**清 华 大 学**

**综 合 论 文 训 练**

**题目：社交网络中的谣言检测系统**

|  |  |
| --- | --- |
| 系 别： | 软件学院 |
| 专 业： | 计算机软件 |
| 姓 名： | 钟仰新 |
| 指导教师： | 刘世霞 副教授 |

2016年 月 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签 名： 导师签名： 日 期：

中文摘要

近年来，社交网络（如推特、微博）被互联网用户广泛使用。但同时此类平台中存在大量不真实谣言，其传播造成的危害巨大。由于社交网络消息数目庞大，使得人工排查谣言成本很高，因此对谣言的自动检测技术具有重大意义。

本文描述了一个谣言检测系统，此系统首先对社交网络中的消息进行谣言特征匹配，检测出疑似谣言的消息；然后对这些候选消息进行聚类，生成疑似的谣言话题；接着对候选话题进行特征抽取，利用监督学习技术对其进行可疑度排名，最终检测出最有可能为谣言的话题。

在对消息进行聚类时，本文采取了基于相似度矩阵的聚类方法，研究了包括发布时间、消息标签、命名实体、文本单词、用户群体的相似度度量，聚类上采用了k均值、层级聚类以及谱聚类等方法，并对比了不同度量和聚类算法之间的优劣。实验表明，此系统能有效地将属于同一话题的不同消息归为一类，降低人工排查谣言的成本。

在对候选话题进行特征抽取时，本文总结相关工作，选取了45种不同的特征，并引入了名为过滤器和包装器的两种特征选择技术，对比两种技术并将其结合使用，最终选择出有效的特征子集。实验证明，经过选择的特征子集能有效提高系统对谣言的识别准确度。

在对话题进行可疑度排序时，系统选取了决策树和朴素贝叶斯两种分类器算法，并引入多分类器组合投票技术。实验表明，引入组合投票技术能有效提高排名系统的实用性，显著降低人工排查的成本。

**关键词**：社交网络；谣言检测；聚类；特征选择；监督学习

ABSTRACT

In recent years, social network (e.g., twitter and weibo) has been widely used on the Internet. But at the same time, there are a large number of false rumors in social network. Their spread may have harmful effect on individuals and society. The large amount of messages on social network make the cost of manual checking extremely high. As a result, automatic rumor detection techniques are of great significance.

This thesis describes a rumor detection system. It first filter out suspected rumor tweets by pattern matching, and then cluster these candidate tweets into topics. After that, it extract features of the candidate topics and rank them by the likelihood of false rumor through supervised learning technique. In this way, the most likely rumor topics can be detected automatically.

The system applies clustering techniques based on similarity matrix. The thesis studies similarity metrics that consider different factors including published time, hashtags, named-entities, terms and users. The system employs k-means, hierarchical and spectral clustering algorithms. Experiments demonstrate that the system can lower the cost of rumor checking by effectively clustering similar messages into same topics.

For feature extraction, the thesis chooses 45 different features and introduces two feature selection techniques, filter and wrapper, to select effective feature subset for the training of next step. A method of combining these two feature selection techniques have been proposed in this thesis. Experiments show that the new method and the step of feature selection can effectively improve the accuracy of rumor detection.

To rank and find most likely rumor topics, the system employs Decision Tree and Naive Bayes as classifiers and a multiple classifiers combining technique by voting has been introduced. Experiments demonstrate that the combining method can effectively improve practicability of the detection system.

**Keywords:** Social network; rumor detection; clustering; feature selection; supervised learning

目录

[第1章 引言 1](#_Toc452072034)

[1.1 问题提出 1](#_Toc452072035)

[1.2 光场简介 1](#_Toc452072036)

[1.3 已有研究概述 2](#_Toc452072037)

[1.4 论文组织结构 3](#_Toc452072038)

[第2章 基于手持相机的光场采集系统 4](#_Toc452072039)

[2.1 系统目标 4](#_Toc452072042)

[2.2 光场采集 4](#_Toc452072044)

[2.3 光场渲染 4](#_Toc452072045)

[2.4 实现细节 4](#_Toc452072046)

[第3章 光场采集 4](#_Toc452072047)

[3.1 基于标志的采集方法 4](#_Toc452072048)

[3.2 基于场景特征的采集方法 9](#_Toc452072049)

[3.3 种采集方法的对比 10](#_Toc452072050)

[第4章 光场渲染 12](#_Toc452072051)

[4.1 逐段线性的渲染方法 12](#_Toc452072052)

[4.2 N近邻的渲染方法 16](#_Toc452072053)

[4.3 两种渲染方法的对比 18](#_Toc452072054)

[第5章 结论 21](#_Toc452072055)

[5.1 工作总结 21](#_Toc452072056)

[5.2 存在问题与未来工作 21](#_Toc452072057)

[插图索引 22](#_Toc452072058)

[表格索引 23](#_Toc452072059)

[参考文献 24](#_Toc452072060)

[致 谢 25](#_Toc452072061)

[声 明 26](#_Toc452072062)

[附录 A 外文资料的调研阅读报告 27](#_Toc452072063)

# 引言

## 问题提出

光场（light field）渲染技术属于基于图像渲染（Image based rendering，简称IBR）技术的一种。这类技术一般以真实世界拍摄的照片作为输入，输出三维场景的渲染图。正因为以照片作为输入，这类渲染技术更容易获取更高的真实性，渲染出来的图片更逼近真实世界。比如谷歌街景，通过使用相机在不同角度拍摄几张照片，就可以实现让用户在计算机前通过鼠标操作的方式连续地改变视角，就好像走在真实的街道中一样。

虽然光场渲染可以带来很强的真实感，但是传统光场采集却不是轻松的事情。传统的光场采集需要特殊的设备，造价昂贵，携带也不方便。本文致力于开发一个基于手持相机的光场采集系统，给予用户使用手持相机也能采集光场并实时查看渲染结果的能力。

## 光场简介

光场的定义是空间中某点向某个方向的光照度（illumination）[1]。根据定义，光场可以表示为空间点三维坐标(x,y,z)，以及方位角(θ)和仰角(φ)的五维函数：，也称为全光函数（plenoptic function）。

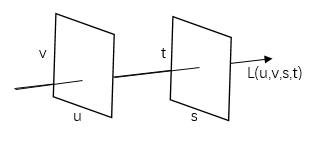
但是考虑到在没有遮挡的情况下，空间中同一条光线上的光照度应该相同，全光函数表示方法有冗余，可以简化为四维函数：假设空间中有两个参数曲面，从一个曲面上一点(u, v)到第二个曲面上一点(s, t)对应一条直线，空间中每条这样的直线都对应一个光照度，如图 1.1。这样两个曲面之间的光场就可以表示为形式是的四维函数。

