北京大学

本科生毕业论文

题目：基于触摸屏的立体图像标注系统

姓　　名：张吉安

学　　号：00848160

院　　系：信息科学技术学院

专　　业：计算机

研究方向：计算机应用技术

导师姓名：王亦洲

二〇一二年五月

基于触摸屏的立体图像标注系统

张吉安　计算机
导师姓名：王亦洲

摘要

由于3D播放设备以及3D电影和电视的日益普及，将普通的视频转换为3D视频也成为了一个极其具有实用意义的研究课题。随着3D技术的日益普及，立体视频播放设备开始普及，使得普通用户也可以使用。但是另一方面立体视频的缺乏导致实际上普通用户还是要依赖于专业的立体视频制作公司。我们希望能够开发出普通用户也可以使用的工具，让人人都可以制作自己的立体视频。因为技术水平以及一些固有的难题，目前的3D视频转换仍然不可避免地要加入人工交互的操作使得生成的3D视频尽量逼近真实3D视频。但是现有已知的交互标注的工具，它们对于图像的标注部分无一例外基于传统PC，这对于提高标注效率是不利的。所以我们希望开发出一套基于触摸屏设备的标注工具，以提高标注工作的效率。

在毕业设计的过程中，我以已有的标注模型和图像分割算法为基础，完成了基于触摸屏的图像三维信息标注工具的设计，并且基于Android 移动平台开发了一款图像标注工具。

关键词：触摸屏，3D，人机交互，Android

An interactive system for image stereolization on touch screens

Zhangji'an Computer Science
Directed by Prof.Wang Yizhou

Abstract

Since the popularization of 3D playback device and 3D video, converting stereoscopic video from traditional ones has become a practical topic in movie industry. Due to the technical limitations and some immanent problem, we have to do this conversion with the help of human interaction in order to make the result more close to real ones which shot with real stereoscopic camera. But for all the known convert tools now, their image annotation parts is based on traditional PC, which is harmful to raise work efficiency of labeling. So we hope to develop a annotation tool based on touch screen that benefits labeling efficiency.

I design an annotation tool based on touch screen and develop it on Android mobile device as my graduation project.

Keywords: Touch Screen, 3D, Human-Computer Interaction, Android

目录

第一章　项目介绍

1.1　视频3D转制现况及前景

3D电影技术最早产生于19世纪90年代末期，英国电影先驱威廉·弗里斯格林(William Friese-Green)发明了使用两台播放机放映3D电影的技术，这是可以考证的最早的3D电影的播放技术。而世界上第一部真正的 3D长片则是1952年的《非洲历险记》。由于技术手段不足，拍摄成本高昂等原因，3D电影始终没有摆脱在电影工业中的边缘地位。一直到进入21世纪，随着相关技术手段的完善，出现了诸如《极地特快》这样有质量的3D长片，3D视频技术才渐渐重新出现在人们的视野中。

在被称为“3D电影的新元年”的2009年，涌现了诸如《飞屋环游记》《冰河世纪3》以及堪称电影史伟大里程碑的《阿凡达》。2009年底由詹姆斯·卡梅隆(James Cameron)导演的3D电影《阿凡达》的上映则使得原先渐渐回暖的3D电影市场火爆异常，使得大家不再将3D电影视为游乐场中的一种游乐项目，而是真正将其视为电影工业技术中的一支。

而随着《阿凡达》获得巨大成功以来，各个制片公司都纷纷为最新的主打电影采用3D技术。经过不完全统计，2012年在中国上映的3D电影就有22部之多。而随着3D电影带来的热潮，各类3D相关的产品也大行其道。2010年迪士尼公司旗下的美国娱乐和ESPN 宣布成立了全球首个3D电视频道ESPN 3D，美国探索传播公司旗下的探索发现频道也在同年同索尼公司和IMAX公司联手成立了3D频道。而我国的首个3D频道也在2012年元旦开始播放节目。在播放设备方面，三星，松下，康佳等国内外公司都推出了3D平板电视，任天堂，LG和HTC等公司推出了支持裸眼3D 显示的游戏机和移动电话，英伟达公司推出了基于PC的3D立体显示技术3D Vision。在娱乐方面则出现了诸如《半条命2》,《孤岛惊魂2》等支持3D效

