



同濟大學  
TONGJI UNIVERSITY

## 工学硕士学位论文

# 你的标题

(国家自然科学基金 (No.XXXXXXXX) 支持)

姓 名：你的姓名

学 号：10XXXXXXXXXX

所在院系：电子与信息工程学院

学科门类：计算机科学与技术

学科专业：计算机应用技术

指导教师：你的教授

二〇一三年五月





同濟大學  
TONGJI UNIVERSITY

A dissertation submitted to  
Tongji University in conformity with the requirements for  
the degree of Master of Science

## Your title

(Supported by the Natural Science Foundation of China for  
Grant No.XXXXXXXX)

Candidate: Tongji Ren  
Student Number: 10XXXXXXXXX  
School/Department: School of Electrical and Informational Engineering  
Discipline: Computer Science and Technology  
Major: Computer Application Technology  
Supervisor: Prof. XXXXXX

May, 2013



# 学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以盈利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：

年   月   日



# 同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：

年      月      日



## 摘 要

论文的摘要是对论文研究内容和成果的高度概括。摘要应对论文所研究的问题及其研究目的进行描述，对研究方法和过程进行简单介绍，对研究成果和所得结论进行概括。摘要应具有独立性和自明性，其内容应包含与论文全文同等量的主要信息。使读者即使不阅读全文，通过摘要就能了解论文的总体内容和主要成果。

论文摘要的书写应力求精确、简明。切忌写成对论文书写内容进行提要的形式，尤其要避免“第1章……；第2章……；……”这种或类似的陈述方式。

本文介绍同济大学论文模板 TONGJITHESIS 的使用方法。本模板符合学校的硕士、博士论文格式要求。

本文的创新点主要有：

- 用例子来解释模板的使用方法；
- 用废话来填充无关紧要的部分；
- 一边学习摸索一边编写新代码。

关键词是为了文献标引工作、用以表示全文主要内容信息的单词或术语。关键词不超过5个，每个关键词中间用分号分隔。（模板作者注：关键词分隔符不用考虑，模板会自动处理。英文关键词同理。）

**关键词：**T<sub>E</sub>X, L<sub>A</sub>T<sub>E</sub>X, CJK, 模板, 论文



## Abstract

An abstract of a dissertation is a summary and extraction of research work and contributions. Included in an abstract should be description of research topic and research objective, brief introduction to methodology and research process, and summarization of conclusion and contributions of the research. An abstract should be characterized by independence and clarity and carry identical information with the dissertation. It should be such that the general idea and major contributions of the dissertation are conveyed without reading the dissertation.

An abstract should be concise and to the point. It is a misunderstanding to make an abstract an outline of the dissertation and words “the first chapter”, “the second chapter” and the like should be avoided in the abstract.

Key words are terms used in a dissertation for indexing, reflecting core information of the dissertation. An abstract may contain a maximum of 5 key words, with semicolons used in between to separate one another.

**Key words:**  $\text{\TeX}$ ,  $\text{\LaTeX}$ , CJK, template, thesis

## 目录

第 1 章 引言 .....	1
1.1 背景介绍 .....	1
1.2 课题的主要工作 .....	1
第 2 章 树木建模的研究综述 .....	3
2.1 虚拟树木建模技术 .....	3
2.2 现实树木重建技术 .....	4
2.3 本章小节 .....	5
第 3 章 基于图像的树木轻量化3D建模方法 .....	6
3.1 技术路线 .....	6
3.1.1 基于改进的PyrLK光流法的图像特征匹配 .....	6
3.1.2 三维重建 .....	6
3.1.3 基于三维体素泛洪与线性拟合的三维树木骨架抽取 .....	7
3.1.4 基于用户交互的模型改善与轻量化 .....	7
3.1.5 建模质量评估 .....	7
3.2 技术路线图 .....	7
第 4 章 基于图像树木轻量化建模的若干算法 .....	8
4.1 基于改进的PyrLK光流法的特征点匹配方法 .....	8
4.1.1 光流法简介 .....	8
4.1.2 PyrLK光流算法 .....	8
4.1.3 改进的PyrLK光流算法 .....	10
4.1.3.1 加入放射变换 .....	10
4.1.3.2 提高鲁棒性 .....	11
4.2 基于三维体素泛洪与线性拟合的三维树木骨架抽取 .....	11
4.2.1 三维体素模型 .....	13
4.2.2 三维体素泛洪确定邻域范围 .....	14
4.2.3 通过最小二乘法线性拟合确定分支 .....	14
4.2.4 获取骨架半径 .....	16

---

4.3 基于枝干合并的轻量化处理 .....	17
4.3.1 L-System的尝试 .....	18
4.3.1.1 L-System简介 .....	18
4.3.1.2 树木模型的参数化L-System规则抽取 .....	18
4.3.1.3 使用L-System进行树木轻量化建模遇到的问题 .....	19
4.3.2 树木轻量化？枝干合并！ .....	19
4.4 建模质量评估算法 .....	22
4.5 本章小节 .....	22
第 5 章 实验过程与分析 .....	23
5.1 实验环境 .....	23
5.2 实验结果与分析 .....	23
第 6 章 总结与展望 .....	24
6.1 总结 .....	24
6.2 未来的工作 .....	24
参考文献 .....	25

# 第1章 引言

## 1.1 背景介绍

在互联网飞速发展的今天，网络应用已经延伸到生活的方方面面。微博、人人网、在线购物、在线音乐等已经成为当今人们生活的一部分。面向Web的虚拟现实应用如WebVR、WebGame、WebGIS等也必然将成为虚拟现实发展的重要方向。树木作为自然界常见的事物，在各种虚拟现实的场景中出现的频率很高。然而树木形态各异，结构复杂，给3D建模带来了很大的难度。通常单棵树的数据量已经不小，对于构建一个树木的聚集场景(如森林)就更加庞大，这容易使得场景负荷变大而产生延迟。因此，树木建模的质量和效率将直接决定面向Web的虚拟现实应用的成败。

目前的树木的3D建模，主要是通过专业的3D建模工具(3DSMAX、Maya等)进行手工建模。这种建模方法对建模人员的要求较高，并且需要的时间长。而且这种方法通常最终生成的是面片信息，要表达一棵形态复杂的树木需要大量的顶点信息，导致最终生成的模型体积较大，对于需要大批树木的场景，负荷就会变得更大。

