据手绘花卉草图和花序图进行建模的交互式花卉建模系统。唐棣^[20]等提出了一种素描式三维花朵建模方法，通过提取草图中的侧影线与边界轮廓线等笔画信息计算出花瓣的横截曲线来构造花瓣的曲面形状；濮群^[21]等针对现有花朵建模方法形成的花瓣式样与形态单一的缺陷，提出首先根据手绘花瓣的轮廓图提取花瓣模型的二维控制点，然后通过弯曲控制函数得到花瓣的三维控制点，继而利用三维控制点构造三维花瓣模型的方法；目前大部分基于手绘草图的植物建模研究主要围绕树木的建模，其他植物的建模少有涉及，主要原因是树木的建模研究工作较多，已有的相关的研究能较好转移到草图建模中来。

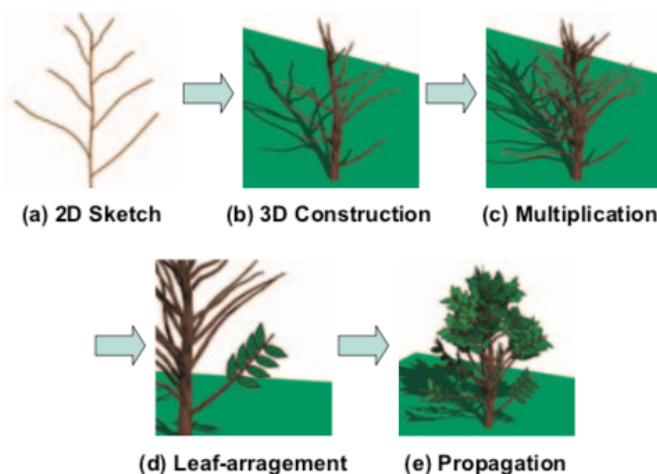


图 1.5 Okabe 的系统的植物模型生成过程

1.3 课题研究内容与目标

本课题的主要研究内容是基于手绘草图的快速植物建模方法的研究，特别是花卉、草本植物的建模，探索如何从二维手绘草图中重建出具有较强真实感的三维植物模型；课题的研究内容包括草图的获取、草图的预处理与特征分析、深度信息的获取、模型建立和细节调整、模型的渲染与导出，其中最为核心的问题是如何从二维草图中恢复出深度信息，并根据三维信息构建出合理的植物模型。

本课题的主要研究目标有：

- (1) 对创作者自由绘制的草图进行预处理和分析，识别草图中的元素并推测创作者建模意图；
- (2) 结合植物学知识，完成能够从二维草图中恢复出模型的深度信息的算法；
- (3) 根据三维信息构建出具有一定的真实感、符合自然规律的植物模型，首先以花为主要关注点，后期再扩展对其他植物如草本植物、树木的建模支持。
- (4) 实现对生成的植物模型的实时渲染和常见三维模型格式的导出。

本课题最终实现的效果是：完成一个基于手绘草图的植物建模系统，创作者可以通过交互式界面，在草图绘制面板绘制出欲构建的植物草图，植物的种类可以是花朵、草本植物或者简单的树木，在进行少量参数的设置后点击生成按钮即可获得一个较为真实的植物模型，创作者可以实时的从各个角度观察模型，并可以选择导出具有纹理和材质的 OBJ 格式的三维植物模型。

1.4 论文结构与内容

本文共分为五个章节：

第一章介绍课题背景、植物建模相关研究的历史与现状以及几种常见的植物建模方法，提出了本课题的主要研究内容与目标。

第二章介绍草图的获取、对草图进行预处理的算法。

第三章介绍对草图进行分析，提取出各种草图特征的方法。

第四章介绍深度信息的获取，如何借助植物学先验知识，从二维草图中还原出合理的深度信息。

第五章介绍从前几步获得的三维数据中构建植物模型的算法以及能够使得模型具有较强的真实感的细节优化算法。

第六章介绍植物模型的实时渲染与 OBJ 格式的模型导出。

2 草图的获取、预处理

2.1 草图获取

草图的获取是基于手绘草图建模的第一步，在获取草图时要充分考虑到系统对创作者创作的自由度与便利性，同时也要兼顾对后续处理的难易度。本课题设计了一个基于 Qt 框架的草图绘制程序，创作者可以使用鼠标或者数位板在绘图区域自由绘制植物草图，其中草图支持以下元素：花瓣、叶片、枝干、细枝干轮廓、叶片轮廓，各个元素的示意简图如图 2.1 所示。

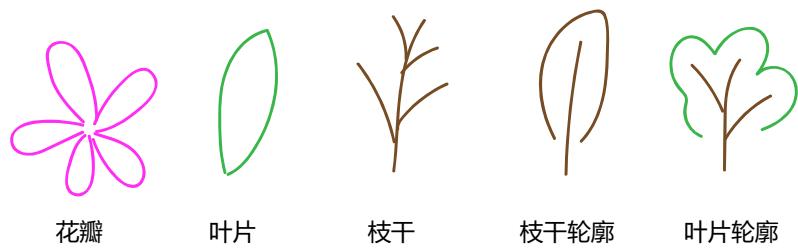


图 2.1 草图中支持绘制的各元素示意简图

为了处理方便，程序规定创作者绘制花瓣、叶片、枝干时需要选择元素模式（Element Mode），绘制细枝干轮廓需要选择枝干轮廓模式（Branch Profile Mode），绘制叶片轮廓需要选择叶片轮廓模式（Leaf Profile Mode），且需要一笔绘制出单个元素，即一个笔划的笔迹中不能有断笔；考虑到有些缺乏绘画经验的创作者可能出现绘制失误的情况，系统还设置了撤销和重做功能，便于创作者修改草图。

当创作者使用程序进行绘图时，程序会在后台实时地对笔划像素进行标记并储存像素信息，当一个笔划绘制完毕后，程序将刚刚绘制完的笔划储存到笔划队列中去；笔划队列中储存了创作者所有已经绘制的笔划，包括单像素宽的笔划的所有像素信息、笔划 ID 以及当前选择的绘制模式，作为原始草图的存储数据结构；当草图的绘制完成后，程序将笔划队列传递到下一步进行处理。

2.2 草图预处理

创作者在绘制草图时大概率会出现一些微小失误，例如笔划的抖动、应当接触的元素之间有空隙等，这些失误如果不进行处理则将会在模型中如实反映，影响到模型的质感；除此之外上一步的草图数据不仅存在着数据冗余，同时也不利于后续步骤的处理，故应当将草图原始数据转换为易于计算机处理的表达形式，因此草图的预处理是十分必要的，良好的草图预处理算法可以提高建模系统的鲁棒性、方便后续处理的进行。

2.2.1 草图重采样

从上一步得到的草图笔划队列中储存了草图笔划的所有像素信息，在后续的处理中并不需要用到全部信息，因此这些信息是冗余的，增加了后续处理的负担；此外由于创作者绘图时光标移动速度不均一，可能出现像素点分布不均匀的情况。为了简化笔划信息、获得间隔均匀的像素点，在预处理步骤中首先应当进行笔划的重采样，即将笔划所有的像素重新采样为若干像素点的顺序集合，通过连接这些采样点形成一系列线段来近似拟合原笔划的轨迹。

本文的笔划采样方法如下：

- (1) 设置采样值 S ，并从笔划像素起点开始，将起点加入采样序列中，将起点设置为上一采样点；
- (2) 移动到笔划原像素下一点，若没有下一点了，则转到 (4)；
- (3) 计算当前点与上一采样点的曼哈顿距离，若该距离在 $S \pm 0.2*S$ 之内，则将当前点加入采样序列中，将当前点设置为上一采样点，转到 (2)；若该距离小于 $0.8*S$ 则忽略当前点，转到 (2)；若该距离大于 $1.2*S$ ，则计算当前点与上一采样点线段上与上一采样点距离为 S 的点 M ，将点 M 加入采样序列中，并将 M 设置为上一采样点，转到 (3)；
- (4) 结束，返回采样序列。

笔划重采样的结果如图 2.2 所示，由图可知，经过重采样操作，原本不均匀的笔划点集变得均匀，有较好的采样效果。

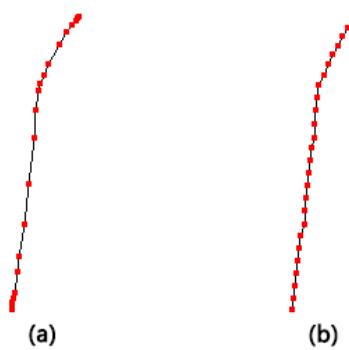


图 2.2 草图笔划重采样结果。 (a) 原始笔划点集； (b) 重采样后笔划点集；

2.2.2 草图平滑

如前文所述，创作者在绘制枝干这类长线条时，很可能会出现抖动等失误，如果不对这种情况进行修正，这些缺陷会如实反映到模型中去，造成模型形态的不自然，影响

模型质感；此外在前一步骤中还进行了草图笔迹的重采样，采样过程可能会导致笔划线段的突变和不流畅；因此，对草图笔划的平滑处理是不可或缺的。

本文使用均值滤波的方法对笔划进行快速平滑处理，即对各点的坐标与其周围点坐标进行取平均操作。

设采样后得到的点集 $P = \{p_1, p_2, \dots, p_n\}$ ，平滑后的点集 $P' = \{p'_1, p'_2, \dots, p'_n\}$ ，有：

$$p'_i = \frac{p_{i-2} + p_{i-1} + p_i + p_{i+1} + p_{i+2}}{5} \quad (2.1)$$

通过平滑处理的笔划效果如图 2.3 所示，可见平滑处理后的草图的笔划更加流畅自然。

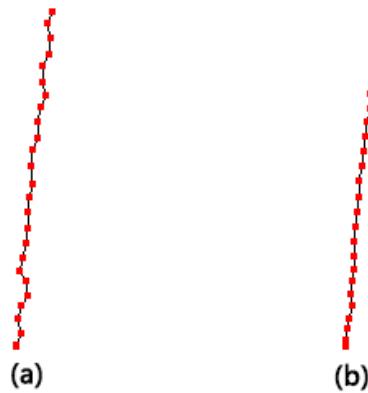


图 2.3 平滑处理后的笔划效果。 (a) 原始笔划； (b) 平滑处理后的笔划；

3 草图分析

通过草图获取和预处理后得到的仅仅是一些点集的集合，这些点集所包含的信息十分有限，因此需要对这些代表原草图的点集集合进行分析，理解草图中所包含的元素，推测创作者的作图意图，赋予点集一定的上下文语义，最终得到建模对象以及相关的参数信息，提供给后续步骤进行真正的建模操作。

3.1 笔划分类

本课题的草图建模系统支持绘制花瓣、叶片、枝干、细枝干轮廓、叶片轮廓五种元素，其中细枝干轮廓和叶片轮廓分别在枝干轮廓模式和叶片轮廓模式下绘制，在这两个模式下均只可能有一种笔划，故无需进行进一步的分类，而花瓣、叶片、枝干均在元素模式下绘制；为了能够对花瓣、叶片和枝干进行分别建模，需要对在元素模式下绘制得到的笔划进行分类。

花瓣和叶片在二维形态上是相似的，而仅仅是纹理以及空间位置上的不同，花瓣和叶片的细分类会在后续的步骤中进行，因此在元素模式下绘制的笔划仅仅需要将分成花瓣、叶片一类和枝干一类。

本文的具体分类方法是：对于点集 $P = \{p_1, p_2, \dots, p_n\}$ ，根据首尾两点信息计算

$$T = |p_1 - p_n| + \frac{1}{|p_i - \frac{p_1 + p_n}{2}|}, \quad i = \frac{1+n}{2} \quad (3.1)$$

若 T 小于设定的阈值 t ，则认为点集 P 代表的笔划是花瓣或者叶片，否则属于枝干。在本文中 t 取值为 8。

3.2 笔划特征提取

枝干、枝干轮廓与叶片轮廓较为简单，而相比之下叶片和花瓣的形态与几何结构较为复杂，需要提取出更多的特征信息进行建模，主要有：根点、尖点、主轴。

(1) 根点是指叶片或花瓣的根部，通常是与枝干接触的部位，根点 p_r 的定义为笔划的起点 p_s 与终点 p_e 的平均值。

(2) 尖点是指叶片或者花瓣的最外的尖峰处点，大多数情况下是离根点最远的点，但并不总是，尖点在位置上的具体特点是远离根点且尖点与两侧点的切线斜率变化较为突变，具体的定义如下：

对于点集 $P = \{p_1, p_2, \dots, p_n\}$ ，若点 p_t 是尖点，则满足使得下式

$$w_1 \cdot |p_r - p_t| + w_2 \cdot \left| \frac{p_{t+1}.y - p_t.y}{p_{t+1}.x - p_t.x} - \frac{p_{t+1}.y - p_t.y}{p_{t+1}.x - p_t.x} \right| \quad (3.2)$$

取得最大值，其中 w_1 、 w_2 为权重因子，在本文中取 $w_1 = 1, w_2 = 0.5$ 。

(3) 主轴是由根点指向尖点的向量，计算出了根点与尖点，主轴也就很容易得出了。

3.3 花瓣聚类

在笔划分类步骤中我们将枝干分为了一类，花瓣和叶片分为了一类；在这个步骤中我们对花瓣和叶片进行进一步分类，同时还将对花瓣进行聚类，找出属于同一朵花的花瓣。

本文的花瓣聚类是基于这样一个观察假设：在自然界中同属于一朵花的花瓣们，它们的由同一个原基区域发育而来，因此根点都彼此非常靠近形成了所谓的花心，而叶片则通常从枝干的侧面单独生长发育而来，很少有几片叶片的根点都挤在一处。因此我们可以对叶片和花瓣的根点进行聚类，若某几个根点相聚集，则说明此处很可能有一朵花的形成。