图 1.1 光场的四维函数表示

现在考虑如何用相机进行光场采集。假设uv，st是两个平面。如果我们在uv平面上每个点都放置一个相机，相机都朝向st平面进行拍照。拍照完成后，uv平面上每个点都存储着一张照片，照片上的每个点的颜色值对应的就是4D函数的光照度。实际采集的时候我们只能进行离散的采集：即在uv平面上的采集点是离散的，st平面对应的像素点是离散的。对于未采集到的光线要用临近点插值得到。

## 已有研究概述

对于与本文相关的研究，主要分为两部分进行叙述：光场采集和光场渲染。

主流的光场采集方法主要分为三大类[2]:

1. 用运动相机进行采集。这类方法是将相机固定在某个机械装置上，在机械装置不断运动的过程中连续用相机进行拍照，利用相机在不同位置拍的照片进行光场重建。比如相应的例子比如苹果的QuickTime VR[2]，微软的concentric mosaics[4]等。
2. 用相机阵列进行采集。这类方法是将一系列相机按照一定间距固定，然后同时进行拍照，利用不同相机拍的照片进行光场重建。对应的例子比如斯坦福大学建造的光场采集系统[5]。
3. 用透镜阵列进行采集。这类方法是在相机的主透镜之后添加一个透镜阵列，透镜阵列的每一个透镜都会成一个像，利用不同透镜成的像进行光场的重建。相应的例子比如Lytro相机[6]。

对于以上所述的三类主流的光场采集方法都需要特殊的设备进行支持。由于有机械设备的支持，相机在进行拍照的时候可以较为方便地知道相机坐标。而本文所希望达到的目标是允许用户使用手持相机就可以实现光场捕捉，与以上方法的差别在于拍照的时候需要通过定位算法计算相机坐标。

光场渲染的方法主要分为两大类：

* + - 1. 结构化的光场渲染。这类方法采集点之间的位置具有一定结构，有可能是共平面的或是等间距的。代表工作包括[1]。其主要原理是在uvst四维空间找最近的四个采集点进行插值。
      2. 无结构的光场渲染。这类方法采集点之间的位置没有特定的结构。代表工作包括[7]。其主要原理是找和目标射线角度最接近的N条射线进行插值。还有[8]。其主要原理是在视角空间找和目标射线最接近的3条射线进行插值。

本文由于是让用户用手持相机进行光场采集，所以只能采用无结构的光场渲染。本文对以上提到的两种无结构的光场渲染方式都有实现。

## 论文组织结构

接下来本文将会以以下方式介绍系统实现：第二章中将会对系统的实现概况进行介绍，第三章将详细介绍光场的采集方法，包括基于标志的采集方法和基于场景特征的采集方法，第四章将详细介绍光场的渲染方法，包括逐段线性的渲染方法和N近邻的渲染方法，最后第五章进行总结并介绍未来工作。

# 基于手持相机的光场采集系统

## 系统目标

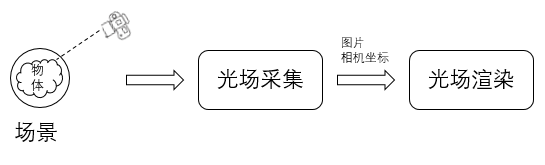
本系统的目标在于通过用户使用手持相机所拥有的硬件资源实现光场采集。想要实现这个目标，主要要解决两个问题。第一个问题来自于光场采集方面：如何辅助用户实现高质量的场景覆盖，如何在已有数据的基础上引导用户进行光场采集，如何实时地向用户反馈辅助采集过程。第二个问题来自于光场渲染方面：用户通过手持相机采集到的数据一般是无结构的数据，如何针对无结构的光场实现高效的渲染。系统框架图如图 2.1所示。

图 2.1 光场采集系统框架图

## 光场采集

本系统使用增强现实技术，根据一定的先验知识和场景信息，实时地计算相机坐标，然后利用坐标将目标图形绘制在场景中。系统采用了两种计算相机坐标的方式：基于标志的和基于场景特征的。实现细节会在第三章详细说明。无论哪种那种方式，采集流程都是一样的。

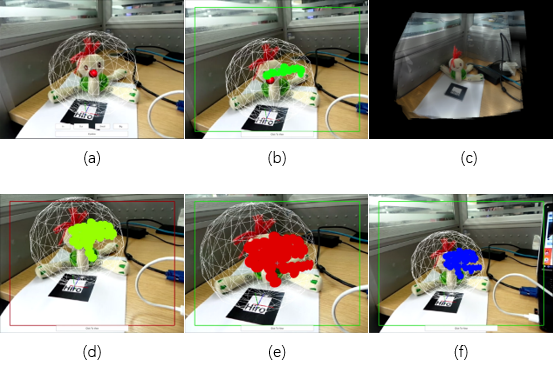
首先用户需要根据指示，在采集物体周围放置一个线框球（后面都修改）。界面上有四个按钮，分别可以将线框球沿着相机朝向向内，向外移动，将线框球放大，缩小。用户可以使用这些按钮，调整线框球的位置和大小。当用户调整到线框球包围住场景物体之后，就可以进入采集流程。详细的调整过程会在第三章具体介绍光场采集的时候进行介绍。如图 2.2 (a)是调整好后的线框球包围住场景物体。

图 2.2 光场采集系统概述

用户移动相机的位置就可以进行采集。用户每采集到一张图片，系统就会在相机朝向上与线框球的交点处绘制一个圆片，作为标志表示该点已经采集过了。圆片半径越小对场景的采集越密集，渲染效果越好。但是半径过小占用过多存储空间，增加渲染的复杂度。系统实现中设定圆片半径为0.05倍的线框球半径。如 图 2.2 (b)是采集了11个点的情形。如果当前采集点没有被之前的圆片所覆盖，系统将认为该点没有被采集，自动保存一张对应的图片，相反如果采集点在圆片所覆盖区域，则表示该点被采集过了，系统不做保存。线框球被圆片覆盖的部分越大表示采集的越充分。系统在任何时间都可以退出并查看采集到的光场，如图 2.2 (c)显示的是查看光场的情形。