目前树木的轻量化建模，从最简单的基于分形，广告牌技术的建模到稍微复杂的基于规则的建模，都存在一个共同的问题，就是在轻量化的同时，很大程度上舍弃了模型的真实感和树木本身的形态特征。随着当今应用对真实度要求的升高，这类轻量化的建模方法已经不能完全满足需求。真实感与轻量化之间的权衡也成为了当今应用需要考虑的一个重要因素。

本课题基于以上的考虑，从基于图片对树木结构进行完整的恢复，到面向应用需要对真实感与轻量化进行人工控制，到最后模型重建质量的评估，给出了一套完整的解决方案。

## 1.2 课题的主要工作

本课题的主要工作有：

1. 对PyrLK光流法进行改进，并将其运用于三维重建算法中的特征点匹配步骤，使树木重建的模型更加准确和精细。

2. 提出了基于三维体素泛洪和线性拟合的树木骨架抽取方法。该方法区别于传统的3D瘦化骨架抽取方法，它只适用于具有分形结构的3D骨架，所以更能够准确的抽取出树木的骨架。
3. 提出了基于用户交互对树木模型进行完善和轻量化，让最终的应用来决定其所需的树木模型，避免了主观的一味轻量化或一味追求真实感而带来的需求矛盾，将模型的成型延迟至具体的应用。
4. 提出了基于图像的树木重建质量评估方法，对于建模质量和轻量化过程中真实感的下降程度给出了函数化和量化的评价依据。

## 第2章 树木建模的研究综述

树木的建模和造型技术，是计算机图形领域颇具挑战性的研究方向之一。自上世纪六十年代起，大批国内外的研究工作者利用各种不同的方法和技术来构建树木的形态，大大地推进了树木建模技术的发展。目前，在树木的建模领域，主要存在两种不同的类别：虚拟树木建模和对真实世界的树木进行重建。

### 2.1 虚拟树木建模技术

虚拟树木建模主要是指所建模的树木对象并不是直接从现实生活中获取，而是根据一定的规律或生长机理模拟化地对树木进行建模，树木的结构等都是通过过程化的方法所生成，而非从点云中抽取。

一个虚拟树木建模的经典方法是由Lindenmayer于1968年提出的L-System<sup>[1]</sup>，它是一个“字符串重写系统”，后在90年代初，Prusinkiewicz与Lindenmayer一起将L-System规则系统用于描述树木的生长过程<sup>[2,3]</sup>。它用语法表达了植物的生长规则，加入了分枝角度、长度等信息，以便于植物的表达与生长。在此后以L-System为基础的研究工作中，部分研究人员用若干几何模型构建出植物的枝干，并且引入不同的参数来表示植物的生长。另一部分研究人员使用分形的方法来进行树木建模。近几年来，L-System的方法常常以用户勾画为引导，让用户在简便操作的前提下设定L-System的参数，从而构建出树木模型，Okabe在2006年，Anastacio在2009年<sup>[4]</sup>都在这方面作出了贡献。2010年，Hongchun Qu使用BHA自动机和马尔科夫方法自动化地从输入图像中提取了L-System规则，在自动化提取规则方面迈出了第一步。

虚拟树木建模的另一个研究领域是AMAP系统<sup>[5]</sup>。该系统通过观察植物的结构，对植物形式和结构获得定性地理解，然后定量的测量植物形态的数据。植物的生长有一定的随机性，通过概率分布和应用理论来表达随机过程。系统依靠强大的实地数据采集和分析模块，将植物的各项测量数据整合到植物数据库，植物的拓扑结构演化由马尔可夫过程进行分析获得。再通过模式识别方法分析数据中提取生长规则的类型来构造植物的几何形状，应用蒙特卡罗的方法仿真模拟植物的模型，再应用几何的方法来表达其形成规律，并由此制作模型参数表，最后在场景中生成植物的图形。

## 2.2 现实树木重建技术

现实树木重建是指从现实生活中通过照片，视频或者三维扫描，来获得树木的实际数据，通过一系列重建的方法来对树木的几何，物理信息进行复原的过程。

基于图像的树木建模技术，是以在现实中拍摄的树木的图片作为输入，然后根据图片附带的树木信息重建出树木结构的技术。2004年，Reche-Martinez以空间体素的形式重建出了照片中的树<sup>[6]</sup>。2007年，Neubert不仅近似的生成了空间体素，而且还进一步以3D粒子流的方法来模拟生成了细枝和枝干<sup>[7]</sup>。该方法将树木上的点看作吸引子，以物理的吸引力等概念重建出了树木的信息，十分新颖。图片同样可以用来抽取L-System规则，对于树冠密集的树木，Shlyakhter在2001年首次从图片中抽取出了可见部分的规则，然后运用L-System进行处理。对于树叶占据大部分区域的树木图像，香港科技大学学者权龙在2006年，用交互式勾画辅以稀疏3D重建的方法，很好的恢复了树木的信息<sup>[8]</sup>。而对于树干信息占据大部分区域的树木图像，谭平又分别在2007年和2008年，用自动化L-System和用户交互L-System 的方法完成了树木的重建<sup>[9,10]</sup>。2010年，Luis D.Lopez等人提出了一种从稀疏图像序列重建无叶树木的方法。2011年，Chuan Li等人提出了一种从树木视频输入重建树木模型的方法<sup>[11]</sup>。这两种方法均先从图像中获取2D树木骨架，然后对2D树木骨架序列进行三维重建获得三维骨架。由于树木图像本身存在遮挡导致2D树木骨架无法准确抽取，从而影响3D树木骨架的生成，因此这两种方法还原度都比较低。

基于激光扫描仪的体素化模型生成技术，是一种更直接地现实树木重建技术。它利用激光的单色性好、方向性强、能量高、光束窄等特点，直接对树木进行激光扫描，从而得到非常密集的点云数据。大多数基于激光扫描仪的方法都将重点放在了恢复代表枝干的骨架上，因为扫描出的叶子的点云含有太多噪声，以致无法准确重建其信息。1999年，Lazarus和Verroust用生成树的边长来聚合点云，从而获取骨架<sup>[12]</sup>。Bucksch在2008年和2009年将点云分块到八叉树表示的格子，然后用相邻格子间的连线来模拟骨架的曲线<sup>[13]</sup>。2007年Xu用启发式的方法来从扫描的点云中重建树木的主干，然后再用人工合成的办法在其上添加细枝和叶子<sup>[14]</sup>。Côte在2009年同样用人工合成的方式去构建细枝和树叶，但是他对光照散步的合成是通过在扫描的时候进行采样<sup>[15]</sup>。