由于本文程序并未要求创作者提供花朵的数目，聚类簇的数目是未知的，因此本文使用 Mean Shift 算法对花瓣进行聚类。

Mean Shift 算法是一种非参数的特征空间分析方法，其主要思想是沿着密度上升的方向寻找同属一个簇的数据集合，该算法由 Fukunaga^[22]在 1975 年提出相关概念，并后来由 Cheng^[23]进行了改进和完善。Mean Shift 算法在聚类分析、计算机视觉和图像处理中有着广泛的应用。

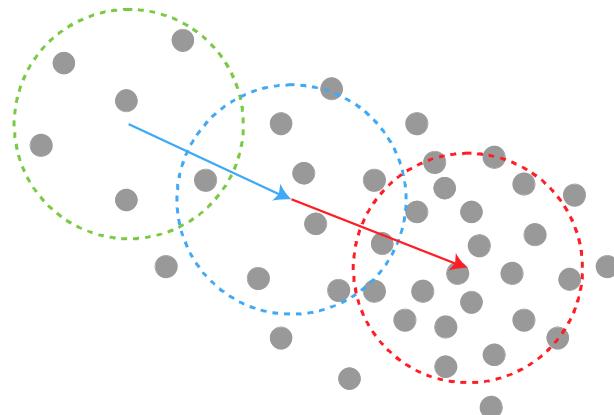


图 3.1 Mean Shift 算法主要思想：沿着密度上升方向移动

Mean Shift 聚类算法相比与 K-means 等聚类算法的最大优势是无需指定聚类簇的数目，算法可以自动给出聚类簇的数目；该特性能够很好地满足本文的实际聚类需求。本文的 Mean Shift 聚类算法如下：

- (1) 将所有叶片和花瓣的根点加入点集 MP 中，并设定聚类半径 r 与中心距离阈值 c ，设定初始簇个数 $i=0$ ；
- (2) 在点集 MP 中随机选择一个未标记过的点作为 $center$ ，更新 $i=i+1$ ；若 MP 中所有点均已被标记，则转到 (5)；
- (3) 遍历点集 MP ，若存在点 p 与 $center$ 的距离小于聚类半径 r ，则对 p 添加标记属于簇 i ，并将 p 加入点集 $TEMP$ 中；
- (4) 计算点集 $TEMP$ 的聚类中心，将该中心设置为新的 $center$ ；若新 $center$ 与旧 $center$ 的欧几里得距离小于 c ，则转到 (2)，否则转到 (3)；
- (5) 遍历点集 MP ，统计每个点 p 的各簇的标记次数，若点 p 被标记为 j 的次数最多，则将该点加入簇 j 中；
- (6) 返回簇数目 i 和各簇的点集。

通过 Mean Shift 算法，可以将对花瓣和叶片进行聚类，若其根点被聚为一类，则说明花瓣或叶片同属一朵花或一簇。根据植物学原理，一般花朵的花瓣数遵从斐波那契数列规律且一般大于等于 5，而叶片很少互相聚集^[24]，故将元素数目小于 5 的簇归类为叶片簇，大于等于 5 的归类为花瓣簇，即一朵花；由此完成了对花瓣和叶片的再分类以及同属于一朵花的花瓣的聚类，本文算法聚类结果如图 3.2 所示。

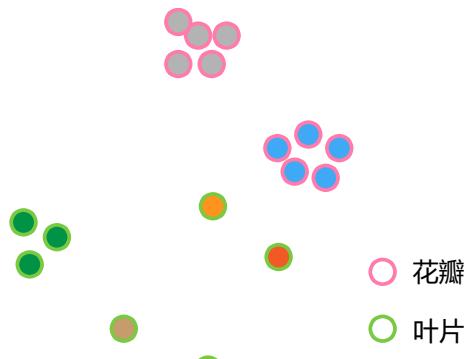


图 3.2 Mean Shift 聚类结果

3.4 枝干层级分析

在对枝干进行建模时，往往需要知道枝干的父子层级关系，这样在模型生成时可以通过这些信息调整枝干的半径、枝干的延伸偏向和姿态等参数，来增强模型的真实感。

在草图获取与预处理一节中已经得到了不同枝干的点集序列，本文设计了一个递归式的算法来对这些点集序列进行父子层级的关系构建；算法的主要思想是首先对点集进

行排序，排序的顺序以点集中 y 坐标最小值为依据进行升序排序；再对枝干点集进行匹配，匹配的依据是若枝干点集的端点与某一枝干节点点集内任意一点的欧几里得距离小于阈值，则当前枝干点集与这一枝干节点匹配，并为该点集创建子枝干节点；匹配时先与根节点进行匹配，若根节点无匹配则递归对子节点进行匹配；若某点集不与任何枝干节点匹配，则认定该点集为根节点枝干，为其创建相应的根节点；重复上述操作直到所有枝干点集均已处理完毕；算法的伪代码如下：

算法：枝干父子层级关系构建算法

输入： *branches* 枝干的集合， *attachThreshold* 父子枝干距离阈值
输出： *rootBranches* 枝干的关系树

```
1:function isBelongToNode(node, skeleton, attachThreshold)
2:    if node = nullptr then
3:        return false
4:    end if
5:    if nearestDistance(node.skeleton, skeleton) < attachThreshold then
6:        node.offshoots.addNode(skeleton)
7:        return true
8:    else
9:        for each n in node.offshoots
10:            if isBelongToNode(n, skeleton, attachThreshold)=true then
11:                return true
12:            end if
13:        end for
14:    end if
15:    return false
16:end function
17:
18:function branchConstruct(branches,attachThreshold)
19:    SortAsMaxYAscend(branches.begin,branches.end)
20:    for each branch in branches
21:        isRootBranch←true
22:        for each rootBranch in rootBranches
23:            if isBelongToNode(rootBranch,branch,attachThreshold)=true then
24:                isRootBranch←false
25:            break
```

```

26:           end if
27:       end for
28:       if isRootBranch=true then
29:           rootBranches.addNode(branch)
30:       end if
31:   end for
32:   return rootBranch
33:end function

```

通过该算法的处理，可以得到如图 3.3 所示的结果，在图中不同颜色的枝干代表着不同的父子层级，从图可知，本文算法较好地构建了枝干的父子层级关系，有利于后续建模操作的进行。

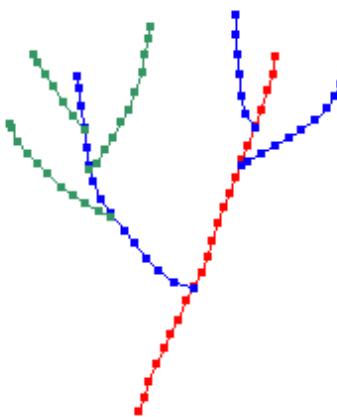


图 3.3 枝干的父子层级分析结果

3.5 轮廓填充

当创作者需要表示密集的叶片结构或者枝干结构时，如果逐个去绘制叶片或者枝干会使整个草图建模过程变得十分繁琐，本文在设计草图建模系统时充分考虑到了这一点，并允许创作者勾勒一个轮廓来代表大量叶片或者细枝干结构；因此在得到轮廓笔划之后，还需要对轮廓进行填充以满足创作者的创作企图，针对叶片和枝干结构的不同，本文将枝干轮廓和叶片轮廓进行分别处理，根据相应的轮廓来进行草图细节的丰富。

3.5.1 枝干轮廓填充

当创作者期望描述一个密集的细枝干结构时，本文系统允许创作者在待分枝的主枝干周围绘制一个轮廓笔划，代表轮廓内有许多细枝干从主枝干生成且形成与轮廓笔划相

符的枝干结构；为了完成这个目标，需要对轮廓进行填充，添加新的细枝干，本文参考了 Tan^[10]基于单张照片的树木建模工作中提出的数据驱动的吸引子算法，对其进行了适当的修改以适应本文的需求，提出了轮廓点吸引算法；算法的具体步骤如下：

- (1) 确定轮廓的重采样率 s , 轮廓吸引系数 k_p , 枝干吸引系数 k_b , 生长点间距 g , 生长速率 r , 最大生长次数 n , 拟合阈值 Th ;
- (2) 对枝干轮廓以采样率 s 进行重采样，获得轮廓点集 ProfilePoints;
- (3) 遍历在上一小节中构建的枝干层级节点，若某个枝干存在点在由轮廓点集围成的多边形内，则将该枝干加入生长枝干集合 GrowthBranch 中去；
- (4) 对 GrowthBranches 中的枝干以生长点间距 g 进行重采样，采样得到的点若在由轮廓点集围成的多边形内，则将点加入到生长点集合 GrowthPoints 中去，同时添加原枝干的方向向量（采样枝干根点到端点的方向向量）信息 v_b ;
- (5) 如果 GrowthPoint 已为空则转到 (7)，否则在生长点集合 GrowthPoints 中取一点 p , 设 $i=0$, 并建立一个枝干点集 NewBranch, 将 p 加入点集 NewBranch, 并将点 p 从 GrowthPoints 中除去;
- (6) 找到点 p 离轮廓点集 ProfilePoints 中最近的点记为 a , 计算向量 $\vec{v}_p = \frac{a-p}{|a-p|}$, 根据公式

$$p_{new} = r \cdot (k_p \cdot \vec{v}_p + k_b \cdot \vec{v}_b) \quad (3.3)$$

计算新的生长点 p_{new} , 并更新 $i=i+1$, 若 p_{new} 在由轮廓点集围成的多边形内且 $|a-p| > Th$ 且 $i < n$, 则将 p_{new} 加入 NewBranch 中并使 $p=p_{new}$; 否则将 NewBranch 加入到现有的枝干集合中去，并转到 (5)；

- (7) 对当前的枝干点集重新进行 3.4 节中的父子层级关系构建操作，算法结束。

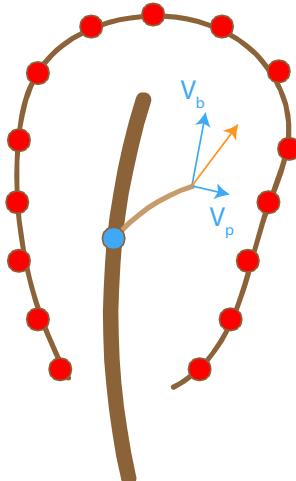


图 3.4 子枝干生长过程示意图

通过以上算法，可以设置不同的参数来达到不同的枝干轮廓填充效果，一般而言，当 k_p 较大时，新生成的子枝干倾向于顺着原父枝干的方向生长；当 k_b 较大时，新生成的子枝干倾向于向轮廓生长；当生长点间距 g 较小时新生成的细枝干会更加密集；当生长速率 r 较大时枝干会显得更平直，较小时会显得更弯曲；而最大生长次数 n 和拟合阈值 Th 则影响到新生成枝干的长度；各个参数对枝干的形态影响可由图 3.5 所示。

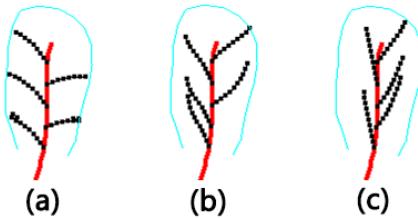


图 3.5 不同参数对枝干形态影响图。 (a) $k_p=0.5$, $k_b=0.5$ 时结果; (b) $k_p=0.6$, $k_b=0.4$ 时结果; (c) $k_p=0.7$, $k_b=0.3$ 时结果

3.5.2 叶片轮廓填充

当创作者需要在一个枝干周围布满叶片时，可以在枝干的周围绘制一个轮廓代表枝干上生长有若干叶片并形成与轮廓相符的形状。

叶片的轮廓填充算法与枝干填充算法类似，唯一的不同是在枝干填充算法最后一步中，叶片轮廓填充算法会在已经绘制的叶片中随机选择一个叶片，并对该叶片进行仿射变换，使得仿射变换后的叶片的根点与 NewBranch 的生长点对齐、叶片的尖点与 NewBranch 的末点对齐，并将所得的新叶片加入到现有的叶片集合中去，该过程如图 3.6 所示。

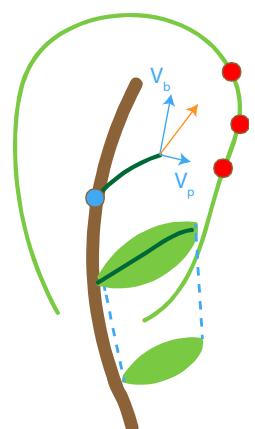


图 3.6 叶片生长和填充过程示意图

4 深度信息获取

基于草图的建模最关键的问题在于如何从二维图像中恢复出深度信息，只有二维信息是不足以建立出三维模型的。由于二维草图的深度信息是缺失的，用二维草图来表示三维模型具有歧义性，草图中的二维投影所对应的三维物体可以有无数种可能，如何寻找到一个合理的二维—三维物体映射是一个巨大的挑战，因此在三维模型的逆向推测的过程中需要结合一定的先验知识，以保证逆向推测得到的三维模型的合理性。本文以植物学知识为向导，结合大量真实样本的观察，提出一定的约束与假设来分别对花朵、枝干、叶片进行深度信息的恢复。

4.1 花朵深度信息获取

通过观察大量的花朵实物样本并参考 Yan^[25]相关工作，本文提出了一个圆锥模型，即认为一朵由若干花瓣组成的花是高度对称的，且花瓣环绕花心近似形成一个圆锥；圆锥的顶点位于花心处，花瓣大致地贴在圆锥侧面上，各个花瓣的尖点围成圆锥底面圆；该模型的示意图如图 4.1 所示。