果的游戏。

1.2 项目背景

在我们回顾3D视频相关发展的时候可以注意到，走在前沿的相关技术往往是3D显示技术。而真正的3D视频制作技术的发展却跟不上显示设备的发展。事实上现有的3D视频拍摄的成本是十分昂贵的，《阿凡达》的制作成本高达3亿美金，而《变形金刚3》的3D版本仅仅在制作费的支出上就要高出2D版本3000万美金。所以为了应对各种需要较多3D片源的同时要求控制成本的情况(例如3D电视频道)，发展将普通视频转换为3D视频的技术就显得十分必要。

不仅如此，随着3D电影越来越受到欢迎，将原先的经典电影转换为3D电影也成了一种潮流。最成功的例子就是经典电影《泰坦尼克号》的 3D转制版，在2012年4月上映之后在全球获得了3.3亿美元的票房。而我国经典动画片《大闹天宫》的3D转制版也在同年上映。但是《3D 泰坦尼克号》巨大的成功背后却是高达1800万美金的制作成本。所以研发低制作成本的3D转制技术是3D产业发展的关键所在。

目前存在的3D转制技术可以分为全自动和半自动两种，其中全自动技术所产生的效果和真实3D视频观影感受相去甚远。半自动技术虽然转换效果较好，但是由于加入了人工交互的环节所以存在效率不高的问题。在之前开发2D到3D视频交互式转换系统的过程中就发现由于传统PC在图像标注的交互中存在的不够直观，以及频繁的鼠标操作容易给工作人员带来疲劳而导致工作效率下降的问题。

基于触摸屏的标注工具的设计和开发就是针对上述问题，力图使用触摸屏交互直观的特性解决由于传统PC交互所带来的效率低下的缺陷。希望能够通过新的交互方式提高人工标注的效率，从而提高软件整体的工作效率。