综上所述，基于L-System的方法虽然轻量化，但是其试图用少量的规则来刻画自然中本就不规则生长的复杂的树木，导致了真实感的缺乏，同时在一个复杂结构中抽取L-System本身也是带有人为主观性和二义性的。基于AMAP系

统的树木建模，虽然其建模的还原度较高，但是它需要大量的数据采集和专业的植物学知识，这对于一般性的应用来说显得超负荷。对于目前的基于图像的建模技术，如何准确的进行三维重建和骨架恢复仍然是一个难点。而基于激光扫描仪的树木建模，设备的价格又太高昂，而且对于骨架的抽取仍有一定的二义性存在。

为了解决这些问题，弥补现有方法的局限，本课题结合现有方法的优点，改进其不足，提出了一整套基于多张图像的树木轻量化建模的解决方案。该方法较好的综合了真实感与轻量化。

### 2.3 本章小节

国内外学者们多年来通过各种不同的技术和思想对树木建模的发展作出了卓越共享，也为后续工作者的研究奠定了坚实的基础。本章首先简要介绍了两大类树木建模的领域，即虚拟树木建模与现实树木重建。然后通过总结这两大领域中的已有方法，从规则生成，植物学领域，三维重建，骨架抽取等角度思考了树木建模问题。在本章最后部分对比和分析了各种方法的特点和不足，为提出改进方案做好了准备。

## 第3章 基于图像的树木轻量化3D建模方法

### 3.1 技术路线

本文提出了一套完整的基于图像的树木轻量化3D建模的方法。该方法首先以树木图片序列作为输入，用经过改进的方法对树木进行三维重建，使三维重建得到的模型精确度和完整性都得以提高。然后再用基于空间方向迭代和步长探索的方法抽取树木的骨架，最终再基于用户交互对骨架进行改善与轻量化。

该技术路线旨在实现一个对建模设备和条件要求不高，适用于一般应用的方法。在方便和简单的基础上，尽可能多的加入自动化，并结合少量用户交互，以实现高效、精确的树木轻量化3D建模。

该方法的主要步骤如下：

#### 3.1.1 基于改进的PyrLK光流法的图像特征匹配

基于图像的树木建模第一步是三维重建，而三维重建的第一步则是特征点的匹配。所谓的特征点匹配，是在多张图片中找到空间同一个点在其上的投影位置，从而为三维重建的后续步骤提供数据支持。这里的特征点，本文选择了具有平移和旋转不变性的 Harris角点，以便用已有的方法快速找出图片中的特殊位置点。然后再结合改进的PyrLK光流法，对找到的特征点在一定的容错区间进行3D匹配，最终将匹配结果存储到匹配文件以供后续使用。

#### 3.1.2 三维重建

特征匹配完成以后，本文使用了美国华盛顿大学西雅图分校Changchang Wu的可视化运动恢复工具VisualSFM来完成基于多张图片的树木三维重建。VisualSFM实现了SiftGPU(GPU加速) 和多核的捆集调整(Multicore Bundle Adjustment)，使得相机参数的恢复更加快速和精确。在这个步骤本文用经过改进的PyrLK光流法的匹配结果替换VisualSFM中的 SIFT特征点匹配文件，进一步地改进了相机参数恢复的准确度和可信度。

### 3.1.3 基于三维体素泛洪与线性拟合的三维树木骨架抽取

在完成了三维重建之后，将会得到一个比较完整的树木空间点云模型。本文根据该点云的空间分布，并结合树木自底向上的自然生长规律和分形的逻辑结构特征，在阈值范围内，进行三维的体素泛洪，同时向多个子方向进行迭代，不断增加步长来扩大邻域范围。在确定邻域以后将几个点数比例较大的方向作为分支方向，并用线性拟合的方法确定其精确的分支方向。同时在迭代过程中及时剔除已经形成枝干的点云，来加速泛洪算法的完成。最终获取到的骨架信息是含有父子关系的节点信息，相比起3DSMAX等手工工具导出的面片模型，这种逻辑结构的模型大大的减小了其尺寸，但是由于逻辑结构并没有多少丢失，所以极具真实感。并且这种结构更便于后续的处理和进一步轻量化。

### 3.1.4 基于用户交互的模型改善与轻量化

由前面方法所得到的树木三维骨架虽然已经是含有父子信息的树木逻辑结构，但是由于前面的步骤都是自动化生成，所得到的结果不可能100%地保证符合具体应用的需求。并且前面的骨架信息虽然比起用面片来表示树木模型已经大大的轻量化了，但是针对实际的应用，本文还可以根据用户的交互来合并分支，从而进一步对模型进行轻量化，以适用于轻量化要求更高的应用。

### 3.1.5 建模质量评估

## 3.2 技术路线图

insertplace...

## 第4章 基于图像树木轻量化建模的若干算法

### 4.1 基于改进的PyrLK光流法的特征点匹配方法

#### 4.1.1 光流法简介

光流的概念最早是由James Gibson提出的。1981年，Horn和Schunck创造性地将二维速度场和灰度联系起来，提出了一种有效的光流计算方法<sup>[16]</sup>。基于亮度不变的假设，图像灰度分布的变化由背景或目标的运动引起，背景或目标的灰度不随时间变化。在这种假设中，光流法通过目标和背景的不同速度来检测运动目标。

进一步说，将三维空间中的目标和场景对应于二维图像平面运动时，他们在二维图像平面的投影就形成了运动，这种运动以图像平面亮度模式表现出来的流动就称为光流(Optical Flow)。也就是说，光流是空间运动物体在观测成像面上对应像素点运动的瞬时速度，这个速度在图像中以每秒像素点的位移个数来衡量，它巧妙地运用2D的灰度变化来表征3D物体的位置和结构变化。而光流场(Optical Flow Field)就是所有光流点的集合，是一个2D瞬时速度场。光流场能够表征整个图像的位移变化，从而对3D运动目标进行检测和跟踪。