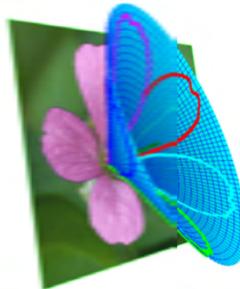


图 4.1 花朵的圆锥模型示意图

本文假设草图是由花朵实体经过垂直于绘图屏幕的方向的平行投影所得的，因此，通过分析草图中每朵花各个花瓣形成的圆锥投影可以恢复得到三维空间圆锥的方程，从而可以代入各个花瓣的 XY 二维坐标解得花瓣的 Z 轴深度信息。

本文的花朵深度信息恢复算法如下：

- (1) 在 3.3 章节中已经完成花瓣的聚类并得到 n 朵花的集合 FlowerSet，从 FlowerSet 取一朵花 F 进行深度信息恢复；
- (2) 计算花朵 F 中所有花瓣的根点的均值，记为花心 center；
- (3) 使用最小二乘法，对花朵 F 中所有花瓣的尖点进行椭圆拟合，得到拟合的椭圆 E ；

(4) 根据假设, 二维草图是通过垂直于绘图屏幕的方向的平行投影得到的, 因此椭圆 E 的长轴 b 即为原三维空间中圆锥底面半径 r; 圆锥的主轴与绘图平面所成的夹角 θ 有关系: $\sin \theta = \frac{a}{b}$, 其中 a 为椭圆 E 的短轴, b 为椭圆 E 的长轴, 根据此关系可求得 θ 。

(5) 计算草图平面上椭圆 E 到花心 center 的距离 l, 由于上一步已经求得 θ , 根据几何关系可知, 三维空间中的圆锥的高度 $h = \frac{l}{\sin \theta}$; 整合已知条件, 可以求得三维空间的椭圆方程。

(6) 上一步所得椭圆方程进行是以 Z 轴作为圆锥主轴的标准形式, 没有考虑到椭圆以 θ 角旋转。计算草图平面空间花心到椭圆 E 圆心 c 的向量 $\vec{v} = \frac{c - center}{|c - center|}$, 空间 Z 轴单位向量 $\vec{vz} = [0, 0, 1]$, 可计算得到椭圆的旋转轴 $\vec{A} = \vec{v} \times \vec{vz}$; 计算绕 \vec{A} 旋转 θ 得到的旋转矩阵 RM:

$$RM = \begin{bmatrix} A_x^2(1 - \cos \theta) + \cos \theta & A_x A_y(1 - \cos \theta) + A_z \sin \theta & A_x A_z(1 - \cos \theta) - A_y \sin \theta \\ A_x A_y(1 - \cos \theta) - A_z \sin \theta & A_y^2(1 - \cos \theta) + \cos \theta & A_z A_y(1 - \cos \theta) + A_x \sin \theta \\ A_x A_z(1 - \cos \theta) + A_y \sin \theta & A_y A_z(1 - \cos \theta) - A_x \sin \theta & A_z^2(1 - \cos \theta) + \cos \theta \end{bmatrix} \quad (4.1)$$

(7) 对花朵 F 中每个花瓣的每一点 p, 将坐标变换到以花心为圆点的坐标系坐标中, 得到点 $p' = [x', y']$; 设 p'' 为 p' 对应的三维空间点 $p'' = [x', y', z]$, 其中 z 为未知数; 联立下式,

$$\begin{cases} p_r = RM \cdot p'' \\ p_r = [x_r, y_r, z_r] \\ x_r^2 + y_r^2 = \frac{r^2 \cdot z_r^2}{h^2} \end{cases} \quad (4.2)$$

解出 z, z 即为点 p 所对应的深度信息。

该算法过程示意简图如图 4.2 所示。

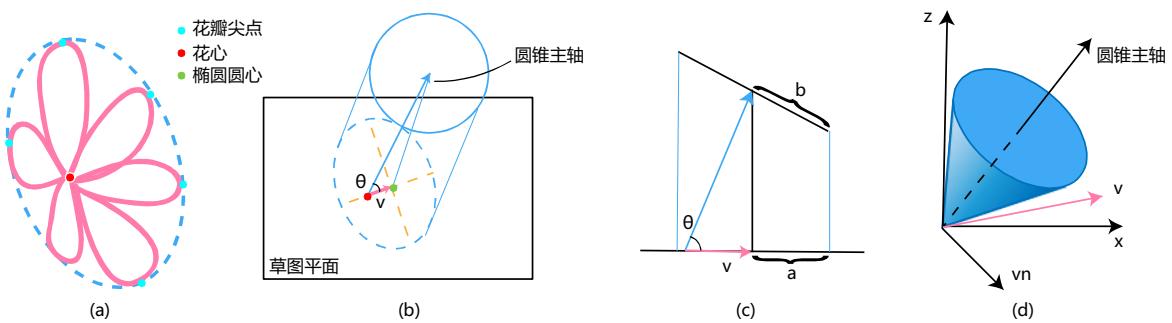


图 4.2 花朵的深度信息恢复算法示意图。 (a) 使用花瓣尖点拟合椭圆; (b) 草图平面椭圆与三维空间圆锥; (c) 圆锥在草图平面的横切面投影; (d) 三维空间中圆锥的主轴与旋转轴

4.2 枝干深度信息获取

枝干的几何结构较为简单，只需知道枝干两端点的深度信息便可通过插值得到枝干上各个点的深度信息。本文参考 Okabe^[15]的基于草图的树木建模工作，设计了一个获取枝干深度信息的算法；算法主要的思想是根据之前得到的枝干父子层级关系让子枝围绕父枝旋转，且尽可能地让子枝围着父枝最大角度分散开来，在三维空间中形成均匀的分布效果。

本文的算法是递归式的，考虑当前枝干作为父枝干，遍历其所有子枝干，将子枝干分成两组，左侧的子枝干和右侧的子枝干，让子枝干均匀地旋转分布在父枝干左右两侧 30° ~ 150° 的位置上，同时对每个子枝干的旋转角度添加一个服从高斯分布的随机扰动来增加真实感；值得注意的是，若不对子枝干进行左右区分，则可能计算得到一个较大的旋转角使得位于一个旋转半球内的枝干旋转到另一个旋转半球内，造成与原二维草图的表达意图相冲突；通过三维空间的旋转角度可以由二维的投影坐标反解出三维空间的 z 轴深度坐标，从而实现深度信息的获取，递归执行这个过程直到当前枝干无子枝干为止。算法的伪代码如下：

算法：枝干深度信息获取算法

输入： node 节点, axis 父轴, anchor 锚点, rotateAngle 旋转角, isRoot 是否是根节点

输出： 重建的三维点集，保存在 node 中

```

1:function constructSkeleton(node, axis, anchor, rotateAngle, isRoot)
2:   if isRoot=true then
3:     anchor←vec3(node.bottomPoint.x, node.bottomPoint.y, 0)
4:     top←vec3(node.topPoint.x, node.topPoint.y, 0)
5:   else
6:     top←vec3(node.topPoint.x, node.topPoint.y,
7:                 getZ(node.topPoint, axis, anchor, rotateAngle)
8:   end if
9:   dis←manhattanDistance(node.anchor2D, node.top2D)
10:  differ←top.z-anchor.z
11:  for each p in node.skeleton2D
12:    pDis←manhattanDistance(node.anchor2D, p)
13:    node.skeleton3D.addPoint(p.x, p.y, pDis/Dis*differ+anchor.z)
14:  end for
15:  leftBranchNum←0

```

```

16:   rightBranchNum ← 0
17:   for each branch in node.offshoots
18:     if isLeftBranch(branch)=true then
19:       leftBranches.addBranch(branch)
20:       leftBranchNum ← leftBranchNum + 1
21:     else
22:       rightBranches.addBranch(branch)
23:       rightBranchesNum ← rightBranchNum + 1
24:     end if
25:   end for
26:   leftAngle ← 30
27:   rightAngle ← 30
28:   leftUnitAngle ← 120/leftBranchNum
29:   rightUnitAngle ← 120/rightBranchNum
30:   axis ← normalize(top-anchor)
31:   for each b in node.offshoots
32:     if b in leftBranches then
33:       constructSkeleton(b,axis,b.anchor,leftAngle+random,false)
34:       leftAngle ← leftAngle + leftUnitAngle
35:     else
36:       constructSkeleton(b,axis,b.anchor,rightAngle+random,false)
37:       rightAngle ← rightAngle + rightUnitAngle
38:     end if
39:   end for
40:end function

```

根据算法所得的枝干深度信息重建得到的枝干模型如图 4.3 所示。

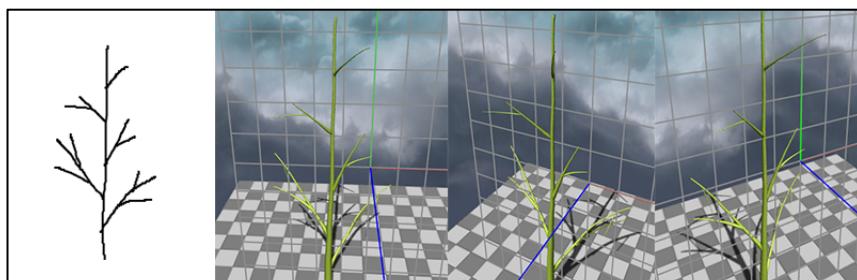


图 4.3 通过枝干深度信息重建得到的枝干模型

4.3 叶片深度信息获取

由于叶片的生长较为自由，相关工作中也鲜有对叶片草图建模的研究，这对叶片的深度信息获取提出了一定的挑战。本文通过研究一些植物样本观察到：叶片通常由枝干旁的小芽发育而来，由于叶片作为植物养分生产器官的特点，叶片的生长总是趋向于接触到最多阳光的方向生长，因此叶片大部分情况下总是位于一个垂直于阳光的平面上。

根据以上观测结果，本文提出了一个简单的叶片深度信息获取算法，算法首先认为所有的叶片初始状态均位于 XY 平面上，深度值 z 均为 0；取叶片的根点 p_r ，与由之前步骤得到的枝干节点进行匹配，匹配的依据是若某枝干点集里存在一点 p ，满足 $\sqrt{(p_r.x - p.x)^2 + (p_r.y - p.y)^2} < Th$ ，其中 Th 为距离阈值，则表明叶片与该枝干匹配，将叶片所有点的深度值 z 修改为 $p.z$ ；定义叶片的主轴 \vec{A} 为由根点指向尖点的向量，然后将草图平面的叶片以自身主轴旋转 90 度，此时叶片平面垂直于 XY 平面；计算主轴 \vec{A} 与 x 轴的夹角 θ ，将叶片绕-z 轴方向旋转 $\theta + \Delta$ ，其中 Δ 是一个在 $-30^\circ \sim 30^\circ$ 满足均匀分布的随机扰动；计算原二维草图中叶片主轴 \vec{A} 与枝干匹配点到枝干端点的向量 v_b 的夹角 Φ ；计算旋转角 $\theta_b = \Phi \cdot \pi^{-|\cos \Phi|} + \Delta$ ，其中 Δ 是一个在 $-30^\circ \sim 30^\circ$ 满足均匀分布的随机扰动；若在原二维草图中叶片位于枝干的左侧则将叶片绕+y 轴旋转 θ_b ，若在原二维草图中叶片位于枝干的右侧则将叶片绕+y 轴旋转 $-\theta_b$ ；此时叶片的位置即为三维空间中的最终位置，叶片各点变换得到的深度值即为最终深度值，算法过程的示意图如图 4.4 所示。

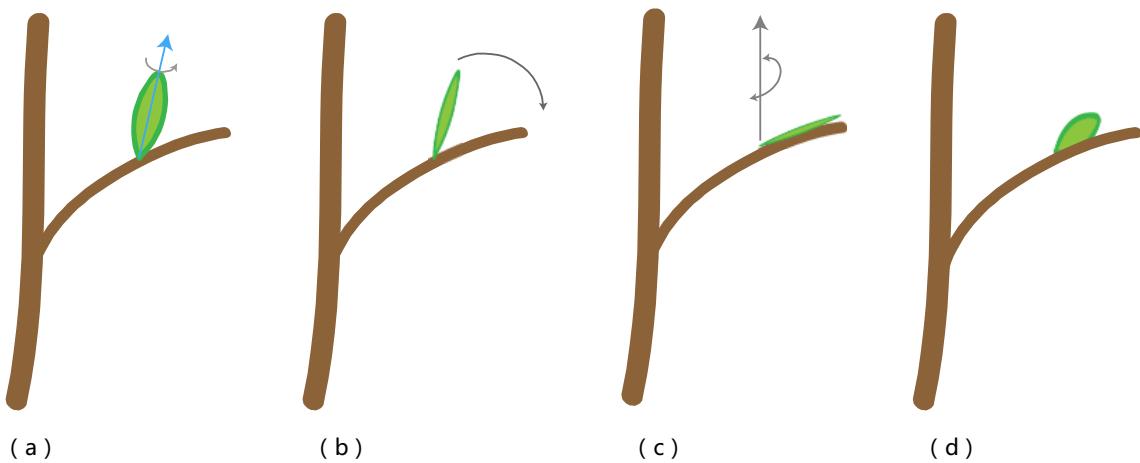


图 4.4 叶片深度信息获取示意图。 (a) 叶片绕自身主轴选择； (b) 叶片与 XZ 平面对齐；
(c) 叶片绕 Y 轴旋转； (d) 叶片最终所处状态

5 模型构建与优化

在获取到各个部分的深度信息之后，生成模型的条件均已具备，本节将详细说明如何根据已有信息对花朵、叶片和枝干进行模型构建，输出最终的三维模型。由于植物的结构较为复杂、曲面形状多变，在建模时应该将这些特点充分考虑并在模型中加以体现，否则产生的模型过于呆板、不生动、真实感较差。