1.3 项目具体工作

在本项目中我的工作主要是设计基于触摸屏的单幅图像的标注工具及其实现。

第一，根据视频标注操作的特性结合触摸屏交互的特点，整理出视频交互标注操作的具体模型。依据所整理出的模型为标注工具设计完整的软件模型。

第二，在Android平台的移动设备上实现第一步设计出的软件模型，根据实现的结果对于该软件模型的可行性做出评估。

第二章　系统设计概述

2.1　视频标注系统整体概述

下图是一个成熟的视频标注系统的整体结构。 可以看出，在一个视频标注系

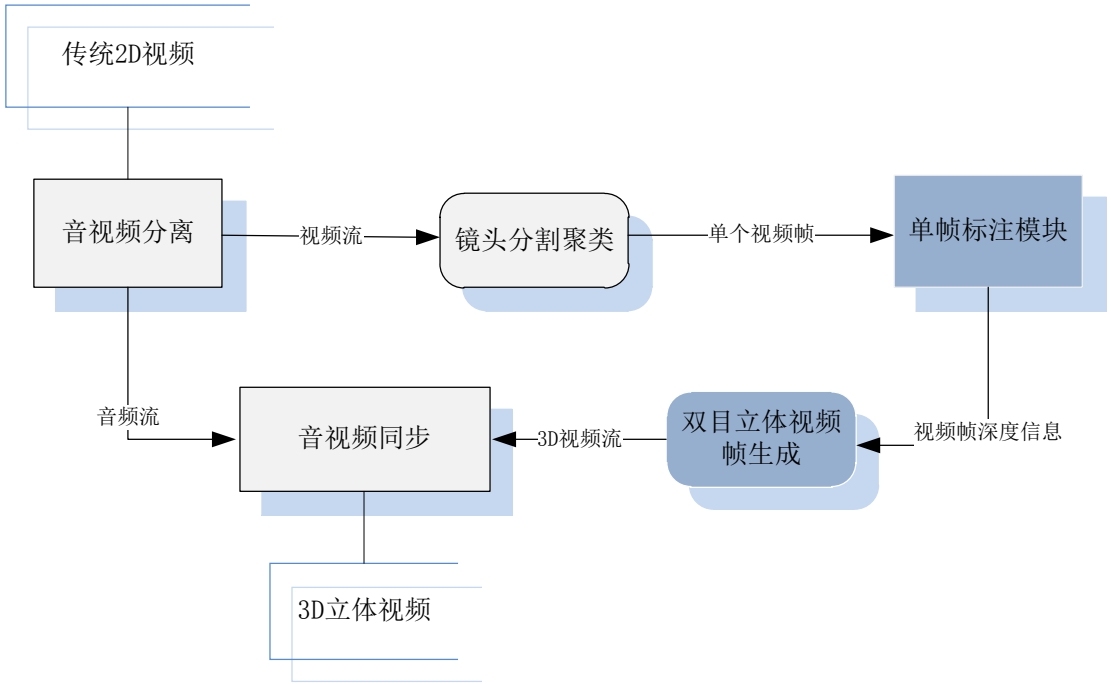

图 2.1: 系统流程

统中，已经存在完善且成熟的技术的有音视频分离和音视频同步部分，而镜头聚类分割和双目立体视频的生成也有了成熟的自动转换的技术。

所以实际上交互式的视频标注系统中的交互标注操作几乎都完全花费在单帧的标注操作上，事实上在我们已有的基于传统PC上的标注工具的使用中发现，除了视频的单帧标注之外，其余的操作可以完全地实现自动化。

2.2 视频帧交互式标注模块

单帧的交互式工具的目的是为了在单幅图像上标记出图像的深度信息，生成对应于图像的depth map。同时这也是本项目工作的重点。

目前针对视频帧的标注模块大致可以分为两个方面，分别为前景部分的深度标注和背景的深度标注。其中背景的深度估计对于估计的准确度和物体互相之间的深度的把握较前景的要求要低，而前景的深度估计则在物体的深度变化和物体之间的深度关系上和观影者的感受较为密切。也就是说，在背景的估计上我们可以使用较为粗略但是效率较高的方法，而在前景的标注上则要求尽量能够反映前景物体的深度信息以带给观影者较为真实的3D观影感受。 目前可利用的背景估

图 2.2: 自动/手动深度估计的对比

计的方法有两种，一种是利用Stage Model[1]来对于背景场景建模，另一种则是利用自动的深度估计来估计背景的深度。

从图2.2中可以看出，由于仅仅对于场景的几何结构做出了估计，Stage Model对于背景的估计效果实际上是很不好的。例如背景中一些有层次的效果在Stage Model中体现不出来。相反地，对于背景中的物体，背景深度自动估计的方法却可以估计出比较真实的效果。所以最终我们选择了以背景深度自动估计的方法来做背景深度的标注。

而同样来自于图2.2的信息也可以看出对于自动的深度估计在细致的部分上的表现效果很差，这样在观影的时候由于观众的注意力集中在前景物体上，所以微小的缺陷也容易给观众带来不适。

所以在前景的标注上我们需要采用手工标注深度的方法使得在前景部分的深度估计较为准确，从而得到更为精确的深度图，在转换成立体视频的时候更加逼近真实拍摄的立体视频。

2.3 系统其他部分介绍

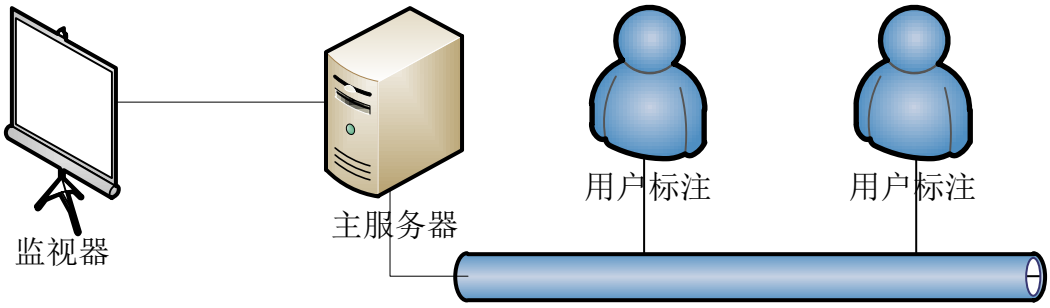

图 2.3: 系统架构

- 音视频分离与合成模块:

  目前有两种比较成熟的技术分别为微软公司的DirectShow技术。DirectShow是目前商业上成熟的视频播放软件与视频处理软件所常用的技术。另一种为FFmpeg项目。FFmpeg是一个开源的免费的跨平台音视频流方案。它属于自由软件，根据所选择应用的组件不同需要遵守LGPL 或者GPL许可证，是众多优秀开源视频处理软件所采用的方案。

在这两种方案之中Directshow是一个比较成熟的方案，优点在于创建简单的音视频处理流程

- 镜头分割和聚类模块和立体图像生成：

  这两项工作已经有成熟的可运行的工作可以借鉴。这里就不再赘述。

图2.3是我们构想中的标注系统的示意图，可以看到最终的设想架构是在传统具有高计算性能的服务器上运行音视频分离，背景深度图自动估计和立体视频生成等需要消耗大量计算量的模块。而客户端则采用本项目中设计的标注模块运行单帧的标注工作。为了提高计算资源的使用率可以将系统设计成分布式系统，多个客户端共享一个高性能服务器。

第三章　标注模型的设计

在设计标注模型的时候，我们充分考虑了触摸屏的操作特性，综合了计算方法的计算资源开销和人工操作的时间开销：

3.1　基于触摸屏的标注模型

在作为标注媒介方面，和传统的PC操作触摸屏具有以下几个优势:

首先是触摸屏的操作直观，由于图像的深度信息标注类似于在图上作画，而基于传统PC使用鼠标点击特定图片已达到标注信息的目的是非常不直观而且效率低下的。

与此同时，触摸屏的标注就要显得直接的多，可以将屏幕作为画布，在屏幕上直接点击对应点达到标注的效果。虽然面临着类似于手指标注不够准确的问题，但是这个问题可以用触控笔和屏幕的结合来解决。

其次多点触控(但是据悉苹果公司已经获得了该项技术的专利，将其应用到产品上可能带来潜在的版权风险)以及手势识别相较点击按键更为便利。

在我们之前的软件实际操作中发现过多的按键对于工作人员的效率影响很大，从而降低工具的标注效率。另一方面如果更多地使用手势来代替按键，就能够将更多的屏幕面积用于显示标注的图像，这样就能够减少由于不必要的图像缩放造成的标注之外的操作。而且使用手势来代替按钮从软件的使用上就更为简易，对于培训新的工作人员上手操作是十分有利的。

第三，触摸屏能够避免长时间操控鼠标带来的伤害

事实上在设计标注工具的时候我们需要考虑到该工具的易用性，由于在电影工业中实际的图像立体信息的标注需要工作人员长时间的专注于重复的信息标注工作。长时间操作鼠标容易给人带来疲劳的感觉。另一方面长期长时间在工作时