在光流法提出以后，很多学者对其进行了研究和改进，并且它们的方法各具特点，算法性能和运用场景各异。其中颇具代表性的是Lucas-Kanade局部平滑法(LK光流法)<sup>[17]</sup>，它用基于微分的方法，利用时变图像灰度的时空微分来计算速度矢量，并且加以图像平滑处理，来进行光流跟踪。后来在2000年Jean-Yves又提出的基于图像金字塔实现的LK光流法，称为PyrLK光流法<sup>[18]</sup>。

#### 4.1.2 PyrLK光流算法

假设图片上的像素点的值函数为 $I(x, y, t)$ ，表示坐标位于 $(x, y)$ 的像素点在时刻 $t$ 的像素值为 $I(x, y, t)$ 。那么经过 $\Delta t$ 时间后，像素值将变为 $I(x + \Delta x, y + \Delta y, t + \Delta t)$ 。有如下推导：

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t$$

$$\begin{aligned} &\Rightarrow \frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t = 0 \\ &\Rightarrow \frac{\partial I}{\partial x} V_x + \frac{\partial I}{\partial y} V_y + \frac{\partial I}{\partial t} = 0 \\ &\Rightarrow I_x V_x + I_y V_y = -I_t \end{aligned} \quad (4.1)$$

Lucas-Kanade光流法算法基于以上原理，并假设两帧图像之间发生的位移是微量的，而且在一个点的邻域内这个位移量是常数。这样可以对一个以 $p$ 点为中心的窗口内的像素点写出一个光流方程组，表征局部图像的运动向量( $V_x, V_y$ )需要满足以下方程组：

$$\left\{ \begin{array}{l} I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ \vdots \\ I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{array} \right. \quad (4.2)$$

这里的 $q_1, q_2, \dots, q_n$ 是局部窗口内的点， $I_x(q_i), I_y(q_i), I_z(q_i)$ 是图片 $I$ 对 $x, y, t$ 的偏导函数在 $q_i$ 处的值。将其写为矩阵形式得：

$$A = \begin{pmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{pmatrix}, \quad v = \begin{pmatrix} V_x \\ V_y \end{pmatrix}, \quad b = \begin{pmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{pmatrix} \quad (4.3)$$

这个方程组的方程个数远远多于未知数，所以 $A$ 是过约束的，LK光流法运用最小二乘法来求解出其光流速度。最小二乘法可以参考4.2.3或查阅相关资料。

LK光流法虽然比较直观，但是存在一个问题，由于能够探测到的运动块的大小和所选窗口的大小呈正相关，为了能够捕捉到大像素块的运动，需要将窗口大小相应调大。但是窗口大小越大，速度就需要在越大的邻域内保持稳定，就越不符合光流在小范围内稳定的假设。基于金字塔的LK光流法是Lucas-Kanada方法的一种改进版实现，它解决了窗口大小的与大块运动捕捉的矛盾。其具体思想如下：

设 $I$ 和 $J$ 是两张灰度图片， $I(x)$ 和 $J(x)$ 分别表示图片 $I$ 和 $J$ 在位置 $(x, y)$ 处的灰度值。现考虑图片 $I$ 上的一点 $\mathbf{u} = (u_x, u_y)$ ，特征追踪的目标就是找到图片 $J$ 上的一点 $\mathbf{v} = \mathbf{u} + \mathbf{d} = (u_x + d_x, u_y + d_y)$ ，使得 $I(\mathbf{u})$ 和 $J(\mathbf{v})$ “相似”。其中向量 $\mathbf{d} = (d_x, d_y)$ 表示图片在点 $\mathbf{x}$ 处的光流速度。下面来定义基于邻域的相似，设 $\omega_x$ 和 $\omega_y$ 是两个整

数，将使得下面式子最小化的向量 $\mathbf{d}$ 定义为光流速度：

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x, y) - J(x + d_x, y + d_y))^2. \quad (4.4)$$

其中邻域窗口大小为 $(2\omega_x + 1) \times (2\omega_y + 1)$ 。式子的含义为寻找向量 $\mathbf{d}$ 使得  $\mathbf{u}$ 和 $\mathbf{v}$ 在邻域窗口大小内的差异最小化。

然后该方法将图像金字塔化，即将原图像作为最高分辨率层，逐步降低图像的分辨率，并作为新的一层，加入到 LK光流法的迭代序列。通过这样多分辨率图层，使得邻域窗口的大小在低分辨率图像对应的区域可以映射到高分辨率图像的更大的像素区域，从而支持了大块运动。

### 4.1.3 改进的PyrLK光流算法

#### 4.1.3.1 加入放射变换

对于PyrLK光流算法，已经能够很好的解决几乎任何像素块大小由平移主导的匹配。然而，这并不足以完美地解决树木上点的匹配问题。因为相邻的两帧图像要求在空间形成一定的夹角进行拍摄，这样在两帧图像上，也一定会产生由空间旋转投影过后带来的平面旋转。而这样的变换在PyrLK光流算法里是无法解决的，因为PyrLK 只是简单的将点的匹配依赖于点的平移。所以，有必要对PyrLK光流算法进行由平移变换到放射变换的扩展。

假设两个点的匹配满足仿射矩阵 $A$ ，那么有：

$$\begin{pmatrix} \Delta x' \\ \Delta y' \\ 0 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \\ 1 \end{pmatrix} \quad (4.5)$$

将其代入式4.1得：

$$(a_{11} \ a_{12} \ a_{13} \ a_{21} \ a_{22} \ a_{23}) \cdot \begin{pmatrix} \frac{\partial I}{\partial x} \Delta x \\ \frac{\partial I}{\partial x} \Delta y \\ \frac{\partial I}{\partial y} \Delta x \\ \frac{\partial I}{\partial y} \Delta y \\ \frac{\partial I}{\partial y} \end{pmatrix} = -I_t \quad (4.6)$$

运用最小二乘法可以得到A的解。

将PyrLK中的定义式4.4稍作修改可使得其支持仿射变换：

$$\text{Let } \mathbf{a}_1 = (a_{11}, a_{12}, a_{13}) \quad \mathbf{a}_2 = (a_{21}, a_{22}, a_{23}) \quad \mathbf{b} = (d_x, d_y, 1)$$