为了保证采集质量，采集过程中还对用户增加了如下约束：

1. 为保证线框球在视野中央，系统在屏幕中央绘制一个矩形框，如果线框球超出矩形框的范围，将会停止采集过程，如图 2.2(d)线框球超出了矩形框的范围。
2. 为保证采集点到线框球的距离差别不是太大，系统会通过圆片的颜色给予用户提示。如果相机到线框球的距离变近，圆片会逐渐变为红色。系统实现中设定当距离小至初始距离的0.8倍的时候圆片完全由绿色变成红色。如图 2.2(e)距离过近圆片变为了红色。
3. 如果相机到线框球的距离变远，圆片会逐渐变为蓝色。系统实现中设定当距离小至初始距离的1.2倍的时候圆片完全由绿色变成蓝色。如图 2.2(f)。如果相机到线框球的距离过近（0.8倍初始距离）或是过远（1.2倍初始距离），则会停止采集过程。

## 光场渲染

在光场采集的任意时刻，用户都可以点击界面上的按钮查看已采集的光场。用户可以通过与屏幕的交互，连续地改变光场观察的视角。图 2.3中白色的球表示框线球，三色坐标轴表示的是观察相机的位置，三角网格表示的是成像平面。本系统实现了三种交互的方式：

1. 使用一根手指滑动屏幕，用户可以在保持视点到线框球球心距离不变的情况下改变视角的位置，也就是说用户可以左右上下地改变观测位置来观测场景。如图 2.3(a)，相机由左边转到右边。
2. 使用两根手指滑动屏幕，用户可以改变当前视角到球心的距离，也就是说用户可以前后运动改变观测位置来观测场景。如图 2.3(b)，相机距离线框球由近至远。
3. 使用三根手指滑动屏幕，用户可以改变当前成像平面到视角的距离。默认的成像平面是过球心与相机朝向垂直的平面。改变成像平面到视点的距离意味着调节焦距，通过这种方法可以调节场景对焦的方式。如图 2.3(c)，成像平面从球心向远离相机的方向运动。系统实现了两种光场渲染的方式：N近邻方法和逐段线性方法。实现细节会在第四章详细说明。

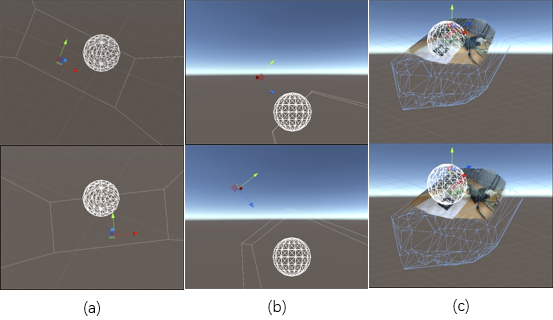


图 2.3 视角和线框球的相对位置

## 实现细节

实验环境：本系统主要在Android系统上进行了实现，分别在Samsung SM-N7506V, Android 4.3 手机和 Samsung EK-GC110 Android 4.1 相机上进行了实验。

系统实现：本系统基于Unity进行开发，代码部分主要使用C#语言和ShaderLab[9]语言进行编写。

其他依赖：光场采集部分。基于标志的采集方法使用ARToolkit[10]进行编写，基于场景特征的方法使用Wikitude[11]进行编写。

# 光场采集

这里的光场采集指的是用户使用单孔的手持相机进行场景中光场数据的捕捉。光场采集中最重要的问题在于如何根据手持相机拍摄下来的视频帧数据，确定相机的三维坐标和旋转角度。

## 基于标志的采集方法

标志一般指的是8厘米×8厘米的正方形图案。如图 3.1(a)中的“岳”字标志。ARToolkit的功能是以相机采集的视频为输入，根据视频帧里图案的大小和旋转角度，来计算相机的相对坐标和旋转角度。相机的相对坐标和旋转角度可以直接在ARToolkit在Unity中提供的ARCamera读取得到。当关于相机的三维信息确定了以后，就可以视频帧上绘制三维图形了。

一个图案想要使用作为标志需要具有如下特征：

1. 图案需要有一定复杂性。且图案不能是对称图形，否则系统无法正确通过图案识别相机坐标。
2. 图案需要放置在一个黑色正方形框内，且边框宽度占图案的25%，即2厘米。
3. 黑色边框之外还需要有一段白色或浅色的填充。

关于标志的图案可以自由指定，当设计好标志之后，就可以在白纸上打印出来。在矫正好相机之后，就可以进行标志的训练。将相机连接电脑，打开ARToolkit提供的训练程序，当标志外被标志上红色和绿色的边框，表示系统可以识别标志，此时保存标志即可。如图 3.1(a)。

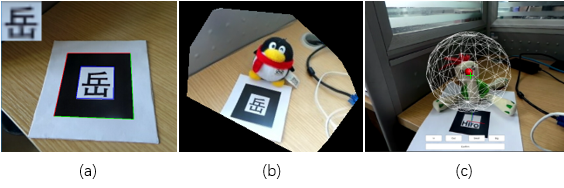
标志训练好之后就可以使用标志进行采集了。采集过程分为两个阶段。第一阶段是调整阶段。在这一阶段用户需要把线框球放置到场景物体的周围。在三维空间中想要把线框球放到合适的位置并不是一件容易的事情。最简单的方法是要求用户在x轴，y轴，z轴三个维度以及球的半径一共四个维度上进行调整，但是这样调整费时费力。本系统中，用户只需要对线框球到相机距离以及线框球半径这两个维度进行调整。具体调整过程如下。在调整阶段，屏幕下方有四个按钮以及屏幕中央有一个按钮可供用户操作，来调整线框球的位置。如图 3.1(c)。屏幕中央的十字为绿色时，表示锁定线框球在屏幕中央，此时线框球只能沿着相机所正对的方向进行前后运动。当十字为绿色时，屏幕下方的四个按钮被激活，分别对应增大/减少线框球到相机的距离，增大/减小线框球半径。当屏幕中央的十字为灰色的时候，表示解除线框球的锁定，屏幕下方的四个按钮被禁止。此时无论相机如何运动，线框球在三维空间中的位置保持不变。然后用户可以前后左右移动相机，查看线框球是否处在了合适的位置。在整个调整过程中，用户需要反复切换锁定与解锁状态，在锁定状态下调整球的位置，在解锁状态下查看球的三维坐标，直到球处在场景物体周围时才能结束，这一过程即需要用户拥有一定的经验，同时需要有一定的空间想象能力。