5.1 花朵模型构建

5.1.1 曲面细分

在之前步骤中得到了花朵的轮廓的点集以及相应的空间坐标，但如果对这些点集直接进行三角剖分，所得的模型结构不仅因多边形数较少而显得十分粗糙而且过少的顶点数也难以模拟花瓣的各种复杂形态，因此十分有必要对花瓣进行曲面细分以增加顶点数。

本文中花朵顶点的增加沿着花瓣主轴方向以及垂直主轴的方向进行，首先以一定的移动间距在花瓣的主轴上取新顶点，同时在该新顶点位置处以垂直主轴的方向以一定间隔向两边增加顶点，直到新顶点超出花朵轮廓的范围。

具体算法如下：

(1) 确定主轴方向的顶点个数 N_a ，垂直主轴方向新增的最大顶点个数 N_p 以及垂直主轴方向的顶点个数 N_u 与长度 u 的函数 $N_u = f(u)$ ；

(2) 计算花瓣的主轴向量 $\vec{v}_a = \frac{p_t - p_r}{|p_t - p_r|}$ ，主轴长度 $L = |p_t - p_r|$ ，其中 p_r 为花瓣根点， p_t 为花瓣尖点；计算单位移动距离 $s_v = \frac{L}{N_a + 1}$ ，以花瓣根点为原点建立坐标系，主轴向量 v_a 记为 v 轴，垂直主轴向量 \vec{v}_a 向右方向记为 u，令 i 等于 1；

(3) 若 i 大于 N_a ，转到步骤 (7)；否则令点 $p = p_r + s_v \cdot \vec{v}_a \cdot i$ ，并增加顶点 p；

(4) 计算以点 p 为起点、方向为 u 轴正方向的射线与花瓣右轮廓的交点 e_r ，计算点 p 与 e_r 之间的距离 u_r ，通过函数 $f(u)$ 计算点 p 与点 e_r 之间的顶点个数 N_u ，在点 p 与点 e_r 之间均匀插入 N_u 个顶点；

(5) 与步骤 (4) 类似，在点 p 左侧插入若干顶点；

(6) 令 $i = i + 1$ ，转到步骤 (3)；

(7) 算法结束。

曲面细分的整个流程的示意图如图 5.1 所示，通过曲面细分花瓣在 UV 两个方向上均匀添加了若干顶点，花瓣的顶点变得更为密集，大大改善了三角剖分后模型的观感，同时也让点集能够进行后续弯曲和内凹的形变模拟，增加真实感。

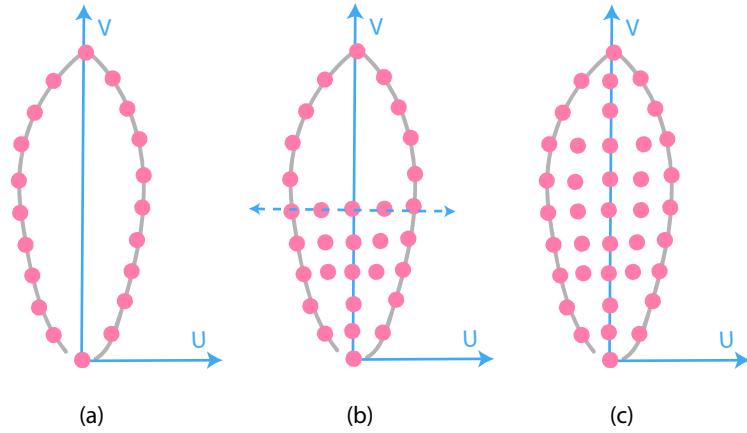


图 5.1 花瓣曲面细分算法示意图。 (a) 花瓣初始状态; (b) 在 u 轴和 v 轴方向上添加顶点; (c) 花瓣最终细分效果

5.1.2 花瓣弯曲模拟

花瓣的弯曲是各种类型的花朵中广泛存在的现象，花瓣的弯曲有的是受重力影响的结果，也有的是因花瓣本身固有的形态产生的，在建模的过程中如果能对此现象加以模拟可以大大提高模型的真实感。

本文认为花瓣是一个软体平面，满足弹性力学原理；根据力学的相关知识可以推出一个理想化的长度为 L 软体棒其在距离底端长度为 x 处的弯曲形变的距离为 y ， y 与 x 之间的关系有公式：

$$y = A \cdot \left(\frac{x}{L}\right)^4 + B \cdot \left(\frac{x}{L}\right)^2 \quad (5.1)$$

其中 A 、 B 为物体本身的材料弹性系数。

本文认为花瓣平面可以看做是一个质量均匀的弹性软体平面，可应用上述公式计算各点的弯曲偏移量，具体的算法如下：

(1) 在 4.1 小节中可得花瓣所在圆锥的主轴 $\overrightarrow{\text{Axis}}$ ，同时计算花瓣的主轴向量 $\overrightarrow{v_a} = \frac{p_t - p_r}{|p_t - p_r|}$ ，主轴长度 $L = |p_t - p_r|$ ；

(2) 对于花瓣中的一点 p ，计算向量 $\overrightarrow{v_p} = p - p_r$ ，计算 v_p 在同时计算花瓣的主轴向量 $\overrightarrow{v_a}$ 上投影 $x = \overrightarrow{v_p} \cdot \overrightarrow{v_a}$ ；

(3) 根据 (5.1) 式计算偏移量 y ，使点 p 沿着 $\overrightarrow{\text{Axis}}$ 方向移动偏移量 y ；

(4) 对花瓣每一点应用该算法；

该算法流程的示意图如图 5.2 所示。

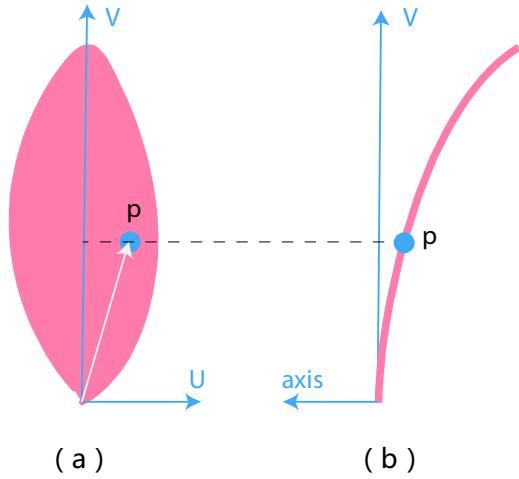


图 5.2 花瓣弯曲算法示意图。 (a) 花瓣任意一点 P 的位置; (b) 对应 P 点处的弯曲大小关系

在实际的建模中可以调整系数 A、B 以及对偏移量 y 乘上偏移系数 k 以达到不同的弯曲效果。本文算法实现的花瓣弯曲模拟效果如图 5.3 所示。

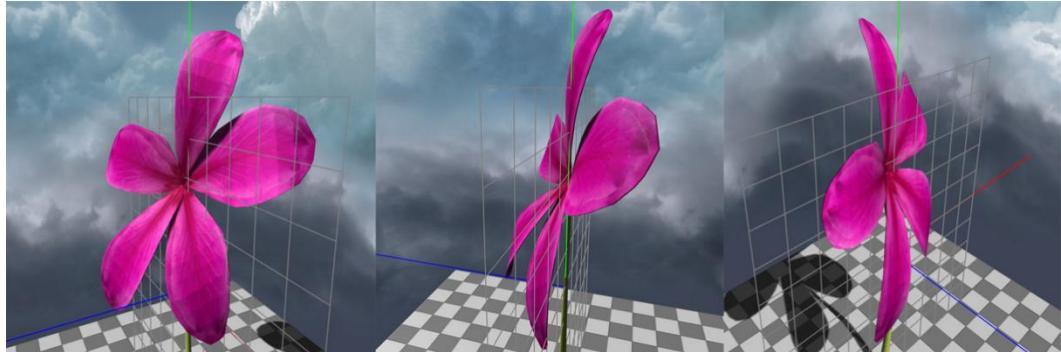


图 5.3 花瓣弯曲模拟结果

5.1.3 花瓣内凹模拟

花瓣的内凹也是一种在花瓣中广泛存在的现象，许多花瓣的内部产生弯曲凹陷，越靠近花瓣中心凹陷程度越大，从而产生优美的形状与曲线，给人赏心悦目的感觉。

为了对这种现象进行模拟，本文设计了一个算法，其主要思想是以花瓣根点为原点建立坐标系，计算花瓣的主轴向量 $\vec{v}_a = \frac{\vec{p}_t - \vec{p}_r}{|\vec{p}_t - \vec{p}_r|}$ ，主轴长度 $L = |\vec{p}_t - \vec{p}_r|$ ，将主轴向量 \vec{v}_a 记为 v 轴，垂直主轴向量 \vec{v}_a 向右方向记为 u；当 v 一定时，u 越接近 0，顶点的凹陷程度越大；u 一定时，v 越接近 $\frac{L}{2}$ ，顶点的凹陷程度越大。

具体算法流程如下：

(1) 确定 v 方向的凹陷函数 $y = f(v)$ 和 u 方向的凹陷函数 $y = g(u)$, 本文取:

$$f(v) = A \cdot \frac{v}{L} \cdot \left(1 - \frac{v}{L}\right) \quad (5.2)$$

$$g(u) = B \cdot \left(-\left(\frac{u}{W}\right)^2 + 1\right) \quad (5.3)$$

其中 A、B 为弯曲程度的控制系数, W 为点 p 所在 v 轴位置上, 沿着 u 轴方向距离花瓣边缘最近的距离;

(2) 从花瓣点集中取出一点 p, 若点集中点已经全部访问, 则转到步骤 (6);

(3) 计算点 p 在坐标轴 v 上的投影 $v_p = \overrightarrow{v_a} \cdot (p - p_r)$, 以及坐标轴 u 上的投影 $u_p = \sqrt{|p - p_r|^2 - v_p^2}$;

(4) 计算点 p 的偏移距离 $offset = f(v_p) \cdot g(u_p)$;

(5) 将 p 点沿着花瓣所属的花朵的圆锥主轴负方向移动距离 offset; 完成操作后转到步骤 (2);

(6) 算法结束。

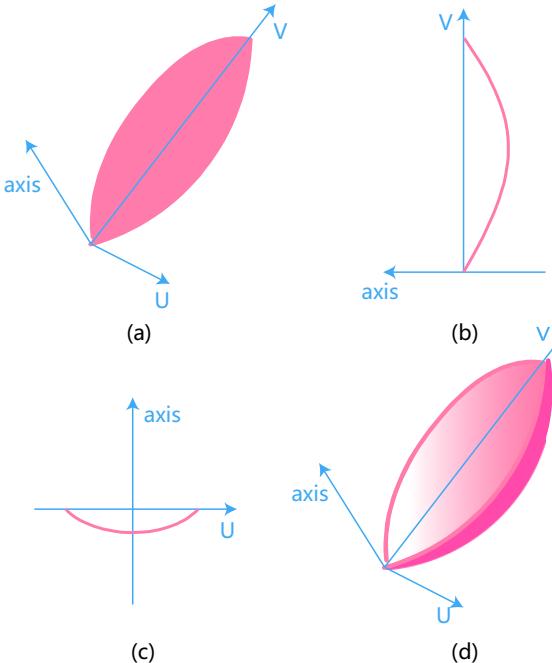


图 5.4 花瓣内凹算法示意图。 (a) 花瓣初始状态; (b) 花瓣 V 轴方向凹陷程度与 V 坐标值关系; (c) 花瓣 U 轴方向凹陷程度与 U 坐标值关系; (d) 花瓣最终凹陷效果示意图

算法的结果如图 5.5 所示, 可以看到通过本文算法很好地模拟了花瓣内凹陷的现象, 增加了模型的真实感与生动性。

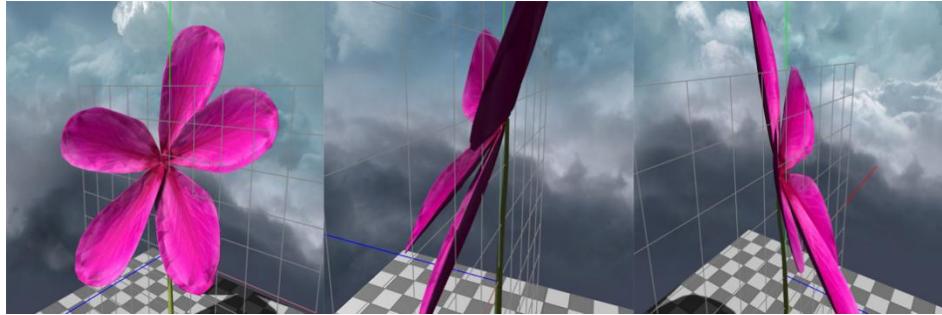


图 5.5 花瓣内凹模拟效果图

5.1.4 Delaunay 三角剖分

截止到目前，本文一直对花瓣的点集进行操作，这些离散的点集是不足以作为模型输出的，因此还需要将这些点进行连接，生成一系列的三角形集合，最终的结果应满足：

(1) 除了端点，平面图中不包含点集中的任何点；(2) 任意两条边均不相交；(3) 连接所得的所有面都是三角面，且三角面的集合是点集的凸包；这个过程被称为三角剖分。