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x, y) - J(x + \mathbf{a}_1 \cdot \mathbf{b}, y + \mathbf{a}_2 \cdot \mathbf{b}))^2. \quad (4.7)$$

#### 4.1.3.2 提高鲁棒性

本文前面几个小节一直在探讨如何改进和完善匹配的方法，从而提高精度和匹配可信度。然而，这其中有一个问题，单方向的去追踪匹配点是否就能确定该两个匹配点真正的匹配呢？其实不然，要确定两个点完全符合之前算法描述的特点，还需要反向进行检查，看 $\mathbf{u}$ 和 $\mathbf{v}$ 之间的匹配是否是双向和可逆的。换句话说，本文之前定义的“相似”其实是单方面的 $\mathbf{u}$ 相似于 $\mathbf{v}$ ，而 $\mathbf{v}$ 是否相似于 $\mathbf{u}$ 还不得而知。因此，考虑到算法的完整性和鲁棒性，有必要进行反向的匹配来确定它们完全匹配。或者退一步，给出一个容错的区间，定义当差异度小于多少时，两个点“相似”。本文采用后一种容错的机制。下面给出了经过添加仿射变换支持和提高鲁棒性的PyrLK 光流法算法的伪代码：

## 4.2 基于三维体素泛洪与线性拟合的三维树木骨架抽取

在获取了精确的点云模型之后，出于后续轻量化的考虑，需要将模型的存储方式由密集的点云转化为逻辑的父子结构。用树形的数据结构来表达现实的树结构，这是很自然的想法，相对于面片结构，树形结构也是一种更为轻量化的存储方式。每个节点表示树枝的起点，存储着该节点的空间位置，半径和该节点的父子枝信息以及兄弟信息。一个节点和它的一个子节点形成一个空间线段，若干空间线段组成一条连续的树枝。

本文从树的生长规律入手，从根节点往子节点生长。生长的依据则为当前节点所在邻域内的空间点云分布，节点邻域大小由步长来控制，步长会探索式地递增，直到达到了增长的阈值，邻域大小才确定下来。然后从其点云分布拟

---

**Algorithm 1** 支持仿射变换和容错机制的PyrLK光流法

---

**Input:** 图像 $I, J$ , 图像 $I$ 中的点 $\mathbf{u}$ , 容错阈值 $\mu$

**Output:** 图像 $J$ 中对应点 $\mathbf{v}$

```
1: 构建图像 $I$ 和 $J$ 的金字塔表示:  $\{I^L\}_{L=0,\dots,L_m}$  和  $\{J^L\}_{L=0,\dots,L_m}$ 
2: 初始化金字塔估计值:  $g^{L_m} = (g_x^{L_m}, g_y^{L_m}) = (0, 0)$ 
3: for  $L = L_m$  to 0 with step of -1 do
4:   定位图像 $I^L$ 上的点 $\mathbf{u}^L$ :  $\mathbf{u}^L = (u_x, u_y) = \mathbf{u}/2^L$ 
5:   设  $\mathbf{a}_1 = (a_{11}, a_{12}, a_{13})$     $\mathbf{a}_2 = (a_{21}, a_{22}, a_{23})$     $\mathbf{b} = (d_x^L, d_y^L, 1)$ 
6:   定义相似度:
    
$$\epsilon(\mathbf{d}^L) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x, y) - J(x + \mathbf{a}_1 \cdot \mathbf{b}, y + \mathbf{a}_2 \cdot \mathbf{b}))^2.$$

7:   最小二乘法估计出 $d^L$ , 使得 $\epsilon$ 达到最小值
8:    $L-1$ 层金字塔估计值:  $g^{L-1} = 2(g^L + d^L)$ 
9: end for
10: 最终光流向量:  $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$ 
11:  $\mathbf{v} = \mathbf{u} + \mathbf{d}$ 
12: 将 $\mathbf{v}$ 作为输入点求出对应点 $\mathbf{u}'$ 
13: if Distance( $\mathbf{u}, \mathbf{u}'$ ) <  $\mu$  then
14:   return  $\mathbf{v}$ 
15: else
16:   return NULL
17: end if
```

---

合出各个分支的方向，从而生长出新的子节点，并递归地生长下去直到点云的边界。

### 4.2.1 三维体素模型

前面三维重建步骤得到的结果是一个点云模型，该模型中的点数量庞大，不适于后续的邻域搜索，因此我们需要对点云进行体素化处理。所谓体素化，就是将点云占据的空间划分成一个个的小立方体，每一个立方体称之为一个体素。

在将点云模型转化为体素模型以后，对于点云的邻域搜索便转化为了对于空间临近体素的搜索，体素的位置就反映了点集的位置，因此不用每次搜索都遍历整个点云，而是只用将步长范围体素中的点集遍历即可。由于体素是我们处理的基本单位，所以体素的大小也直接决定了体素模型的精度，因此，在确保非空体素的空间连续性和效率允许的基础上，本文建议让体素尽可能的小，以保证模型的精度。

insertplace...图片。。。

伪代码。。。

---

#### Algorithm 2 点云模型体素化

---

**Input:** 点云模型  $M$

**Input:** 体素维度  $d$

**Output:** 三维体素数组  $\mathbb{V}[1..d, 1..d, 1..d]$

```

1: 初始化点云边界值  $X_{max} = Y_{max} = Z_{max} = MIN\_FLOAT, X_{min} = Y_{min} = Z_{min} = MAX\_FLOAT$ 
2: for all 空间点  $P(P_x, P_y, P_z) \in M$  do
3:   CheckBoundary( $P$ )
4: end for
5: for all 空间点  $P(P_x, P_y, P_z) \in M$  do
6:    $V_x = \frac{P_x - X_{min}}{X_{max} - X_{min}} \cdot d$ 
7:    $V_y = \frac{P_y - Y_{min}}{Y_{max} - Y_{min}} \cdot d$ 
8:    $V_z = \frac{P_z - Z_{min}}{Z_{max} - Z_{min}} \cdot d$ 
9:    $\mathbb{V}[V_x, V_y, V_z] = \mathbb{V}[V_x, V_y, V_z] \cup \{P\}$ 
10: end for