使用鼠标容易给工作人员的身体健康尤其是指关节带来不利的影响，甚至导致患上相关疾病。

然而触摸屏由于操作更贴进人体正常操作，所以对于减缓工作疲劳和减轻相关疾病的概率都是有利的。这从另一个方面也提高了标注工具的工作效率。

和2.2节中所述的一样，我们将整个标注模型分为了前景标注和背景标注。整体标注流程如图3.1： 可以看出，我们在单帧的前景标注中分别生成了两幅depth

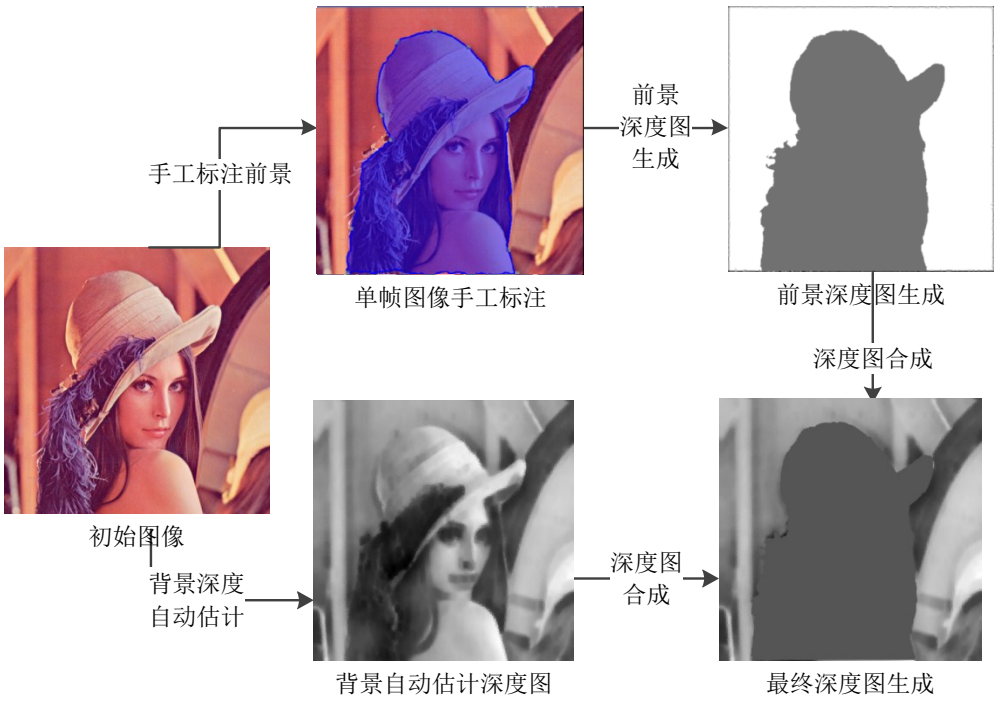

图 3.1: 单帧标注模型的设计

map，最后将其合成为最终的depth map从而达到标注的目的。事实上在这里大量的背景标注工作被交由计算机自动生成。因为背景标注的效果虽然不如手动的标注精确，但是经过我们的实践标明只要保证背景的深度估计在时间序上是连续的就不会给观影者带来不适。这主要是由于观众的注意力主要集中于深度变化较大的前景物体上。

由于背景的深度自动估计已经有了成熟稳定的算法，所以本项目的主要目的是在解决前景的标注问题。

3.2 前景标注方法

前景标注有很多方法，其中最传统的是手工标注出每一个像素点的深度值。这个方法的优点是易于实现，而且如果能保证人工标注完全准确的话，理论上可以得到ground truth。但是这个方法的缺点显而易见，那就是工作量非常大，工作效率非常低。而且如何保证帧之间的标注一致性也是一个问题。

除去逐点标注的方法，由于前景物体大多是独立于背景的一个物体，所以可以将前景图像切分出来，并且给前景图像单独赋予一个深度值的办法来将一整块的代表前景物体的像素都赋予一个相同的深度值用来代表物体的深度。这种图像切割的方法大概有两种成熟的方法可以选择。

第一种较为实用的方法是使用Graph Cuts[2]的方法。其使用方法如图3.2所示，需要先设定一个矩形框，指明前景的大致位置。并且在矩形框中人工地标注上一些属于前景和背景的位置来引导图形的分割。实际上不需要将所有前景和背景都标注出来，只需要使标注能够覆盖大致的前景和背景的范围即可

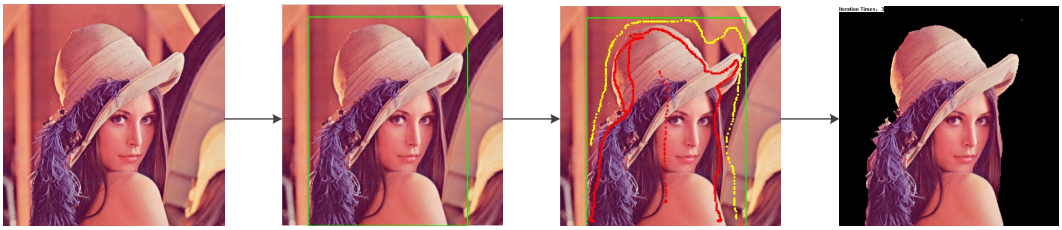

图 3.2: Graph Cuts标注流程

第二种方法是使用Intelligent Scissors[3]的方法来对图像进行分割。如图3.3所示，可以看到Intelligent Scissors类似著名图像处理软件Photoshop的磁性套索。实际上这不是一个整体的图像切分的算法，而仅仅是图像不规则边缘的切分算法。