图 3.1 显示的是基于标志的采集方法

当线框球处在合适位置之后，就可以进入第二阶段，拍摄阶段。如第二章所述，拍摄阶段就是用户移动相机用圆片覆盖线框球进行光场采集的过程。如果当前相机朝向上与线框球的交点没有被之前的圆片所覆盖，则系统会将当前相机的三维坐标与四元数（quaternion）表示的旋转角度共七个数写入到一个二进制文件中。同时将当前视频帧进行保存。考虑到手持设备上文件读写操作速度较慢，系统实现中将图片保存进行了多线程的实现。与此同时，将当前视频帧压缩成320x240的jpg图片进行存储，进一步减小写入SD卡的文件大小。对于上述的一张jpg图片，大小大约为20KB到30KB之间，记录相机坐标的二进制文件大约为5KB到10KB之间。对于一个典型应用，大约需要60到100张图片，最后占用文件系统的总空间在2MB到4MB左右不等。最后，对于基于标志的采集方法，系统还额外增加了一项约束：当标志超出视频帧的边界时，屏幕变红，系统暂停采集。

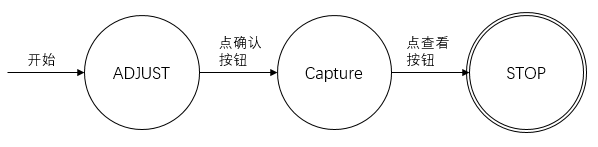
程序使用状态机的方式进行组织，如图图 3.2所示。主要分为三种状态ADJUST, CAPTURE, STOP。程序开始的时候，进入ADJUST状态，此时用户可以点击3.1小节所述的屏幕上的五个按钮，调节线框球的位置和大小。当用户点击确认按钮时，程序进入CAPTURE状态，此时用户开始采集光场数据，具体算法在之后小节中详细描述。当用户点击查看光场时，程序进入STOP状态，此时用户可以查看光场。

图 3.2 光场采集过程程序状态机

由于本程序采用状态机的方式进行组织，所以关键数据结构之一就是记录当前程序所处状态的枚举类型:

enum RecorderState{

ADJUST, CAPTURE, STOP

};

然后由于在采集开始的时候程序不知道用户会采集多少数据点，所以用一个可变长数组来记录当前线框球上每个圆片的中心坐标，用以判断当前交点是否已经被之前的交点所覆盖:

List<Vector3> centers;

当相机距离线框球中心过近，球面上圆片逐渐由绿色变为红色，当相机距离线框球过远，圆片逐渐由绿色变为蓝色，具体方法是初始时设定圆片在HSV颜色空间中的H值为0.33，当距离变小是减小H值，最小到0，当距离变大时增加H值，最大到0.67。算法的伪代码：

|  |
| --- |
| 算法3.1 圆片颜色调整 |
| // 计算距离与初始距离的比值，将0.8至1.2映射到0至1  // 如果过近或过远则停止采集  **i** |

然后在采集的时候需要计算相机和线框球的交点，求交方法使用了经典的射线和球求交的方法。如果交点已经被之前的圆片所覆盖了，那么就停止采集。如果线框球超出了屏幕矩形框的范围，就停止采集。如果不停止采集，就把相机的三维坐标和代表方向的四元数写入二进制文件，同时保存当前帧为jpg文件，具体算法：

|  |
| --- |
| 算法3.2 光场图片采集 |
| // 如果线框球超过矩形框的范围则直接返回  // 如果射线与线框球没有交点则直接返回  // 如果当前交点被之前圆片所覆盖则直接返回 |

在进行系统实现的时候，遇到过以下困难：

首先在进行调试的时候，如何将手持相机中拍摄到的画面传输到电脑上是一个问题。这里使用DroidCam应用，同时在手机上安装安卓版，在电脑上安装PC版，然后用USB进行连接，就可以实现画面的传输。

## 基于场景特征的采集方法

所谓的基于场景特征的采集方法指的是利用场景中的显著点的信息来计算相机三维坐标与旋转角度的方法。Wikitude的功能是以相机采集的视频为输入，自动提取场景中诸如边界点的显著点，然后利用帧与帧之间同一显著点的像素差以及手持相机中内置的陀螺仪与加速器等信息，计算相机的三维坐标与旋转角度。和ARToolkit不同的是，在Wikitude提供的接口中，当真实世界中相机位置发生改变时，程序场景中相机坐标保持不变，场景中所有物体的坐标与旋转角度发生改变。调用的时候是读取线框球的位置和旋转角度，然后假定线框球为世界中心，求解相机的相对三维坐标与相对四元数的旋转角度保存到二进制文件中。

如果想要利用基于场景特征的采集方法，场景需要满足以下条件：

1. 场景中必须有足够多的物体，有足够丰富的元素变化，否则无法提取出足够的特征点用于计算。
2. 场景中不能有太多透明的物体，透明物体反射折射规律复杂，特征点计算的方法不能适用于透明物体的计算。
3. 场景中需要有恒定的光照条件，物体不能剧烈运动，这些都是为了保证特征点的稳定性。
4. 场景不能太小，不适用于厘米级别的场景。场景太小的话受陀螺仪与加速器的精度限制，误差影响较大，降低计算结果的准确性。

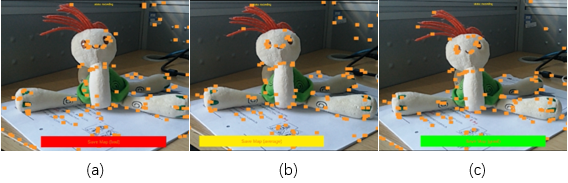
在采集之前，用户首先需要记录一个场景地图，具体方法是用户移动手中的相机，系统根据视频帧之间的不同计算显著点的深度等信息，只有当系统采集到足够多的特征点之后，系统才能够进行稳健的采集。系统会根据当前特征点的多少，给当前场景地图一个质量等级。一共有三种等级：好（good），中（average），差（bad）。最开始时等级是差，只有当质量等级变为好时，系统才可以进入下一阶段。如图 3.3 (a) (b) (c)分别表示等级为差中好时的情形。

图 3.3 基于场景特征的采集方法

## 种采集方法的对比

本系统中实现了两种方法，分别是基于标志的方法与基于场景特征的方法。两种方法各有利弊：

第一，基于标志的方法需要使用标志，这一方面给光场采集尤其是户外采集带来了许多的不变，另一方面，标志的存在也使得场景的美观程度受到影响。而基于场景特征的方法就没有这些缺点。