```

---

#### 4.2.2 三维体素泛洪确定邻域范围

在确定了三维体素模型以后，便需要从根到叶，自底向上地对树的骨架结构进行生长。生长的依据是已经得到的体素模型，将体素模型中点的分布作用于骨架的分支，便可以张成骨架模型。

具体方法是将根节点置为当前节点，对其进行三维泛洪，首先对其相邻的27个体素进行泛洪，若体素不为空，则将其加入邻域范围，若为空，则停止向该方向进行迭代。同时将加入邻域范围的体素置为无效，表示其已经参与了泛洪，不再参与骨架的重建，这样不仅可以对算法的结束有一个很好的约束条件，同时也可减少重复处理的次数，加快算法的完成。然后进行下一次迭代，对新加入的体素进行27方向的泛洪，并把有效的体素加入到邻域范围。接着比较两次迭代体素增加的比例，如果低于设置的阈值，则停止迭代，当前的邻域范围即为三维泛洪得到的当前节点的邻域范围。

insertplace...图片。。。

该算法的伪代码如下：

在进行三维泛洪的时候，可以编程实现27个方向迭代过程的并行化，以提高算法的效率。

#### 4.2.3 通过最小二乘法线性拟合确定分支

当得到邻域范围以后，便得到了邻域内体素在基于当前节点27个方向上的密度分布，而每个体素内又包含着若干的点，因此等于是得到了在当前节点邻域内的点云分布情况。接下来的工作就是怎样从各个方向的点云的分布情况抽取出核心的骨架。本文应用线性拟合的方法来从密集的点中抽取出一条线段，作为该部分的骨架。

该方法首先要剔除掉那些点云密度很小的方向，以免每个节点都朝各个方向长出一些细碎的枝条。因为这些细碎的枝条就算在此步中不剔除，到后续的轻量化的时候也不容许它们的存在。

然后对于剩下的若干方向 $d_1, d_2 \dots d_k$ ，每个方向都对应着树木的一个骨架。在处理某个方向 $d_i$ 时，将其包含的体素中的所有点抽取出来，得到一个密集的点集 $S_i$ 。然后采用待定方程的办法，设直线方程为：

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{d}t, \quad (t \in [0, \infty)) \quad (4.8)$$

其中 $\mathbf{x}_0$ 是当前节点的坐标， $\mathbf{d}$ 是待拟合的直线方向。我们假设点集 $S_i$ 中的

**Algorithm 3** 三维体素泛洪确定邻域范围

**Input:** 当前体素 $C$ , 三维体素数组 $\mathbb{V}[1..d, 1..d, 1..d]$ , 泛洪方向数组 $\mathbb{D}[1..27]$ , 邻域范围增长比例阈值 $\lambda$

**Output:** 邻域范围内体素集合 $\mathbb{S}$

```

1: 初始化单次迭代新增体素集合 $\mathbb{S}'$ 
2:  $\mathbb{S}'.AddVoxel$ (根节点所在体素)
3: for all 泛洪方向 $Direction \in \mathbb{D}$  do
4:    $NewIndex = C.Index + Direction$ 
5:    $NewVoxel = \mathbb{V}[NewIndex.x, NewIndex.y, NewIndex.z]$ 
6:   if  $NewVoxel$ 非空  $\cap$   $NewVoxel$ 有效 then
7:      $\mathbb{S}'.AddVoxel(NewVoxel)$ 
8:   end if
9: end for
10: 体素增长比例 $\mu = \frac{\mathbb{S}'.VoxelCount}{\mathbb{S}.VoxelCount}$ 
11: if  $\mu > \lambda$  then
12:   for all  $voxel \in \mathbb{S}'$  do
13:     把 $voxel$ 作为当前体素进行递归调用
14:   end for
15: end if
16: return  $\mathbb{S}$ 
```

点 $P_1, P_2, \dots, P_m$ 都在直线上, 则可以得到以下方程组:

$$\begin{cases} a_{11}d_x + a_{12}d_y + a_{13}d_z = b_1 \\ a_{21}d_x + a_{22}d_y + a_{23}d_z = b_2 \\ \dots \\ a_{n1}d_x + a_{n2}d_y + a_{n3}d_z = b_n \end{cases} \quad (4.9)$$

其中具体数值未给出, 注意这里的 $n = 3m$ , 因为每个点 $P$ 可以提供三个方向的方程式。在这个方程组中, 令

$$\mathbf{U} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

在实践中，由于筛选方向上的点数较多且发散分布，由线性代数的理论知， $\mathbf{U}$ 是过约束的，即 $n > r$ , 其中 $r$ 是矩阵 $\mathbf{U}$ 的秩。这种情况下没有标准的解，只能找到使误差最小的向量 $\mathbf{d}$ ，误差定义为：

$$E \stackrel{\text{def}}{=} \sum_{i=1}^n (\mathbf{d}_{t_i} - \mathbf{x}_i + \mathbf{x}_0)^2 = \|\mathbf{U}\mathbf{d} - \mathbf{b}\|^2 \quad (4.10)$$

由于 $E$ 正比于方程的均方误差，因此只要 $E$ 达到最小值，那么点集相对于该直线的波动就最小。换句话说，也就是该直线最好的模拟了该点集所表示的骨架。由线性代数的方法很容易可以解得 $\mathbf{d} = [(\mathbf{U}^T \mathbf{U})^{-1} \mathbf{U}^T] \mathbf{b}$ 。

insertplace...图片。。。

下表是得到邻域信息后进行骨架抽取的伪代码，其中*Least Squares Processing*表示运用最小二乘法进行线性拟合。

---

#### Algorithm 4 基于邻域的骨架抽取

---

**Input:** 当前节点体素 $V$

**Input:** 骨架方向数组 $D[1..n]$

**Output:** 当前节点子节点集合 $\$$

```

1: for all 骨架方向 $d \in D$  do
2:    $NewChild \leftarrow LeastSquaresProcessing$ 
3:    $\$.AddChild(NewChild)$ 
4: end for
5: return  $\$$ 

```

---

#### 4.2.4 获取骨架半径

树木骨架的半径对树木模型的真实感有着十分显著的贡献，所以尽可能准确的获得骨架的半径信息能够有助于重建出极具真实感的树木模型。对于树木半径的获取方法也有许多，主要分为根据规则生成半径和从树木点云结构中获取半径两种方式。