为了使Intelligent Scissors能在我们的算法中切分出一个封闭区域，我们对于它的使用做如下的限制：在初始分割一个前景物体的时候可以从前景边缘的任意位置开始标注。而只要这个前景物体被确定了第一段边缘之后，剩下的所有分割操作就只能是在这个已有的边缘上以这条边缘划定的不规则曲线的端点为基础扩展。

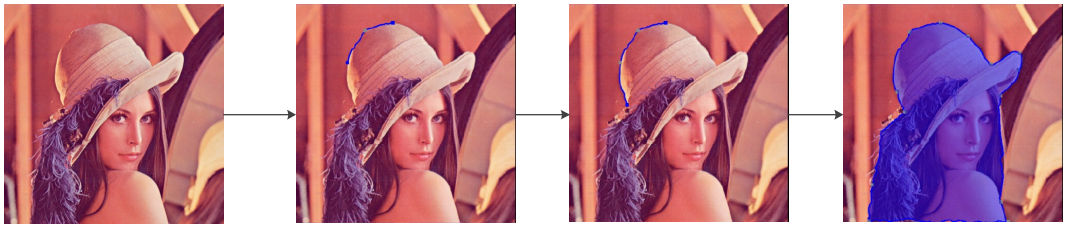

图 3.3: Intelligent Scissors标注过程

如图3.3中所显示的那样，要延伸在标注中的前景物体的边缘的方式就是从当前边缘曲线的两端开始扩展。这样就能够保证最后当用户将当前边缘曲线的两个端点连接起来的时候一定能够得到一个封闭的区域作为前景物体的标注区域。具体的标注如3.4 所示的那样，用户需要将曲线$\widehat{AB}$作为一个边缘分割出来就只需要先点击A，再点击B，则算法会自动将最贴近图像边缘的曲线画出来。

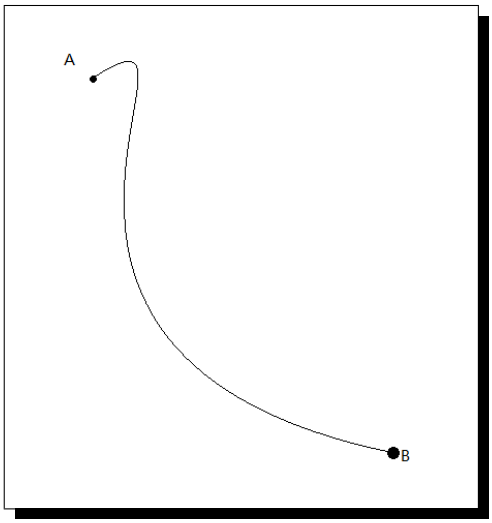

图 3.4: 切割一条曲线

基于Graph Cuts的分割和基于Intelligent Scissors的分割都能够达到我们对于前景物体的抠像的目的。这里我们将两个方法的特点做出一个对比：

基于Graph Cuts的分割可以一次性地分割出整个前景物体的轮廓，这样对于用户来说省时省力。而且分割出来的图像一定是一个封闭的图形。这个特性使得我们在编写代码的时候不需要考虑闭合轮廓的这个过程。并且我们可以一次性得到前景和背景的mask矩阵。

而基于Intelligent Scissors的分割实际上专注于解决物体的轮廓问题。这个方法的计算量较小，但是在理想的情况下来看操作的次数相较Graph Cuts。因为首先Intelligent Sissors不可能一次性将前景物体分割出来。而且由图3.4我们也可以看出，实际上每次标注的两点不能相隔太远或者用来标注如图3.5的情况：我们需要标注曲线a为边缘，但是点击A，B两点之后返回的很大概率(取决于实际图像的