在图形学的实际应用中运用得最广的是 Delaunay 三角剖分，Delaunay 三角剖分是由俄国数学家鲍里斯·德劳内提出的，除了以上三角剖分性质外，Delaunay 三角剖分还应该满足以下两个特性：

(1) 空圆特性：所得的三角面集合中任意一个三角面的外接圆内均不包含其他的顶点；

(2) 最大化最小角特性：点集所有可能的三角剖分结果里，只有 Delaunay 三角剖分的结果里所得的三角面的最小角最大。

由于以上两个特性约束，Delaunay 三角剖分可以保证得到的结果中的三角形均匀、美观，不会出现狭长的三角形；同时 Delaunay 的三角剖分结果唯一且具有较好的稳定性，修改一个顶点仅仅影响邻近的三角形，有利于后续模型的编辑。

Delaunay 三角剖分的方法有逐点插入法、三角网生长算法、分治法等^[26]。本文使用的是由 Guibas 和 Stolfi 于 1985 年提出的分治法^[27]，分治法是目前综合速度最快的 Delaunay 三角剖分算法。

本文的 Delaunay 三角剖分算法具体如下：

(1) 记给定的待三角剖分的点集为 S ，对点集 S 进行排序，以 x 轴坐标大小进行升序排序，当若干点 x 轴坐标值相同时，以 y 轴坐标大小进行升序排序；

(2) 将点集 S 均分为两部分 SL 、 SR ，对分开的两部分点集 SL 、 SR 再进行均分，如此递归分割直到每个集合的顶点个数为 2 个或者 3 个，如图 5.6 (a) 所示；

(3) 对所有点集进行线段连接, 如果点集中含有两个点则直接将这两点连接; 如果点集中含有三个点且三个点不共线, 则将这三个点两两连接; 如果点集中有三个点且三个点共线, 则将中间点与另外两点连接, 如图 5.6 (b) 所示;

(4) 将点集的子集与之前二均分的另一半子集进行递归合并, 为阐述方便, 记二均分时, 左侧子集为 L, 右侧子集为 R; 由左侧子集 L 的顶点连成的边记作 LL 边, 由右侧子集 R 的顶点连成的边记作 RR 边, 由一个属于左侧子集 L 中的顶点 L_p 和一个属于右侧子集 R 中的顶点 R_p 构成的边记为 LR 边, 如图 5.6 (c) 所示; 在 L 与 R 两集合中分别找一点, 连接成为 LR 边, 且这条 LR 边是位于最底端并且不与任何 LL 边、RR 边相交, 这条边记为 base LR 边, 如图 5.6 (d) 所示;

(5) 在子集 L 中找到与 base LR 边位于 L 中的端点所有邻接且与 LR 所成的逆时针角小于 180° 的点, 记为候选集合 candidate L, 并将 candidate L 中的点进行排序, 排序的依据是 candidate L 中的点 p 与 base LR 连接所成角度 $\angle p L_p R_p$ 进行升序排序, 如图 5.6 (e) 所示; 。

(6) 检查候选集合 candidate L 中的点的性质; 顺序遍历 candidate L 中点, 若点 p_i 与 L_p 、 R_p 的外接圆内包含 p_i 之后的点 $p_{i+1}, p_{i+2}, p_{i+3}$ 等, , 如图 5.6 (f) 所示, 则将 p_i 从 candidate L 中去除, 并删除与 base LR 边上顶点相连的边, , 如图 5.6 (g) 所示, 并继续检查下一个点 p_{i+1} 直到某个点 p_j 满足与 L_p 、 R_p 的外接圆不包含任何的后续点, 记点 p_j 为左侧最终候选点 final Candidate L;

(7) 对子集 R 镜像应用步骤 (5) (6) 得到右侧最终候选点 final Candidate R;

(8) 当 final Candidate L 与 L_p 、 R_p 的外接圆包含 final Candidate R 时, 选择 final Candidate R 作为最终连接点, 连接 final Candidate R 与 L_p 作为新的 base LR 边; 当 final Candidate R 与 L_p 、 R_p 的外接圆包含 final Candidate L 时, 选择 final Candidate L 作为最终连接点, 连接 final Candidate L 与 R_p 作为新的 base LR 边, 如图 5.6 (h) 所示; 当存在特殊情况 L_p 、 R_p 、final Candidate L、final Candidate R 四点共圆时, 可任意选择 final Candidate L 或 final Candidate R;

(9) 重复 (5) ~ (8) 直到 L 或 R 中找不到候选集合 candidate L 或 candidate R;

(10) 重复 (4) ~ (9) 直到所有子集都已合并;

(11) 所得连通图即为 Delaunay 三角剖分结果。

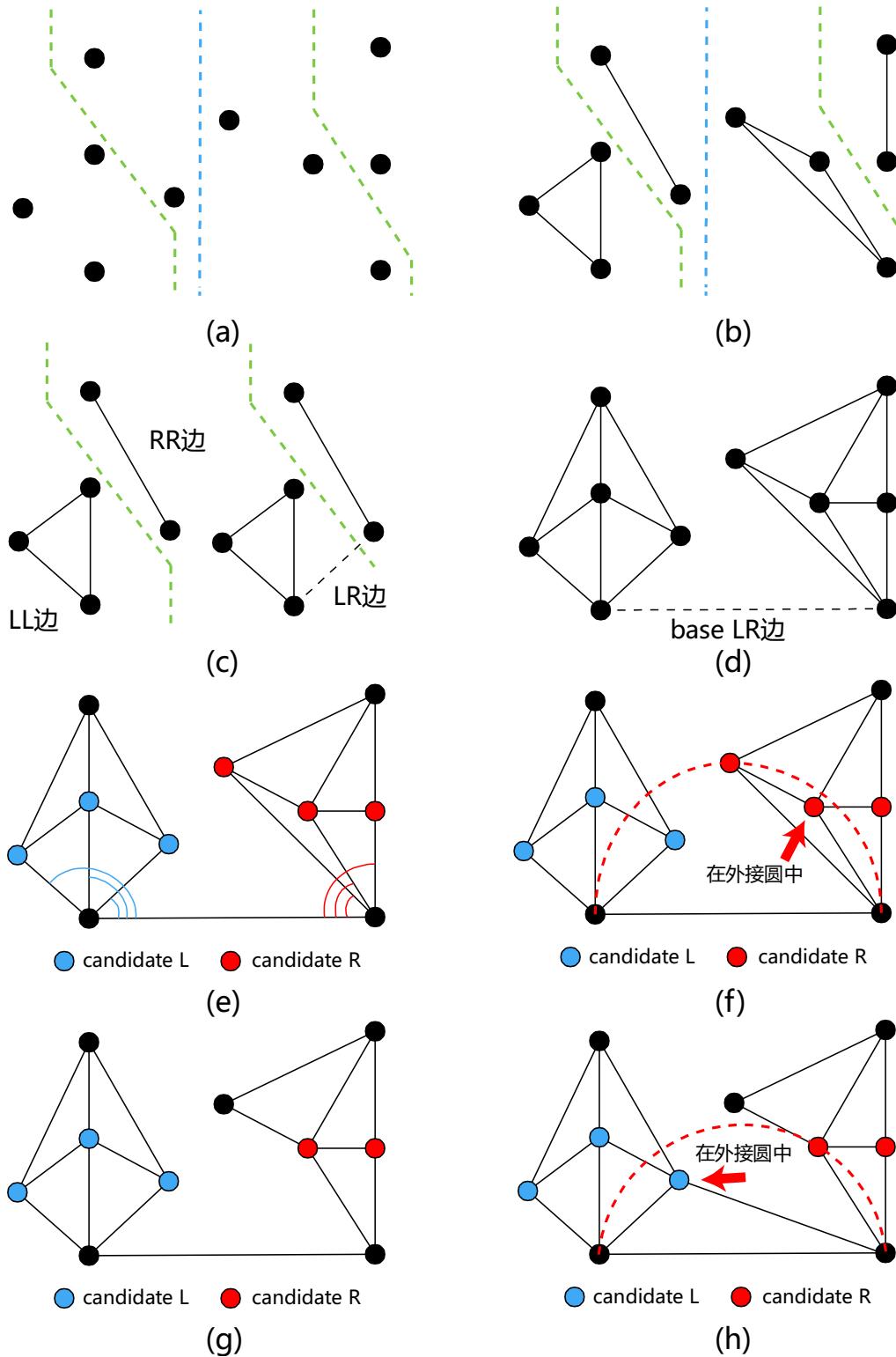


图 5.6 Delaunay 三角剖分示意图。 (a) 子集划分; (b) 子集点连接; (c) LL 边、RR 边与 LR 边; (d) base LR 边; (e) 候选点集; (f) 候选点的删除判定; (g) 删除不满足要求的候选点的边; (h) 选择满足条件的候选点连接成为新的 base LR 边

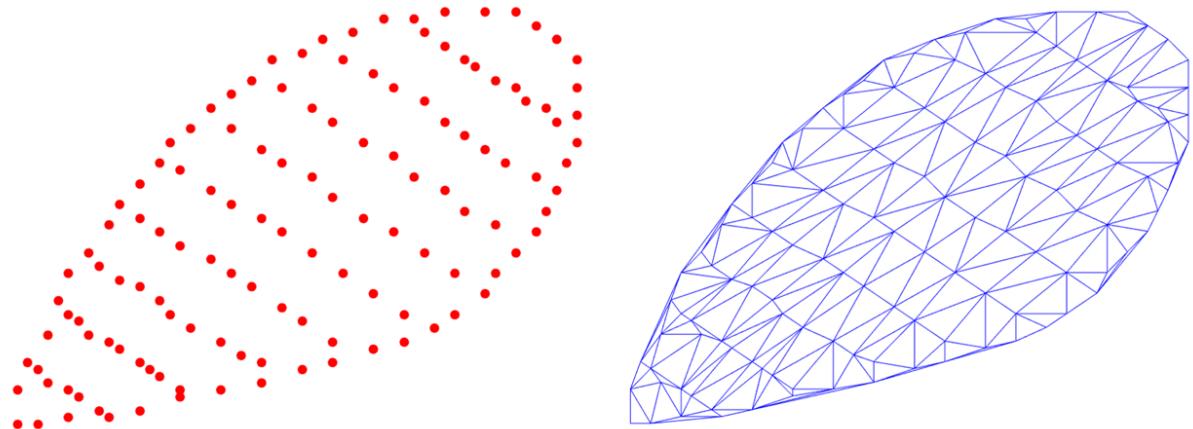


图 5.7 Delaunay 三角剖分结果

Delaunay 三角剖分的结果如图 5.7 所示，由图可知，经过 Delaunay 三角剖分，离散的点集被连接成了若干三角形集合，可以对其进行输出或者渲染。需要注意的是这样得到的花瓣模型仅仅是一个没有厚度的曲面，而现实世界的花瓣均是有一定厚度的，本文使用了一种常见的挤出方法来解决此问题，即对所得曲面沿着花瓣所在花朵的圆锥主轴负方向移动复制一份，并连接相应的轮廓顶点，这样即得到了一个封闭的、具有一定厚度的花瓣模型，如图 5.8 所示。

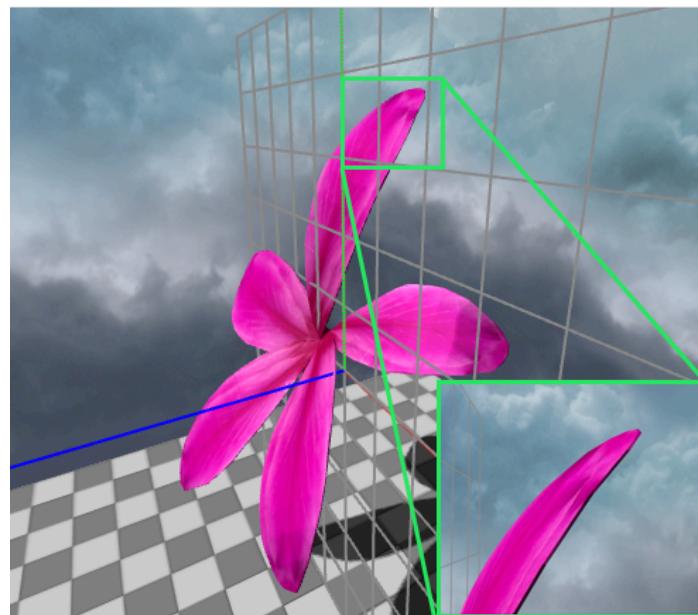


图 5.8 花瓣模型效果

5.2 叶片模型构建

叶片的结构与花瓣类似，事实上叶片可以看作是一种特殊的花瓣，在对单个的叶片建模可以很大程度上借用应用在花瓣建模上的方法。在对花瓣进行弯曲模拟、内凹模拟和曲面的挤出时均需要用到花瓣所在的花朵圆锥主轴向量，而叶片并没有相关的信息，在实际应用中，本文将叶片原始点集坐在平面的法向量作为主轴向量，这样便可以对叶片进行与花瓣同样的曲面细分、弯曲模拟、内凹模拟、Delaunay 三角剖分和挤出操作，建立出叶片网格模型。由于花瓣和叶片形态上的差异，在算法参数上需要做一些调整，以适应叶片的形态特点。