第二，采集范围方面，基于标志的方法相机可转动角度小，仅限于能够完整且清晰地观察到标志的范围，而基于场景特征的方法不受限制，相机可转动角度较大。

第三，精确度方面，基于标志的方法计算相机坐标精确度较高，整个采集过程流畅稳定。而基于场景特征的方法计算精度较低，在实验中发现有明显抖动现象，如图 3.5，（a）是基于标志方法最后的渲染结果，（b）是基于场景特征方法最后的渲染救过。基于场景特征方法最后渲染图中重影较明显。

第四，硬件要求方面，基于标志的方法对硬件几乎没有什么要求，无论是智能手机，还是通过USB与电脑相连的相机，都可以使用，程序既可以运行在手机上，也可以运行在笔记本电脑上。而基于场景特征的方法必须要运行在有惯性测量单元（IMU）的设备上，可以在一般的智能手机上运行但是没有办法运行在笔记本电脑上。

第五，计算资源上，基于标志的方法对计算资源消耗较小，而基于场景特征的方法采集和定位至少需要两个线程，实时运行至少需要双核CPU。

虽然基于场景特征的方法具有诸多优点，但是考虑到程序的稳定性和渲染质量，本系统将以使用基于标志的采集方法为主。



图 3.5 两种采集方法渲染结果对比

# 光场渲染

这里的光场渲染指的是以一系列照片以及照片所对应的三维坐标和四元数旋转角度为输入，给定任意相机坐标和旋转角度，生成出视点处看到的场景图。光场渲染的核心问题在于当给定任意相机坐标和旋转角度时，如何利用已有数据进行插值。考虑到采集质量等问题，本章的光场数据都是通过基于标志的方法采集得到的。

## 逐段线性的渲染方法

本系统中实现的逐段线性方法是针对[8]中逐段线性（piece-wise linear）方法稍作改动形成的。[8]只进行了原理性的说明，这里对本系统中的实现细节进行说明。逐段线性方法是一种基于三角网格的线性插值方法。在第一章中曾提到过光场的四维表示。称uv所对应曲面为相机所在曲面，st所对应曲面为像素成像曲面，逐段线性方法是利用相机曲面的三角坐标和成像曲面的纹理映射进行插值。程序首先需要指定一个平面作为成像曲面，然后将场景中所有相机投影到成像平面上，再将所有相机的投影点在成像平面上的二维坐标进行Delaunay三角化[12]。在渲染场景图的时候，对每一个像素点，找到它在成像平面上的对应点，找出该点所在的三角面片，三角形三个顶点上对应的相机，用纹理映射求出该点在三个相机中对应的颜色，然后利用该点在三角面片上的三角坐标进行对颜色进行插值。实际实现的时候，如果如上所述进行逐点计算速度比较慢，难以达到实时渲染的要求。考虑成像平面上的某个相机投影点，其对最终成像的影响范围被限定在了所有包含该投影点的三角形构成的多边形内，如图 4.1(a)，所以可以不用对每个像素点分别进行插值，而是对每个相机分别进行投影，然后将投影的图像进行混合（Blending）就可以得到最终的成像。这一过程由于可以利用gpu进行计算，可以大大加速渲染过程。具体操作方法是，对于某一个相机投影点，在顶点着色器中将该点的透明度值设为1，其他所有顶点的透明度值设为0，这样在片段着色器中，某片段的透明度值自然就对应了该片段在三角网格上的三角坐标的权重。然后取该片段的世界坐标，利用相机的世界坐标和旋转角度，计算视图矩阵和投影矩阵，就可以得到纹理坐标。之后利用纹理采样器和纹理坐标，求出该点对应的颜色，然后将该点颜色乘以透明度值进行返回。对每个相机都进行如上处理，同时将着色器设置为一加一混合，这样混合出来的图像就是最终的图像。

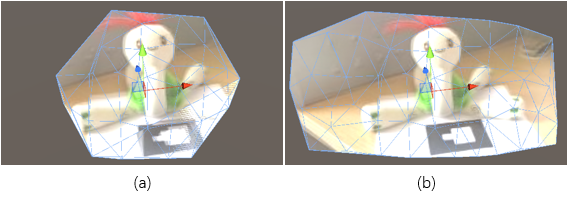
本系统中的实现和[8]中实现的一大不同在于三角化的方式。[8]中在初始化的时候取相机在球面上对应点的球坐标的方位角维度和仰角维度，进行二维的Delaunay三角化，然后将三角化的信息进行全局保存，投影平面上的三角化直接使用之前保存的三角化信息。在实际实验的时候，这种从球面上三角化直接应用于平面上三角化的方式效果并不好，容易出现三角边相交的情形。如图 4.2 (a)使用球面坐标进行三角化，右下角部分就出现了三角边相交的情形，影响渲染质量。(b)中每帧重新三角化就不会出现这种情形。在本系统实现中，在每帧开始的时候，对平面上的相机投影点直接进行Delaunay三角化，实验过程发现即使这样做也并不会影响程序的实时性。另外[8]中也没有提到对于一些边界情况的处理。在本系统实现中，如果相机投影点超出了当前视角成像范围，将予以删除。另外每个相机是有一定角度的，成像也是有一定范围的，比如当前视角是朝向x轴正方向的，某相机是朝向x轴负方向的，该相机很可能就不适合用来进行混合。系统中对相机成像不在视角成像范围的也会进行删除。

图 4.1 两种三角化方式的对比

由于渲染模块事先并不知道有多少个采集点可以正确投影到成像平面上，也不知道对投影点进行三角化之后会出现多少三角形，所以程序中使用变长数组存储这些信息:

List<Vector3> cameraPos;

List<Quaternion> cameraDir;

List<int> triIndices;

由于在进行投影，相机成像等计算中需要用到许多坐标变换的代数运算，实现中给三维空间坐标加一个维度1构成齐次坐标，然后使用4x4的矩阵表示平移，旋转，缩放等变换:

Matrix4x4[] cameraV;

对于逐段线性方法，首先需要设定一个观察相机viewCamera(类型为Unity内置的Camera类型)，设定视点坐标和视角方向。然后需要设定成像平面focalPlane（类型为Unity内置的Plane类型），成像平面的中心就是线框球的圆心，成像平面的法向就是相机坐标到线框球坐标的方向。然后需要将场景中所有相机投影到成像平面，如果投影点超出相机观察范围要进行舍弃。然后需要将投影点的三维坐标转化为平面上的二维坐标，用二维的Delaunay三角化进行处理成网格，最后使用4.1节描述的着色器以及Unity的渲染引擎进行渲染。算法伪代码：

|  |
| --- |
| 算法4.1 逐段线性算法 |
| // 与成像平面没有交点则跳过  // 投影点在相机成像范围之外则跳过 |

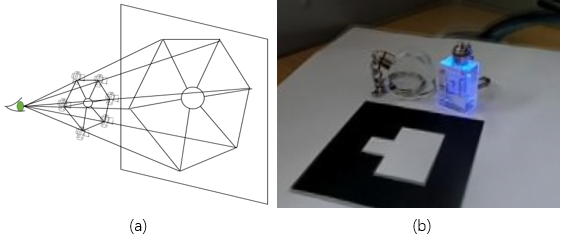
最终实现的渲染效果如图 4.2 (b)。图中场景使用了34张图片，该帧渲染时间为14.8ms。