情况)会是c，以及b。所以为了得到曲线a就需要手工在a上多标注一个点。实际上如果一个前景物体包含了许多类似曲线 a这样弧度很大的边缘部分，那么我们就需要标注每一个顶点才能完整地得到正确的曲线。

而且相比于Graph Cuts松散的标注要求，Intelligent Scissors完全信任用户的标注，所以需要用户标注的每个控制点都是准确的。

不过如果考虑非理想的情况就可以看出Graph Cuts的缺点：这个算法不能100%保证切割出来的前景和背景是一定准确的。或者说很有可能标注的每一个帧上都有细小的误差而需要调整。而调整部分又需要使用Intelligent Scissors算法重新计算不准确部分的边缘。

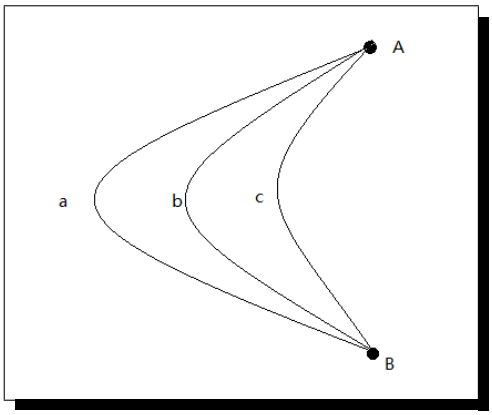

图 3.5: 多条曲线过两点的情况

实际上，我们标注的要求是得到前景物体尽可能精确的前景图形的闭合边缘，至于视频帧上每一个像素点是属于前景亦或是属于背景虽然是我们需要求解的问题，但是一旦我们得到了闭合边缘曲线，那么曲线所包含的部分自然就是前景物体了。另一个方面如果得到了前景和背景的范围根据它们相交的部分也可以得到边缘曲线。

不过由于边缘部分的前背景划分可信度是最低的，所以往往我们最能够信任的是我们不关心的前景中心部分的标注(很少有情况) 会将前景中心的像素错分成背景，我们最关心的边缘部分的划分反而是最不可信的(这会在3.3.1中详细叙述)。

基于上述的考虑，结合我们的设备选取的是触摸屏的情况，最终选择了Intelligent Scissors作为前景分割的工具

3.3 标注中使用的算法简介

在本节中将要简要介绍本章中涉及的算法，并且通过分析算法的特性来暴露出算法中的一些固有的问题。

3.3.1 Graph Cuts算法

Boykov和Jolly在研究连续泛函问题的全局优化时提出了图像分割方法Interactive graph cuts[2]。在Graph Cuts中需要人工地制定前景和背景的一些点，并且从人工标注的点中得到分割的一些基础，例如前景和背景的位置，颜色信息。

如3.6所示，构造一个s-t网络[4]，网络节点集合V的组成部分由网络上的中间节点（每个节点都对应了图像上的一个像素)和源、汇点S，T组成。其中和S相连的表示前景，和T相连的表示背景。

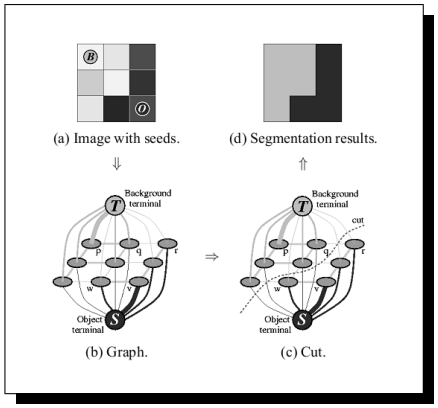

图 3.6: 一个3X3的图像分割例子

将每条边划分为两种类型：

- t-links边集：
  中间节点和源点或者汇点连接的边
- n-links边集：
  相邻像素对p,q之间连接的边。

我们需要对于每条边都赋予一个权值，以下是权值表：

边	权值	类型
$\{p,q\}$	$B_{p,q}$	$\{p,q\} \in N$
	$\lambda \cdot R_p(background)$	$p \in P, p \notin O \cup B$
$\{p,S\}$	K	$p \in O$
	0	$p \in B$
	$\lambda \cdot R_p(front)$	$p \in P, p \notin O \cup B$
$\{p,T\}$	0	$p \in O$
	K	$p \in B$