5.3 枝干模型构建

枝干是植物的重要组成部分，枝干模型的品质直接影响到整个植物模型的观感：本小节将介绍如何从代表枝干骨架的三维点集中建立出具有一定真实感的枝干模型。

5.3.1 各级枝干半径确定

在现实世界中可以经常观察到植物枝干的分叉后具有不同的半径，越末级的枝干半径越小，事实上植物枝干的分支遵从着一定的规律：假设每一级的枝干的半径在各处都是一致的，设 r 为分支点处以下的枝干半径， $r_1 \dots r_m$ 为分支点以上的各个枝干的半径，则满足

$$r^p = r_1^p + \dots + r_m^p \quad (5.4)$$

其中幂级数 p 控制着枝干半径积累程度，影响着下一级枝干的半径相对粗细，不同种类的植物有着不同的幂级数 p ；当 $p=2$ 时，即为达芬奇提出的树木分支枝干半径关系；对于花卉和一些草本植物，取 $p>2$ 可以达到较好的效果。

在 3.4 节中，本文构建了枝干间的父子层级关系，通过遍历枝干节点可以统计得到某个枝干的子节点和子节点个数，将相关信息带入公式 (5.4) 中可以的得到子枝干的半径并记录在节点中供后续建模参考。

5.3.2 枝干半径衰减



图 5.9 自然界植物枝干的半径衰减现象

通过上一小节的工作本文得到了各级枝干的半径，然而事实上植物的枝干的半径并不是恒定的，图 5.9 展示了一些植物的图片，可以观察到对于一级枝干，其半径从根部到顶端是逐渐衰减的，这是植物生长的先后顺序和生长累积效应造成的；为了捕捉这种现象，本文提出了一个半径衰减公式

$$r_a = \frac{s^{-\alpha x} + m}{1 + m} \quad (5.5)$$

其中 x 为枝干某点距离根部长度与枝干总长的比值， s 和 α 控制着衰减的快慢， s 一般取 $s > 10$ ， α 一般取 $0.3 \sim 0.5$ ， m 控制枝干末端半径占根部半径的比值，一般取 $0 \sim 1$ 。

5.3.3 枝干几何形体建模

对于枝干的几何形体建模目前已经有广泛的研究，常用的方法有多棱柱、B 样条曲线、广义圆柱等，基于建模和渲染的综合性能考虑，本文使用八棱柱来对枝干进行建模。

在之前的章节中已经得到了枝干骨架的三维点集、枝干半径和半径衰减公式，本文的枝干构建方法的主要思路是使用一个正八边形，将八边形中心沿着枝干骨架移动放样，同时调整八边形的半径；相邻两个骨架点之间的枝干可以通过连接中心位于这两骨架点的八边形相应的顶点，产生一节八棱柱来表示，重复此过程便可到整个枝干模型。

具体算法如下：

- (1) 对骨架点 p_i ，计算向量 $\vec{v}_1 = \frac{\vec{p}_{i-1} - \vec{p}_i}{|\vec{p}_{i-1} - \vec{p}_i|}$ 和 $\vec{v}_2 = \frac{\vec{p}_{i+1} - \vec{p}_i}{|\vec{p}_{i+1} - \vec{p}_i|}$ ，其中 p_{i-1} 为 p_i 的前一点， p_{i+1} 为 p_i 的后一点，并计算 \vec{v}_1 、 \vec{v}_2 的角平分线向量 $\vec{v}_3 = \frac{\vec{v}_1 + \vec{v}_2}{2}$ ；
- (2) 计算旋转轴 $\overrightarrow{axis} = (\vec{v}_2 \times \vec{v}_1) \times \vec{v}_3$ ；
- (3) 根据之前得到的根半径和半径衰减公式计算得到对骨架点 p_i 处多边形半径 r_i ，并计算前向量 $\overrightarrow{front} = (0, 0, r_i)$ ；
- (4) 令 $p_{i,1} = \overrightarrow{front} + p_i$ 得到八边形的第一个顶点，将 \overrightarrow{front} 绕 \overrightarrow{axis} 旋转 $\frac{2\pi}{8}$ ，令 $p_{i,2} = \overrightarrow{front} + p_i$ 得到八边形的第二个顶点，依次类推得到中心在 p_i 的八边形的所有八个点 $p_{i,1}, p_{i,2}, \dots, p_{i,8}$ ；
- (5) 对枝干骨架点集中所有点重复 (1) ~ (4) 过程，得到所有八边形；
- (6) 用三角形连接相邻的两八边形对应的顶点，更确切地说是对于两相邻八边形连接 $p_{i,j}, p_{i,(j+1) \bmod 8}, p_{i+1,j}$ 和 $p_{i+1,j}, p_{i+1,(j+1) \bmod 8}, p_{i,(j+1) \bmod 8}$ ，所有对应点连接完成之后所得的三角面集合即为枝干的网格模型。

该算法流程示意图如图 5.10 所示。

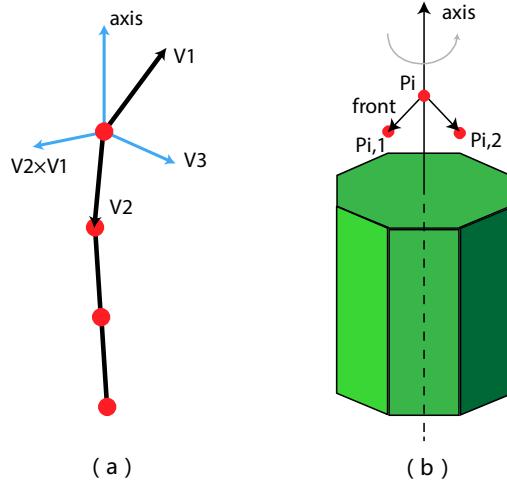


图 5.10 枝干建模算法示意图。 (a) 旋转轴的计算; (b) 八边形各点的生成

通过本文算法得到的枝干模型如图 5.11 所示, 值得注意的是, 本算法中使用两相邻骨架的角平分线向量 \vec{v}_3 , 来计算旋转轴 \overrightarrow{axis} , 使得八边形位于 \vec{v}_3 与 $\vec{v}_2 \times \vec{v}_1$ 所成平面上, 这样生成的枝干在各个骨架点上可以达到非常平滑的连接, 大大提升了模型的观感。

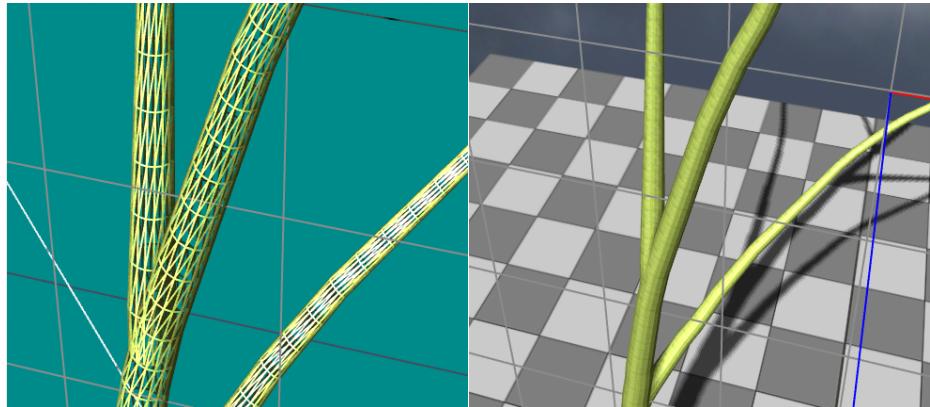


图 5.11 枝干模型结果图

6 模型的渲染与导出

6.1 模型渲染

模型建立之后，系统应当能够对其进行渲染来实时地向创作者展示建模效果，使得创作者能在模型导出之前发现模型存在的问题并进行相应的调整和修改。为了保证渲染的质量，系统应当还支持纹理映射、各种材质、各种光源、阴影等功能，这样能让创作者直观地观察到植物模型应用到电子游戏、虚拟现实场景中可能的显示效果。本文使用目前业界十分流行的 OpenGL 库和 C++ 语言编写了一个渲染框架，支持各种渲染功能，能够较好地向创作者实时地展现植物模型。

6.1.1 纹理与材质库

纹理与材质是影响渲染效果的重要因素之一，给模型赋予合适的纹理与材质能够极大地提升模型渲染的视觉效果。同时为了简化模型后期的处理工作，本系统应该能够在输出植物模型的同时将合适的纹理与材质也一并输出，这样导出的模型便可以直接应用于电子游戏等应用场景中。

由于本系统给予了创作者绘制植物草图很大的自由度，创作者能够绘制和创建出各种形态多样的各种种类的植物模型，如果对这些模型都赋予同一种纹理和模型显然是非常不合适的，在某些情况下可能反而劣化模型质量、产生古怪的视觉效果；因此有必要建立一个纹理和材质库，对于创作者绘制的植物模型，系统能够自动识别植物种类或是经由创作者手动选择，赋予合适的纹理和材质。

本文在互联网上搜集了一些常见的植物的图片，并进行后期处理制作成相应的纹理贴图，涵盖十几种植物种类，如图 6.1a 所示；对于材质，本文使用 Blin-Phong 光照模型中的材质参数来对模型的材质进行描述；本文参考了 Ogre 引擎文档提供的一些材质参数并参考现实世界中一些植物的图片建立了一个小型的材质库，一些材质的效果预览如图 6.1b 所示。

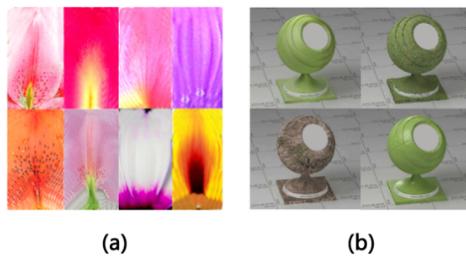


图 6.1 纹理与材质库。 (a) 本文纹理库的一部分； (b) 本文材质库的一部分

6.1.2 渲染模块

本文系统的渲染模块使用 OpenGL 作为底层的渲染应用程序编程接口(API)，OpenGL 是专门设计来用于渲染 2D、3D 图形的跨语言、跨平台 API。OpenGL 提供了非常丰富的函数，用来绘制从简单的图形到各种复杂的三维图形与场景。OpenGL 常用于 CAD、虚拟现实、数据可视化和电子游戏开发等。

OpenGL 可以工作在立即模式(immediate mode)或是核心模式(core-profile mode)；其中立即模式是 OpenGL 的早期工作模式，内置了一些常用的图形学绘图算法，用户可以通过简单的命令进行图形的绘制，但这种模式的绘图效率较低、局限性大，属于被标记为不建议使用的模式；核心模式是一种现代的工作模式，核心模式下则仅保留 OpenGL 核心的函数，能够给予使用者较大的自由度，能够最大限度提升渲染效率和渲染效果，但是此模式下提供的 API 都较为底层，需要使用者去实现常用的渲染算法。

本文生成的植物模型有着较为复杂的曲面结构、顶点数较多，本系统致力于提供创作者实时渲染观察模型的效果，因此本文选择 OpenGL 的核心模式进行植物模型的渲染。由于 OpenGL 核心模式下的 API 较为底层，本文编写了一个三维图形渲染框架，完成了对 OpenGL 的底层绘图 API 的封装并实现了常用的渲染功能如：摄像机、常见几何体的生成、常见光源、材质、Blin-Phong 光照模型、阴影、场景管理等。该三维图形渲染框架的整体结构如图 6.2 所示。

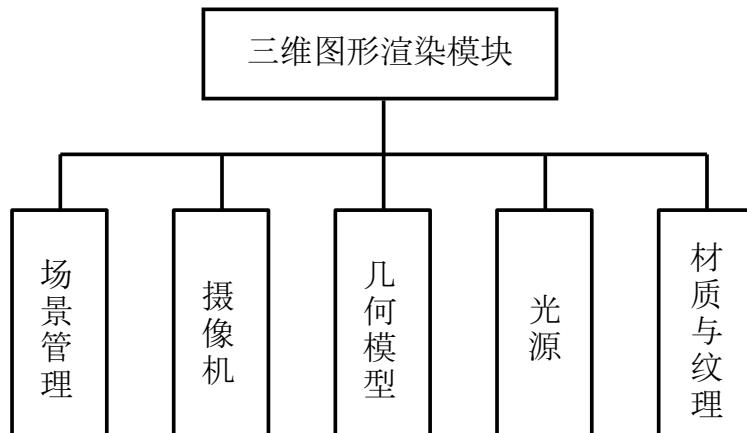


图 6.2 三维图形渲染模块结构图。

本文将该三维图形渲染框架整合进建模系统中去，可以完成植物模型的实时渲染，具体效果如图 6.3 所示，由图可知，本框架可以很好地对植物模型渲染，并有较为真实的渲染效果，有助于创作者实时地观察植物模型、预览建模效果，提升了交互式建模体验。