对于基于规则来生成半径，最简单的方法是对树木半径进行线性地递减，即 $r = cR$ ，其中 $r$ 为子枝半径， $R$ 是父枝半径， $c$ 为一个线性倍数，这个倍数可以固定，也可以进行随机的扰动从而增进多样性。Leonardo da Vinci在经过大量观察后总结出了一种更符合自然规律的树木父子枝直径的关系公式： $D^2 = \sum_{i=1}^n d_i^2$ ，其中 $D$ 为父枝直径， $d_i$ 为第 $i$ 个子枝的直径， $n$ 为子枝的数量。这个公式被广泛地用于树木枝干的半径模拟。

区别于基于规则的半径生成方法，本文为了进一步提升真实感，选择在进行骨架抽取的同时，同样进行半径抽取的方法。注意，用该方法的前提是点云分布须均匀化，然而基于图像进行三维重建得到的树木点云会呈现表皮化的现象，这是由于图片上的点都是树木的表皮点，所以在得到三维点云后，是需要进行一些修复工作的，本文用随机点填充的方法对该点云模型进行了实心化的修复。当点云分布满足均匀化时，在对某个骨架进行拟合之后，对于拟合出来的直线，来计算所有参加拟合该直线的点到该直线的平均距离 $D_{avg}$ ，然后就可以计算该骨架的半径 $R = D_{avg} * 2$ 。由于点云分布均匀，所以半径显然就是平均距离的2倍。该算法的伪代码如下：

---

**Algorithm 5** 骨架半径抽取
 

---

**Input:** 拟合出的当前骨架直线 $L$

**Input:** 当前骨架的点集 $\mathbb{S}$

**Output:** 当前骨架半径 $R$

```

1: 初始化距离和 $D_{sum} = 0$ 
2: for all 空间点 $P \in \mathbb{S}$  do
3:   点到直线距离 $D_{sum}+ = CalculateDistance(P, L)$ 
4: end for
5: 平均距离 $D_{avg} = D_{sum}/\mathbb{S}.Count$ 
6: 骨架半径 $R = D_{avg} * 2$ 
7: return  $R$ 

```

---

insertplace...实验图片对比图(线性、2次方、平均距离)

### 4.3 基于枝干合并的轻量化处理

用基于多方向迭代与步长探索得到的三维树木骨架通常是很细致和准确的，尽管它相对于用3DSMAX等建模工具手工建模得到的面片模型已经大大的轻量

化了。但是如果应用是用于大规模的树木建模，那么我们有必要根据应用需求进一步进行轻量化处理。

### 4.3.1 L-System的尝试

#### 4.3.1.1 L-System简介

L-System是一种并行的重写系统和正规语法，它的结构可以用可以定义为一个3元组：

$$\mathbf{M} = (V, \omega, P)$$

其中：

- **V**(字母表) 表示可以被替代的字符的集合。
- $\omega$ (初始串) 表示L-System的初始状态。
- **P**(规则集合) 表示一系列的衍生规则。

L-System可以根据这三个组成部分的不同而递归地产生形态各异的字符串。由于L-System具有递归生长的特性，因此我们可以用L-System规则来表达一个具有自相似形态或者分形结构的物体，比如本文所研究的对象——树木。

#### 4.3.1.2 树木模型的参数化L-System规则抽取

球面海龟几何的提出，用参数化的L-System规则描述了树木的结构信息。在球面海龟几何中，节点的空间几何信息用4个量(长度 $l$ 、半径 $r$ 、父子枝夹角 $\theta$ 和水平转角 $\phi$ ) 和4个扩展符号(+、-、&、^)来表示：

- $+(l)$  表示以当前位置为起点，在当前方向上前进 $l$ 单位个长度
- $!(r)$  表示设置当前节点半径为 $r$
- $\&(\theta)$  表示设置父子枝夹角为 $\theta$
- $^(\phi)$  设置水平偏角为 $\phi$

在球面海龟几何中，将每个骨架节点生成一条参数化的L-System规则，形如：

$$N(l, r) \rightarrow \&(\theta_0)\wedge(\phi_0)!r+(l)S_0(l*a_0, r*b_0)... \&(\theta_n)\wedge(\phi_n)!r+(l)S_n(l*a_n, r*b_n) \quad (4.11)$$

其中 $N$ 表示当前枝条， $S_0 S_n$ 表示当前枝条的 $n$ 个子枝条， $a_i b_i$ 分别表示第 $i$ 个子枝条与当前枝条的长度比和半径比， $\theta_i \phi_i$ 分别表示第 $i$ 个子枝条与当前枝条的空间夹角和水平偏角。

### 4.3.1.3 使用L-System进行树木轻量化建模遇到的问题

在用参数化L-System进行树木轻量化建模时，在进行规则归纳时，有个难以克服的问题。考虑将规则4.11中的 $a_0$ 换成 $a'_0$ ，则规则变成一个完全不同的规则。这意味着对于两个分支规则，这两个规则中的子枝的长度，半径，转交，偏角等必须完全相等才能归纳为同一个规则。而对于自然界中形态结构复杂的树木，每个分支规则几乎不可能完全等同于另一个规则。

对上面的问题有一种解决方法就是将参数区间化，将属于同一区间的参数的值视为相同。比如我们可以将父子枝间的转角分为18个区间，每个区间的大小为10度。但是经过分析就可以察觉，这并没有从根本上解决这个问题。假设我们将这4个变量都各自划分为10个区间，那么规则总数最多可以有10000个，而且在这种情况下，两个规律相同的几率也是非常小的。如果我们将分区数量减少，则又有可能将本来差异比较大的规则归纳为一个规则，不符合真实感的要求。

所以，经过分析，这种用参数化L-System进行树木轻量化建模的方法并不适用于从骨架中去抽取规则，而是适用于反向地用其描述的规则去产生一棵树，如台湾学者戴文凯就对单棵树的L-System 规则进行随机扰动而轻量化的建模出了整片森林。

### 4.3.2 树木轻量化？枝干合并！

用L-System的方法抽取规则所产生的问题，从本质上讲，是由于自然界的树木形态太复杂和多变。与其从一个本就不规则生长的事物中去抽取规则，还不如直接地在其逻辑结构上进行一系列的轻量化操作。本文提出了对已抽取的树木骨架中对视觉影响不大的部分进行合并的方法，从而在尽可能保证模型的视觉效果的基础上，进一步地减小树木模型的体积，使得其能更广泛地应用到WebVR、WebGame 等各个领域。

insertplace...如图所示.... 进行图片阐述....