图 4.2逐段线性的渲染方法

在上述算法实现的过程中遇到过以下问题：

由于上述算法中坐标变换较多，实际在实现的时候调试起来非常不方便，有时关于坐标问题出错很难定位错误出现在哪里，即使在控制台中将坐标输出，调试者也很看出坐标的问题。解决方法是可以利用Unity中的Gizmos模式。通过在代码中注入诸如Debug.DrawLine(start,end,color,time)的代码，可以把三维点是如何进行投影的清晰地绘制出来，这样就可以清晰地看出来问题出在哪里了。

另外就是开始的时候忽略了相机的投影矩阵，在进行渲染的时候总是会出现重影，原因在于手机相机的投影矩阵和程序中默认设置的投影矩阵是不一样的。解决方法是在录制之前将相机投影矩阵保存到二进制文件中，然后在渲染之前将投影矩阵加载到内存，再对所有的相机进行设置。

还有就是逐段线性方法在进行混合的时候也遇到了问题。开始的时候在片段着色器中将透明度设置在颜色的第四位坐标上，然后使用经典的透明度混合配置:

Blend SrcAlpha OneMinusSrcAlpha

直接这样混合效果很不好，混合的结果偏暗。原因在于最开始的时候屏幕是黑色的，上述方法把最开始屏幕上的黑色也进行混合，于是导致结果偏暗。解决方法是改变透明度混合配置为：

Blend One One

然后在片段着色器中将颜色乘以透明度值，不用设置颜色第四维坐标就可以。

## N近邻的渲染方法

本系统中实现的N近邻的渲染方法是基于[7]中的方法实现的。在进行光场渲染之前，首先需要场景的几何信息。几何信息即可以是一个平面，也可以是一个球面，也可以是场景中物体的真实三角网格，场景的几何信息越接近真实的场景几何形状，渲染的结果越精确。在渲染过程中，当前视角观测到的每个像素都对应一条光线，将光线反向，与场景的几何形状相交于一点，称之为物点。然后连接物点与所有的相机，在连接所得的线段中找出N条与视角发出射线角度最接近的线段，找出对应的相机，求出物点在相机中对应的颜色，对其进行加权平均就可以得到。如图 4.3(a)描述的是就上述过程，D为观测点，P点为物点，C为相机。。至于权重的计算，角度相差约大权重越低。系统中实现的时候，角度的差用射线方向向量归一化后的欧式距离近似，权重用1减距离归一化后近似。

对于N近邻方法，viewCamera与focalPlane的设置与逐段线性方法完全一样，这里不再赘述，不过N近邻方法需要对最终成像图片中每个像素，找到其对应的射线，然后求出射线与平面的交点，再将交点与所有相机相连找出角度最近的N个相机，然后根据角度差确定每个相机对应的权重，最后利用纹理映射对N个颜色值进行加权，算法伪代码如下：

|  |
| --- |
| 算法4.2 N近邻算法 |
| // 对每一个像素进行遍历  // 找到角度最近的N个相机  // 对颜色进行加权平均 |

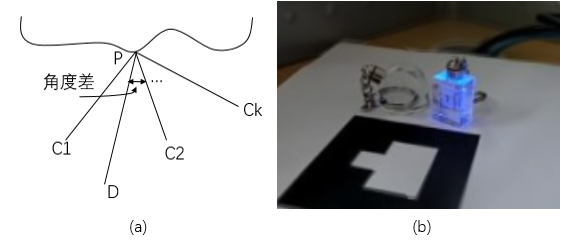
最终实现的渲染效果如图 4.3(b)。图中场景同样使用了34张图片，该帧渲染时间为1565.4ms。

图 4.3 N近邻的渲染方法

## 两种渲染方法的对比

逐段线性的渲染方法和N近邻的渲染方法本质上是类似的。

从st曲面角度来看，逐段线性方法需要在场景中放置一个成像平面，这个成像平面就近似是N近邻方法中的场景几何。只不过N近邻方法不要求场景几何一定是平面，也可以球面，真实场景的三角网格，或任意形状。场景几何越逼近真实场景几何，渲染结果图越逼真。

从uv曲面角度来看，N近邻方法取的角度最接近的N个相机的信息进行加权平均，逐段线性方法取的是成像点所在的三角面片的三个顶点对应的相机信息进行加权平均，三角面片的三个点就可以近似看成在场景几何为平面的时候找的三个角度尽可能接近的相机进行加权平均。

从物理意义上来看，N近邻方法选用N个角度最接近相机信息进行加权平均，直观上理解符合光的反射定律， 有比较明确的物理意义。而逐段线性方法选用三角面片上三个顶点对应的相机信息进行近似，没有什么明确的物理意义。

表 4.1 兔子布偶场景中两种渲染方法的效率比较

|  |  |  |
| --- | --- | --- |
| 采集张数 | 逐段线性法每帧渲染时间  (ms) | N近邻方法每帧渲染时间  (ms) |
| 10 | 15.1 | 964.0 |
| 20 | 15.2 | 1270.8 |
| 30 | 15.0 | 1490.1 |
| 40 | 15.2 | 1707.8 |
| 50 | 13.5 | 1950.9 |
| 60 | 19.8 | 2337.8 |

从适用范围来看，N近邻方法由于有较明确的物理意义，适用范围较广，可以适用于各种场景可以进行大角度的拍摄，而逐段线性方法由于将场景近似看成是一个平面，这种近似带来渲染的不精确，在采集的时候如果相机角度变化过大渲染出来的图片也会出现重影。

从渲染效率上来看，逐段线性方法利用三角网格进行渲染，可以非常容易地利用GPU的特性进行加速渲染。N近邻方法则需要对每个像素的查找角度最接近的N个相机，效率比较低。虽然理论上可以只选取一些关键点求N近邻并求出权重，然后用关键点对其他点进行插值，实际上如果这样做了那就和逐段线性方法很接近了，所以系统没有重复进行实现。如表 4.1所示，逐段线性方法几乎可以做到线性渲染，N近邻方法做不到。值得注意的是渲染时间不仅和采集张数有段，操作系统状态等相关，逐段线性方法在实验所用的采集张数范围下随采集张数变化渲染时间变化不明显，N近邻方法随采集张数增长而明显增长。