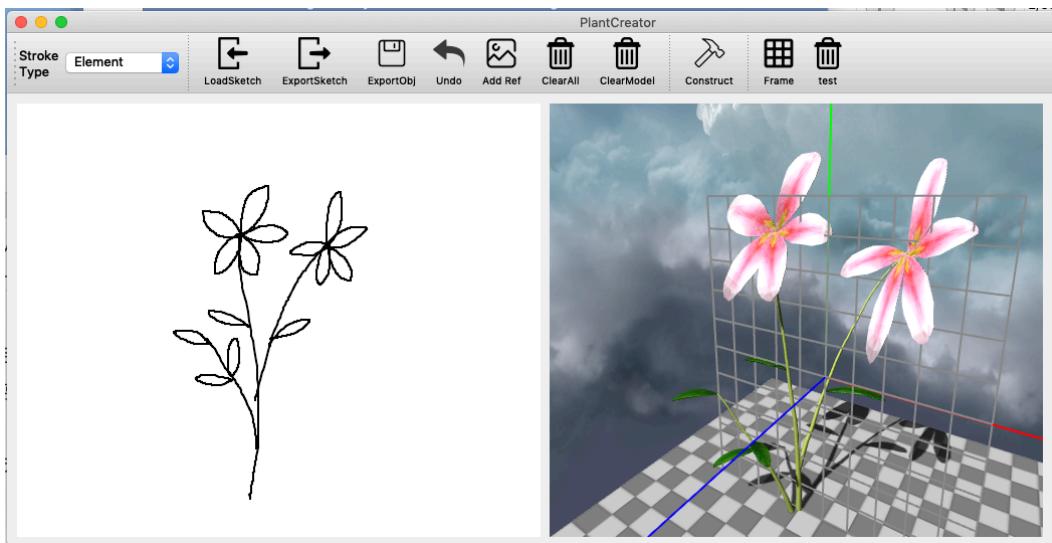


图 6.3 三维图形渲染模块渲染植物模型的效果

6.2 模型导出

为了使手绘草图建模系统具有实用性，系统还应具有将植物模型导出为常见三维模型格式的功能，这样建模所得的模型可以无缝应用于电子游戏开发、虚拟现实应用、影视特效等场景中去。OBJ 格式作为一种简单常用的格式，被大多数三维建模软件、游戏引擎所支持，因此本文选择将模型导出为 OBJ 格式。

6.2.1 OBJ 格式简述

OBJ 文件是 Alias Wavefront 公司开发的一种三维模型文件格式，目前几乎所有知名的三维软件都支持 OBJ 文件的读写。OBJ 文件是一种基于文本的三维模型格式，可以直接用文本编辑器打开 OBJ 文件进行查看和编辑，OBJ 文件模型描述本身的语法也较为简单，OBJ 格式还有相配套的 MTL 文件，用来描述模型的材质与纹理，非常适合本文系统的需求，因此导出为 OBJ 格式文件是比较实用和易行的。

OBJ 文件中常用的信息有顶点数据、法线数据、纹理坐标、三角面数据、组标签和材质信息等。模型中所有的顶点数据以 v 开头，一个空格之后紧接着是该顶点的 xyz 坐标，坐标之间空格隔开，一个顶点信息以回车符结束。法线数据以 vn 开头，一个空格之后紧接着是该法线的 xyz 坐标，坐标之间空格隔开，一个法线信息以回车符结束。纹理坐标以 vt 开头，同样地，一个空格之后紧接着是该法线的 uv 坐标，坐标之间空格隔开，一个纹理坐标信息以回车符结束。对于三角面信息，使用 f 开头作为标记，一个空格隔开，随后是三组顶点数据，顶点数据之间以空格隔开，一个面数据结束后回车符表

示结束；一组顶点数据的格式为“顶点索引/纹理坐标索引/法线索索引”，各个索引均以1开始计数。当一些三角面需要成组标记时，在三角面前一行加上“g 组标记”即表示标记之后的三角面均属于同一组。在三角面前一行加上“usemtl 材质标签”可表示之后的三角面均使用所标记的材质，具体的材质信息则储存在 MTL 文件中。

MTL 文件是与 OBJ 文件配套的材质描述文件，和 OBJ 文件一样，MTL 文件同样是基于文本的文件格式。MTL 的一种材质以“newmtl 材质标签”开头，表明以下参数都属于名为材质标签的材质。MTL 支持各种材质参数如环境光颜色、漫反射颜色、高光颜色、高光系数、漫反射贴图等。其中环境光颜色以标签 Ka 开头、漫反射颜色以标签 Kd 开头、高光颜色以标签 Ks 开头、高光系数以标签 Ns 开头、漫反射贴图以标签 map_Kd 开头，后面紧跟着具体参数值，每个参数以回车符结束。

6.2.2 导出 OBJ 模型

在本文实现的图形渲染框架中，储存所有的模型对象如叶片、花瓣、枝干的类均继承自 GeometryObject 父类中，父类定义了基本的顶点数据、法线数据、纹理坐标和三角面信息的储存结构，为了进行 OBJ 格式模型的导出，对 GeometryObject 父类添加导出模型数据的虚函数。本文在实现时声明了导出顶点数据的虚函数 exportVertices、导出法线数据的虚函数 exportNormals、导出纹理坐标的虚函数 exportUVs、导出三角面的函数 exportFaces 和导出材质的函数 exportMaterial，在子类中具体实现这些算法，来实现对各种对象模型的具体导出操作。

本文的模型导出算法是对所有模型对象进行四次遍历，每次遍历对该模型对象先后调用函数 exportVertices、exportNormals、exportUVs 和 exportFaces，在前三次遍历中每次调用相应函数结束后记录下每个对象的顶点数据、法线数据和纹理坐标的全局起始下标，在最后一次遍历调用 exportFaces 函数时，首先输出模型的名字与材质标签，再输出三角面信息；值得注意的是，exportFaces 函数输出的三角面中的顶点数据是以全局的下标表示的，而在模型对象的内部，三角面的顶点数据下标是以该对象内部的以0开始的局部下标表示的，因此需要使用之前记录的各个模型对象的全局起始下标加上对象内部的相对顶点下标计算出全局的顶点下标输出，这样所有模型的三角面信息均能准确地输出了。

MTL 材质文件的输出的算法是遍历所有模型对象，调用函数 exportMaterial，该函数将会输出材质标签和该材质相应的参数，最后如果创作者需要，将纹理库中使用到的纹理拷贝到 MTL 文件的相同目录下输出，方便创作者将模型导入到其他三维软件中。

本文系统构建出的模型与导出的模型效果如图 6.4 所示，可以看到，导出的模型与系统构建、渲染的模型效果基本一致，经测试导出的 OBJ 格式的模型可以使用 meshlab、3ds max 等常见建模软件正常打开，并且纹理与材质也能正常使用，达到了预期效果。

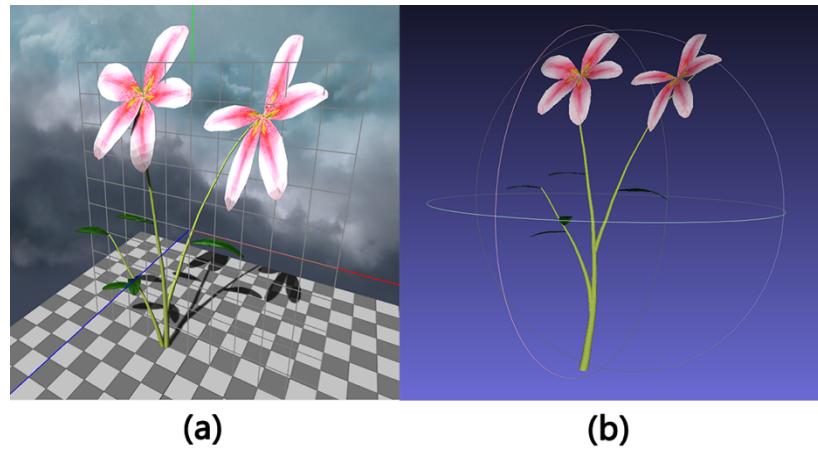


图 6.4 模型导出结果。 (a) 系统中渲染出的模型结果； (b) meshlab 中显示导出模型结果

7 结果与分析

7.1 建模效果

7.1.1 系统建模效果

本文最终实现的基于手绘草图的植物建模系统如图 7.1 所示，系统主要由绘图区和模型显示区域组成，创作者在左侧的绘图区域进行草图的绘制，绘制完毕后单击工具栏的构建按钮进行模型构建，若创作者对模型的属性有特殊要求可以在右侧参数面板手动修改相应算法的各项参数。

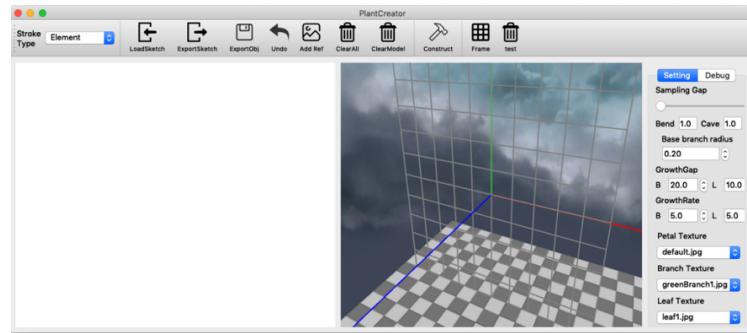


图 7.1 系统主界面截图

系统支持花瓣、叶片、枝干、细枝干轮廓、叶片轮廓这五种草图元素，创作者可以自由组合各种元素绘制出复杂的植物草图并建立相应的植物模型，本系统的各种草图元素的建模效果分别如图 7.2、图 7.3、图 7.4、图 7.5 所示；由图可知，本文系统能够完成对这些植物元素进行建模，在具有较逼真的自然形态同时还能够忠实于创作者的草图。