伪代码。。。

功能：根据纵向合并角度参数，以当前节点为发起点递归式地纵向合并枝干

功能：根据横向合并角度参数，横向合并夹角小于角度阈值的末端叶子枝干

---

### Algorithm 6 纵向合并枝干

---

**Input:** 纵向合并角度 $\alpha$

**Input:** 当前节点 $N$

**Output:** None

```
1: for all 节点 $N' \in N.Children$  do
2:   while  $N'.ChildCount = 1$  do
3:      $\vec{u} \leftarrow N'.Position - N.Position$ 
4:      $\vec{v} \leftarrow N''.Position - N'.Position$ 
5:      $\gamma \leftarrow \cos^{-1}\left(\frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|}\right)$ 
6:     if  $\gamma < \alpha$  then
7:        $N.child \leftarrow N.AddChild(N'')$ 
8:        $N.child \leftarrow N.DeleteChild(N')$ 
9:     end if
10:     $N' \leftarrow N'.FirstChild$ 
11:  end while
12: end for
13: if  $N.ChildCount > 1$  then
14:   for all 节点 $N' \in N.Children$  do
15:     以 $N'$ 为当前节点递归调用该函数
16:   end for
17: end if
```

---

---

**Algorithm 7** 横向合并枝干

---

**Input:** 初始化横向合并角度 $\beta$ **Input:** 设定当前节点 $N$ **Output:** None

```

1: for all 节点对 $P \in N.Children$  do
2:    $N_1 \leftarrow P.FirstNode$ 
3:    $N_2 \leftarrow P.SecondNode$ 
4:   if  $N_1.ChildCount = 0 \wedge N_2.ChildCount = 0$  then
5:      $\vec{u} \leftarrow N_1.Position - N.Position$ 
6:      $\vec{v} \leftarrow N_2.Position - N.Position$ 
7:      $\gamma \leftarrow \cos^{-1}\left(\frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|}\right)$ 
8:     if  $\gamma < \beta$  then
9:       New Node  $N'$ 
10:       $N'.Position \leftarrow (N_1.Position + N_2.Position)/2$ 
11:       $N'.Radius \leftarrow \max(N_1.Radius, N_2.Radius)$ 
12:       $N.child \leftarrow N.DeleteChild(N_1)$ 
13:       $N.child \leftarrow N.DeleteChild(N_2)$ 
14:       $N.child \leftarrow N.AddChild(N')$ 
15:      退出循环并以当前节点 $N$ 重新调用该函数
16:    end if
17:  end if
18: end for
19: for all 节点 $P \in N.Children$  do
20:   以 $P$ 为当前节点递归调用该函数
21: end for

```

---

#### 4.4 建模质量评估算法

分步来： 1. 图片序列信息量 2. 三维重建质量 3. 骨架抽取质量

#### 4.5 本章小节

## 第 5 章 实验过程与分析

5.1 实验环境

5.2 实验结果与分析

## 第 6 章 总结与展望

### 6.1 总结

本文首先总结了国内外树木建模工作者的技术与方法，并分析了它们的优缺点。为了得到真实感强的树木模型，本文从改进了传统三维重建的匹配算法。

### 6.2 未来的工作

## 参考文献

- [1] Lindenmayer A. Mathematical models for cellular interaction in development: Parts I and II. *Journal of Theoretical Biology*, 1968, 18.
- [2] Prusinkiewicz P, Lindenmayer A. The algorithmic beauty of plants. New York, NY, USA: Springer-Verlag New York, Inc., 1990.
- [3] Prusinkiewicz P, Lindenmayer A, Hanan J. Development models of herbaceous plants for computer imagery purposes. *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, New York, NY, USA: ACM, 1988. 141–150.
- [4] Anastacio F, Prusinkiewicz P, Sousa M C. Sketch-Based Interfaces and Modeling (SBIM): Sketch-based parameterization of L-systems using illustration-inspired construction lines and depth modulation. *Comput. Graph.*, 2009, 33(4): 440–451.
- [5] Reffye P, Edelin C, Françon J, et al. Plant models faithful to botanical structure and development. *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, New York, NY, USA: ACM, 1988. 151–158.
- [6] Reche-Martinez A, Martin I, Drettakis G. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph.*, 2004, 23(3): 720–727.
- [7] Neubert B, Franken T, Deussen O. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph.*, 2007, 26(3).
- [8] Quan L, Tan P, Zeng G, et al. Image-based plant modeling. *ACM SIGGRAPH 2006 Papers*, New York, NY, USA: ACM, 2006. 599–604.
- [9] Tan P, Zeng G, Wang J, et al. Image-based tree modeling. *ACM SIGGRAPH 2007 papers*, New York, NY, USA: ACM, 2007.
- [10] Tan P, Fang T, Xiao J, et al. Single image tree modeling. *ACM Trans. Graph.*, 2008, 27(5): 108:1–108:7.
- [11] Li C, Deussen O, Song Y Z, et al. Modeling and generating moving trees from video. *ACM Trans. Graph.*, 2011, 30(6): 127:1–127:12.
- [12] Verroust A, Rocquencourt I, Lazarus F. Extracting Skeletal Curves from 3D Scattered Data. *The Visual Computer*, 1999, 16: 15–25.
- [13] Bucksch A. SKELETONIZATION AND SEGMENTATION OF POINT CLOUDS USING OCTREES AND GRAPH THEORY, 2006.
- [14] Xu H, Gossett N, Chen B. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 2007, 26(4).
- [15] Côté J F, Widlowski J L, Fournier R A, et al. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sensing of Environment*, 2009, 113(5): 1067 – 1081.
- [16] Horn B K P, Schunck B G. Determining Optical Flow. *ARTIFICIAL INTELLIGENCE*, 1981, 17: 185–203.
- [17] Lucas B D, Kanade T. An Iterative Image Registration Technique with an Application to Stereo Vision. 1981. 674–679.
- [18] Bouguet J. Pyramidal implementation of the Lucas Kanade feature tracker. Intel Corporation, Microprocessor Research Labs, 2000..