从渲染质量来看，逐段线性方法在点与点之间进行线性插值，所以可以保证相机混合场（camera blending field）的一阶连续性，而N近邻方法对每个像素查找角度最接近的N个相机，没有连续性的保障。所以理论上N近邻方法在渲染结果上更容易出现缺陷。如图 4.4显示的是不同场景不同方法的渲染结果的比较。奇数列显示的是逐段线性方法的渲染结果，偶数列显示的是N近邻方法的渲染结果。图中一共给出了(a)(b)(c)(d)四个例子。每个例子中第一行显示的是相机混合域，第二行显示的是场景渲染结果，第三行显示的是局部放大图。其中相机混合域指的是给相邻相机赋予不同的颜色值，然后在最终渲染结果里相机是如何插值的，就将对应的颜色值如何进行插值。直接看场景渲染图难以看出渲染方法的差异，可以从相机混合域进行更好地说明。在对应的相机混合域中可以看到，使用逐段线性方法，相机之间呈现线性过渡，过渡比较自然，而使用N近邻方法，相机的作用域之间有明显的分界线，有许多非连续过度的地方。体现在局部放大图可以发现，使用逐段线性方法的渲染结果看起来清晰度更高，而使用N近邻方法相对来说看起来更加模糊，也更容易出现错位重影等缺陷。

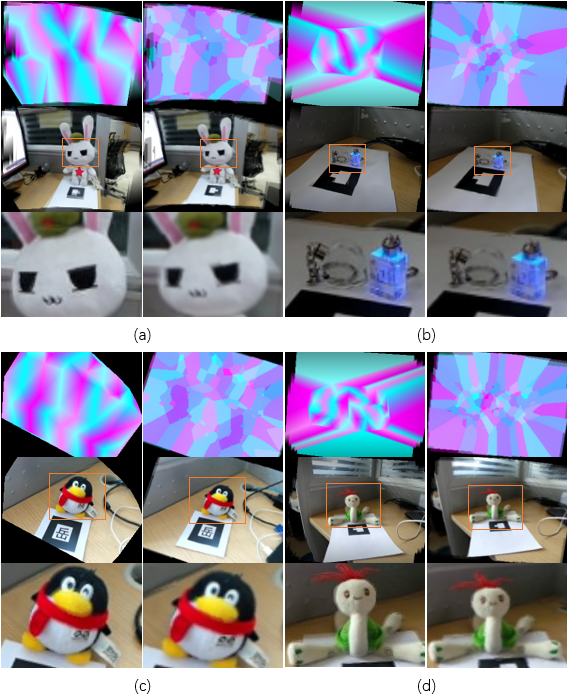


图 4.4 不同渲染方法的效果比较

虽然N近邻方法有明确的物理意义使用范围广，但是从渲染效率和渲染质量上来考虑，本系统还是主要使用逐段线性方法。

# 结论

## 工作总结

本文实现了一个基于手持相机的光场采集系统。为了实现这一系统，主要主要尝试解决了两大问题，第一是如何辅助用户使用手持相机进行光场采集，第二是针对用户采集来的非结构化数据如何进行光场渲染。针对光场采集，系统中实现了两种光场采集的方式，分别是基于标志的采集方式和基于场景特征的采集方式，两种方式各有优点，适合在不同的应用场景下使用。针对光场渲染，系统中实现了两种光场渲染的方式，分别是逐段线性的渲染方式和N近邻的渲染方式，两种方式各有优点，也需要分不同的应用场景使用。

## 存在问题与未来工作

首先是光场采集方面，在利用手持相机进行光场采集的过程中，如何对手机进行定位仍然是一个难题。基于标志的方法需要使用标志，携带很不方便也影响场景美观程度，当相机离标志过远就没有办法进行定位。基于场景特征的方法虽然舍弃了标志，但是程序稳定性较差，实验过程中有明显抖动，影响最后渲染结果。未来的一个工作方向之一就是改进对手机定位的准确性。随着增强现实技术的推广，越来越多的手持设备开始配备深度摄像头，双摄像头等结构，比如谷歌的Tango项目[13]。未来可能定位这个问题上可以做得更好。

然后是光场渲染方面，针对手持相机采集来的光场数据，如何获取场景的几何信息仍然是一个难题。对于逐段线性方法来说，相当于假定场景的几何形状为一个平面，对于小范围的光场采集来说可以实现比较好的渲染效果，但是如果想要进行大范围的光场采集的话就很难有很好的渲染效果了。对于N近邻方法，虽然能够处理大范围场景的光场数据，但是其假定场景的几何信息是已知的，如何获取场景中的几何信息还是一个需要解决的问题。

# 插图索引

[图 1.1 光场的四维函数表示 1](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065168)

[图 2.1 光场采集系统框架图 4](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065169)

[图 2.2 光场采集系统概述 6](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065170)

[图 2.3 视角和线框球的相对位置 7](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065171)

[图 3.1 显示的是基于标志的采集方法 9](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065172)

[图 3.2 光场采集过程程序状态机 10](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065173)

[图 3.3 基于场景特征的采集方法 13](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065174)

[图 3.4基于场景特征的采集方法 13](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065175)

[图 3.5 两种采集方法渲染结果对比 14](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065176)

[图 4.1 两种三角化方式的对比 16](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065177)

[图 4.2逐段线性的渲染方法 18](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065178)

[图 4.3 N近邻的渲染方法 21](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065179)

[图 4.4 不同渲染方法的效果比较 23](C:\\Users\\gengzl\\Downloads\\综合论文训练-耿正霖.docx" \l "_Toc452065180)

# 表格索引

[表 4.1 兔子布偶场景中两种渲染方法的效率比较 14](#_Toc451804965)



# 参考文献

[1] Levoy M, Hanrahan P. Light field rendering ACM, 1996

[2] Levoy M. Light fields and computational imaging. Computer, 2006(8): 46～55

[3] Chen S E. Quicktime VR: An image-based approach to virtual environment navigation ACM, 1995

[4] Shum H, Ng K, Chan S. A virtual reality system using the concentric mosaic: construction, rendering, and data compression. Multimedia, IEEE Transactions on, 2005, 7(1): 85～95