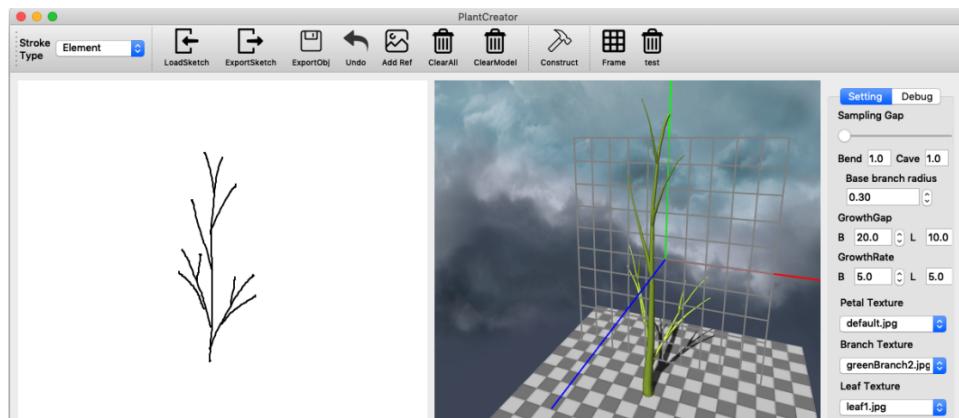


图 7.2 枝干草图建模效果

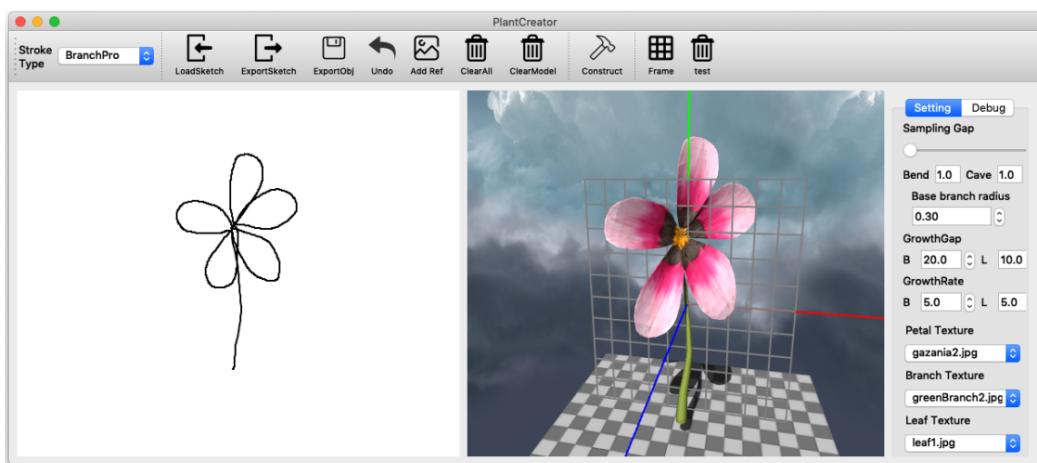


图 7.3 花朵草图建模效果

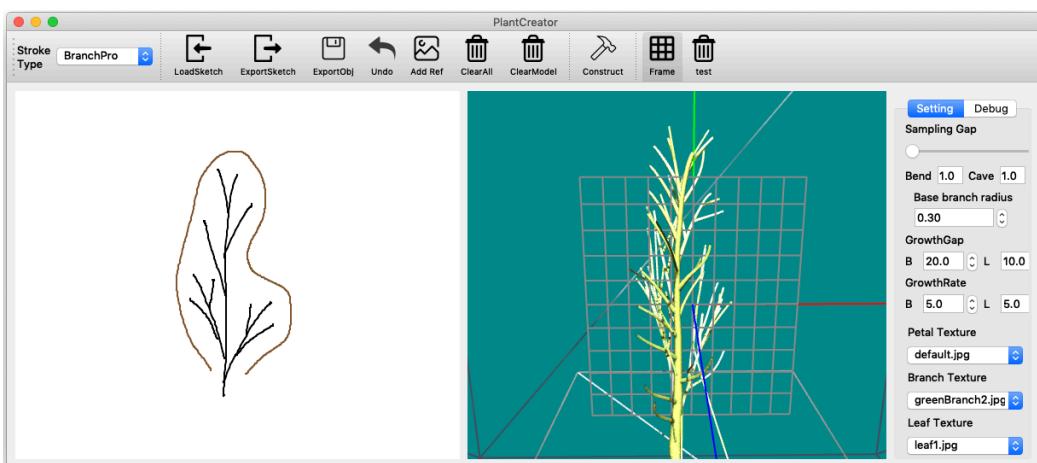


图 7.4 枝干轮廓草图建模效果

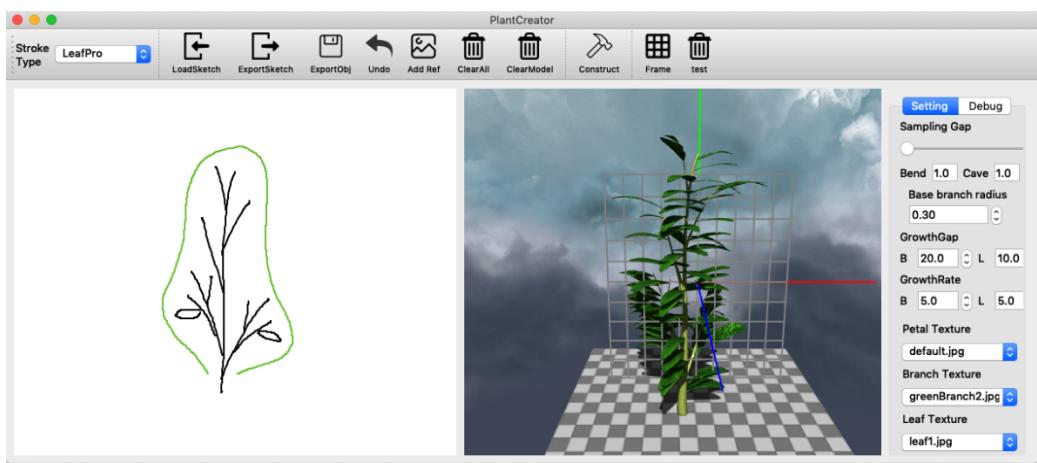


图 7.5 叶片轮廓填充草图建模

7.1.2 模型可视化效果

本文系统中的模型显示视图的渲染效果有限，不能充分将模型的形态特点展现出来，故本文将一些建模所得的模型导出为OBJ格式并将模型导入到KeyShot渲染器中进行渲染；KeyShot渲染器是较为专业的光线追踪渲染器，可以将模型的可视化效果较好地展现出来。本文搜集了一些常见种类的植物的实际图片，对它们进行草图临摹建模并将模型使用KeyShot渲染器渲染，得到的效果如图7.6、图7.7、图7.8、图7.9、图7.10、图7.11所示。通过图片对比可知本文所建立的模型能够较好地还原各类植物的形态特点，具有较强的形状表达能力，充分体现了基于手绘草图的建模方式的优势；同时本文建立的纹理和材质库涵盖了这些常见的植物种类，这些纹理和材质增强了模型的真实感，使得模型的视觉效果与实际植物图片较为一致。



图7.6 实际雏菊图片与本系统生成模型渲染图



图7.7 实际茉莉图片与本系统生成模型渲染图



图 7.8 实际锦葵图片与本系统生成模型渲染图



图 7.9 实际桔梗图片与本系统生成模型渲染图



图 7.10 实际缅栀花图片与本系统生成模型渲染图

7.2 建模性能与质量



图 7.11 测试模型集

本文在一台配置为 8GB DDR3 内存、2.4GHz 主频双核 CPU 的电脑上对系统的建模算法进行测试，测试中建立 8 个不同种类和规模的植物模型的所需时间，测试的草图和模型如图 7.11 所示，这 8 中植物模型的属性和建模时间如表 7.1 所示，由表中的数据可知，对不同规模和种类的植物模型本文系统均能够在 1 秒之内构建完成，具有较高的效率；横向对比其他学者的建模系统，Okabe^[15]等人的系统需要 10 分钟生成一棵树木模型，马玲^[28]等人的系统需要 20~40 秒的时间生成模型，可知本文系统的建模速度有着较大的优势。

表 7.1 不同植物模型的指标

模型名称	顶点数	三角面数	模型构建时间/ms	模型大小/kB
雏菊	4134	8942	209	803
红色雪割草	3820	7958	186	713
蓝色雪割草	2524	5192	148	458
茉莉	3422	7072	167	628
锦葵	2204	5294	148	464
桔梗	4528	9540	230	853
缅栀	2526	5180	154	456
酢浆草	3418	6876	173	623

在模型的质量方面，同样使用上述 8 个模型进行评估，各个模型的顶点数量、三角面数量和 OBJ 格式模型大小如表 7.1 所示，从表中数据可知本文生成的模型文件体积较小，没有过多的数据冗余。同时本文还对 OBJ 模型进行兼容性测试，在测试中本文系统

导出的 OBJ 模型均能被 3ds max、Maya、Cinema 4D、Blender 等主流建模软件正常读取打开、正常显示且附带纹理和材质信息。本文还对模型使用著名模型处理软件 Meshlab 进行诊断，所有模型均不存在网格破损、法线方向不统一、重复三角面、退化三角形等错误，说明模型具有较好的质量。

结 论

本文的主要工作是实现了一个基于草图的快速植物建模系统，创作者可以利用该系统自由绘制包含叶片、花瓣和枝干结构的各种植物模型的草图，系统能够获取创作者绘制的草图，进行预处理和特征分析并推测出创作者的建模意图，结合植物学的先验知识，对二维草图进行深度信息的恢复，得到三维点集数据，接着系统根据这些三维点集数据以植物学知识为导向生成具有逼真效果的三维植物模型，并实时渲染呈现给创作者；同时系统还能够将三维植物模型导出为 OBJ 格式，可以无缝应用于各游戏引擎和三维建模软件中，具有一定的实际应用价值。

本文的主要贡献和创新之处是设计和实现了一个新颖的基于草图的快速植物建模系统，在以往的植物草图建模研究中，许多学者主要专注于树木的相关研究，关于花朵和草本植物的研究较少，本文提出的基于草图的植物建模系统不仅能够进行树木的建模，同时也适用于花朵、草本植物等小型植物；横向对比已有的交互式花朵建模系统，本文系统能够给予创作者较大限度的自由度，创作者仅仅需要绘制一幅二维草图即可，无需或者只需很少的参数设置，创作速度快、效率高，在几秒之内便可生成具有一定真实感的植物模型。

本系统还存在一些不足，如支持的植物元素较少，仅仅支持花瓣、叶片和枝干这三种元素，后续可以增加例如根、花蕊、果实等元素；由于先验知识模型的限制，本文系统生成的模型姿态较为单一，在后续的研究中可以增加更多的模型支持，并结合统计和概率方法生成各种多姿多态的植物模型；近年来植物建模研究的一大趋势是将传统的植物建模方法与机器学习方法和深度学习方法进行结合，在本文的后续研究中可以考虑结合这两种方法辅助深度信息的恢复、植物种类的识别和分类、纹理的补充等。

参 考 文 献

- [1] 杨垠晖, 王锐. 树木的真实感建模与绘制综述[J]. 计算机辅助设计与图形学学报, 2018, 30(02):191–216.
- [2] Fisher J B, Honda H. Computer simulation of branching pattern and geometry in Terminalia (Combretaceae), a tropical tree[J]. Botanical Gazette, 1977, 138(4): 377–384.
- [3] Lindenmayer A. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs[J]. Journal of Theoretical Biology (S0022 -5193), 1968, 18(3):280–299.
- [4] Grzegorz R, Arto S. Mathematical Theory of L Systems[M]. Orlando: Academic Press, Inc., 1980.
- [5] Prusinkiewicz P. Applications of L-systems to computer imagery[C]//Proceedings of the 3rd International Workshop on Graph-Grammars and Their Applications to Computer Science. London: Springer, 1987: 534–548.
- [6] Prusinkiewicz P, James M, Měch R. Synthetic topiary[C]//Proceedings of the 21st annual conference on Computer graphics and interactive technique. New York: ACM Press, 1994:351–358.
- [7] Runions A, Lane B, Prusinkiewicz P. Modeling Trees with a Space Colonization Algorithm[C]//Proceedings of the 3rd Eurographics Conference on Natural Phenomena. Aire-la-Ville: Eurographics Association Press, 2007: 63–70.
- [8] Palubicki W, Horel K, Longay S, et al. Self-organizing tree models for image synthesis[J]. ACM Transactions on Graphics, 2009, 28(3):58.
- [9] Shlyakhter I, Rozenoer M, Dorsey J, et al. Reconstructing 3D tree models from instrumented photographs[J]. IEEE Computer Graphics and Applications, 2001, 21(3):53–61.
- [10] Tan P, Fang T, Xiao J, Zhao P, et al. Single image tree modeling[J]. ACM Transactions on Graphics, 2008, 27(5):108.
- [11] Yan, F, Gong M, Cohen - Or D, Deussen O, et al. Flower reconstruction from a single photo[J]. Computer Graphics Forum, 2014, 33(2):439–447.
- [12] Xu H, Gossett N, Chen B. Knowledge and heuristic-based modeling of laser-scan trees[J]. ACM Transactions on Graphics, 2007, 26(4):19.
- [13] Livny Y, Pirk S, Cheng Z, et al. Texture-lobes for tree modelling[J]. ACM Transactions on Graphics, 2011, 30(4):53.
- [14] Igarashi T, Matsuoka S, Tanaka H, et al. Teddy: a sketching interface for 3D freeform design[C]//ACM siggraph 2007 courses. San Diego: ACM Press, 2007:21.

- [15] Okabe M, Owada S, Igarashi T. Interactive design of botanical trees using freehand sketches and example-based editing[J]. Computer Graphics Forum, 2005, 24(3):487-496.
- [16] Anastacio F, Prusinkiewicz P, Sousa M C. Sketch-based parameterization of L-systems using illustration-inspired construction lines and depth modulation[J]. Computers & Graphics, 2009, 33(4):440-451.
- [17] Ijiri T, Owada S, Igarashi T. The sketch l-system: Global control of tree modeling using free-form strokes[C]//International Symposium on Smart Graphics. Berlin, Heidelberg:Springer, 2006:138-146.
- [18] Longay S, Runions A, Boudon F, et al. Treesketch: interactive procedural modeling of trees on a tablet[C]//Proceedings of the international symposium on sketch-based interfaces and modeling. Annecy: Eurographics Association, 2012: 107-120.
- [19] Ijiri T, Owada S, Okabe M, et al. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints[J]. ACM Transactions on Graphics, 2005, 24(3):720-726.
- [20] 唐棣,雷蕾,韩丽.素描式草绘的三维花朵建模[J].计算机工程与应用,2008(33):170-173+193.
- [21] 濮群,曾兰玲,张建明.基于手绘花瓣的花朵建模方法[J].计算机应用研究,2012,29(05):1959-1962.
- [22] Fukunaga K, Hostetler L. The estimation of the gradient of a density function, with applications in pattern recognition[J]. IEEE Transactions on information theory, 1975, 21(1):32-40.
- [23] Cheng Y. Mean shift, mode seeking, and clustering[J]. IEEE transactions on pattern analysis and machine intelligence, 1995, 17(8):790-799.
- [24] Prusinkiewicz P, Lindenmayer A. The algorithmic beauty of plants[M]. Berlin: Springer Science & Business Media, 2012.
- [25] Yan, F, Gong M, Cohen - Or D, Deussen O, et al. Flower reconstruction from a single photo[J]. Computer Graphics Forum, 2014, 33(2):439-447.
- [26] 余杰,吕品,郑昌文.Delaunay三角网构建方法比较研究[J].中国图象图形学报,2010,15(08):1158-1167.
- [27] Guibas L, Stolfi J. Primitives for the manipulation of general subdivisions and the computation of Voronoi[J]. ACM Transactions on Graphics, 1985, 4(2):74-123.
- [28] 马玲,李书杰,刘晓平.基于简单手绘的树木快速建模[J].系统仿真学报,2017,29(08):1667-1676.

修改记录

第一次修改记录：

本人于 2019 年 1 月申请到中国科学院深圳先进技术研究院做毕业设计（论文），指导教师为：程章林副研究员，校内指导教师为：徐喜荣副教授。2019 年 5 月 22 日回到学校。

第二次修改记录：

原第二章“草图获取、预处理与分析”内容过多。

修改前：原第二章“草图获取、预处理与分析”为一整体。

修改后：将“草图分析”部分从第二章中分离出来单独成为第三章。

第三次修改记录：

为三角剖分算法过程添加算法示意图图 5.6。

修改前：三角剖分算法无示意图。

修改后：添加图 5.6。

第四次修改记录：

文章缺乏系统截图等佐证材料。

修改前：无系统截图。

修改后：在附录 A 中添加若干系统截图。

第五次修改记录：

文章缺少一个总结性结果分析部分。

修改前：附录 A 中有部分结果图片。

修改后：增加“结果与分析”一章，将附录 A 部分移动到该章节去，删去附录 A。

记录人（签字）：

指导教师（签字）：

致 谢

正如牛顿所说“我能看得更远是因为我站在巨人的肩膀上”，任何的科学研究都离不开前人工作的基础以及他人的帮助，同样地，没有他人的帮助，这份毕业论文可能无法顺利完成，在此对那些在完成毕业设计过程中给予我帮助和启发的人表示感谢。

首先感谢我的指导老师徐喜荣副教授，徐喜荣老师不仅在整个毕设过程中都给予了我各种支持与帮助，而且在我大二大三时便引导我参加各种科研训练，在此过程中我积累了许多科研经验，这些经验无不受用终身。

感谢中国科学院深圳先进技术研究院的程章林副研究员，本毕业设计主要是在中国科学院深圳先进技术研究院完成的，感谢程章林副研究员的大力支持，提供了良好的工作环境和硬件支持，在毕设的选题、系统设计、算法实现上提供许多富有见解的建议与指导。

感谢养育我的父母，你们是我永远的后盾，给予我支持。感谢各位陪我度过快乐的本科四年同学朋友们，生活因你们而精彩。

最后感谢答辩组的老师、专家们，感谢你们的耐心审阅和专业的意见。