[5] Wilburn B, Joshi N, Vaish V, et al. High performance imaging using large camera arrays. ACM Transactions on Graphics (TOG), 2005, 24(3): 765～776

[6] Ng R, Levoy M, Brédif M, et al. Light field photography with a hand-held plenoptic camera. Computer Science Technical Report CSTR, 2005, 2(11): 1～11

[7] Buehler C, Bosse M, Mcmillan L, et al. Unstructured lumigraph rendering ACM, 2001

[8] Davis A, Levoy M, Durand F. Unstructured light fields Wiley Online Library, 2012

[9] Unity-Technologies. ShaderLab Syntax, 2016: 2016

[10] . ARToolkit Documentation: 2016

[11] Wikitude-Gmbh. Unity documentation, 2016: 2016

[12] . Delaunay Triangulation in Unity, 2014: 2016

[13] . Google's Project Tango: 2016

# 致 谢

感谢张慧老师对我论文的知道。感谢黄经纬同学和我进行的有关光场的讨论，给我的工作拓展了思路。

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，独立进行研究工作所取得的成果。尽我所知，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享有著作权的内容。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。

签 名：\_\_\_\_\_\_\_\_\_\_\_\_\_ 日 期：\_\_\_\_\_\_\_\_\_\_\_\_

# 附录 A 外文资料的调研阅读报告

In 1996, Marc Levoy and Pat Hanrahan first brought the concept light field to the computer science community, in their siggraph paper Light Field Rendering. The advantage of light field compared to other image-based rendering (IBR) techniques is that no depth information or correspondence knowledge about images is required.

Light field is defined as the radiance at a point in a given direction, which can be described using 5D plenoptic function. But because the radiance along a line is constant (if there is no blocking object), this function can be reduced to 4D function to remove the redundancy.

In their paper, the authors recommended to use two planes (called light slab) for the representation of the light field. The coordinate system on the first plane is (u, v) and on the second plane is (s, t). The radiance along an arbitrary line passing through these two planes is represented by L (u, v, s, t).

In practice, we usually call the uv plane camera plane and the st plane focal plane. Light field can then be represented by 2D array (camera plane) of images (focal plane). So we can use an array of cameras to capture light fields.

When reconstructing image at an arbitrary viewpoint, there are two steps in the process: step 1 is to compute the line parameters (u, v, s, t), and step 2 is to resample the radiance at the line parameters. The first step can be easily done using texture mapping. As for the second step, they simply interpolate the 4D function from the nearest samples in their implementation.

Because 4D light field array is very large, to make creation and transmission possible, they also proposed a compression method. In their application, they first applied vector quantization on the light field data. Typically, they used 4D tiles of the light field, yielding 48-dimensional vectors, and generated codebook using mean-squared error(MSE). Then they further compressed the data using entropy coding, which can be easily done using gzip. Typical compression rate is 100:1.

In Marc et al.’s work, the cameras have to be placed with fixed intervals on a fixed plane, which restricted its application. In 2001, Chris et al. first proposed a real-time rendering method that does not require the input camera to be restricted to a plane, or any specific manifold.

When reconstructing a desired ray, Chris et al. proposed to use source image rays with the most similar angle to the desired ray, instead of the closest ray in (u, v, s, t) parameter space.

The most important concept proposed in their paper is camera blending field. It describes how each source camera is weighted to reconstruct a given pixel.

Some notations: given desired ray , intersects with the object at some frontmost point p. is the center of each camera. is the ray between p and .

The difference between and is defined as:

accounts for resolution difference and accounts for resolution difference.

The larger the difference, the smaller the weight should be:

Because rays may fall outside of the source camera, we have

is a function that continuously goes to zero as approaches the edge.

Find k largest . Finally normalizing over i yields the blending weight:

To achieve real time rendering, in their implementation, they only compute the camera blending at discrete set of points, and interpolate the camera blending over the image.

In 2012, Ade et al. were the first to design a system that allowed user to capture light field using mobile phones. Although in Chris et al.’s work, unstructured lumigraph rendering had been proposed, it is still very challenging for users to capture light fields. Because users usually take images by scanning the subject in a back-and-forth manner, it is crucial to guide the user to make sure the scanline starts at the correct offset from previous one. Without guidance, the distance between the camera and subject can vary discontinuously, which usually will give rise to artifacts. Also it is better to give users an idea how much has been covered.

Ade et al. designed a capturing process and used the augmented reality library SLAM to solve the above issues. In their system, the user first points the camera to the scene and selects a subject to capture. The system then records new images whenever it determines that it is viewing an under-sampled region of light field. They guide the user with a viewpoint coverage map to help them achieve dense coverage.

One of the main contributions of their work is to propose a coverage criterion, called reprojection error. Consider a captured view s, and a new view n. We need then to determine when the view n is not covered by s. Consider a cone of ambiguity associated with a pixel V\_s of s, which means all the object points in this cone may contribute to the pixel color of V\_s. Then consider the intersection part between the core and the object sphere. The reprojection error is then defined as the longest distance (in this case AB) projected to the new view n. If this value is larger than a threshold, we determine that n is no longer determined by s and a new image is recorded. This new criterion is sensitive to both parallax error and resolution difference.

Another contribution of their work is a fast rendering technique that can achieve piecewise-bicubic at the limit. Here we only introduce the piecewise-linear reconstruction because piecewise-bicubic reconstruction is a simple modification of piecewise-linear one by extending the projection to 2-ring and apply the Loop subdivision rules. They first build Delaunay triangulation on the viewpoints. Then instead of finding the k nearest cameras, they intersect the desired ray with the triangle mesh and interpolate using barycentric coordinates. To accelerate this process, they first project the triangle mesh to the target view. For a given vertex, it has a stored view with it. At the vertex, the weight is 1 and at the neighboring vertex, the weight is 0. This is to say a given stored view can only affect a triangle fan. So they can efficiently render the view using texture mapping and alpha blending.

综合论文训练记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学生姓名** |  | **学号** |  | **班级** |  |
| **论文题目** |  | | | | |
| **主要内容以及进度安排** | **指导教师签字：**  **考核组组长签字：**  **年 月 日** | | | | |
| **中期考核意见** | **考核组组长签字：**  **年 月 日** | | | | |
| **指导教师评语** | **指导教师签字：**  **年 月 日** | | | | |
| **评阅教师评语** | **评阅教师签字：**  **年 月 日** | | | | |
| **答辩小组评语** | **答辩小组组长签字：**  **年 月 日** | | | | |

**总成绩：**

**教学负责人签字：**

**年 月 日**