对于s-t网络的切割可以表示为：

$$A = (A_1, \cdots, A_p, \cdots, A_{|p|}) \qquad A_p \in \{"front", "background"\} \tag{3.1}$$

切割之后所有的像素点分为和源点以及与背景相连的两个部分，它们分别代表的前景和背景。定义划分的能量函数如下：

$$E(A) = \lambda \cdot R(A) + B(A) \tag{3.2}$$

$$R(A) = \sum_{p \in P} R_p(A_p) \tag{3.3}$$

$$B(A) = \sum_{\{p,q\} \in N} B_{p,q} \cdot \delta_{A_p \neq A_q} \tag{3.4}$$

可以证明能量函数的最小值就是对应于s-t网络的最小分割，这里不再多做证明。需要说明的是实际应用的方法是Graph Cuts的改进方法Grabcut[5],由Rother等人于2004年提出。主要改进在于能量函数的改进和EM思想的使用。这里不多赘述。

从上可以看出，Graph Cuts几乎无差别地估计图像上的所有点——而非专注地解决边缘问题，这和我们的问题并不是完全相符的。。正因为这个缺陷，我们在以往的实践中发现一个问题：前景的中心部分往往可以很准确地估计出来，但是在边缘处却或多或少总是有估计不准的地方。经常性地需要重新修改的边缘能够达到一半，而且多数存在于拐角等容易引人注意的地方。

在传统PC上计算资源足够的情况下我们可以选择使用Graph Cuts，牺牲一些计算资源来省却人工标注一些不容易出错的边缘。但是在移动设备上计算资源较为紧张，而操作较为便利，因此不再需要 Graph Cuts。但是需要说明的是在计算资源充足或者过剩的情况下还是可以采用这种算法来节省一些操作开销。

3.3.2 Intelligent Scissors算法

Intelligent Scissors[3]由Mortensen和Barrett于1995年提出，最终被选择作为我们的图像分割算法。

该算法如图3.4所示意的那样，用用户给出边缘曲线的起点和终点之后在图上寻找一条最短的路径作为分割出来的曲线。

该算法实际上是将标注图像当作一个8连通的带权图t，其中图t上的每个顶点都对应于真实图像的一个像素$\mathbf{p}$，而对于和像素 $\mathbf{p}$在8连通区域上相邻的一个点$\mathbf{q}$，它们之间的边的权值为$l(\mathbf{p}, \mathbf{q})$。

当我们确定了图t上每一条边的权值之后，给定两个点A和B，则对应的边缘就是A到B的最短路径。而其中最短路径可以由诸如Dijkstra[6]最短路径搜索算法等算法给出，所以这个问题的核心就是如何定义图像中边的权值：

在 3中考虑到了以下三个方面：图像的边缘，边缘的强度，边缘的走向：

图像特征	表示	影响因素
Laplacian zero crossing	f_Z	图像的边缘
梯度方向	f_D	边缘的方向
梯度大小	f_G	边缘的强度

所以对应于$\mathbf{p}$到$\mathbf{q}$的有向边的权重$I(\mathbf{p},\mathbf{q})$被定义为如下形式：

$$l(\mathbf{p},\mathbf{q}) = \omega_Z \cdot f_Z(\mathbf{q}) + \omega_Z \cdot f_G(\mathbf{q}) + \omega_D \cdot f_D(\mathbf{p},\mathbf{q}) \tag{3.5}$$

其中$\omega_Z, \omega_Z, \omega_D$分别为三种特征的权重。鉴于 3中对于三种图像特征已经给出了明确数学形式的定义和解释，这里只简要地列出它们的形式。 $f_Z(\mathbf{q})$：

$$f_Z(\mathbf{q}) = \begin{cases} 0; & if\ I_L(\mathbf{q}) = 0 \\ 1; & if\ I_L(\mathbf{q}) \neq 0 \end{cases} \tag{3.6}$$

f_G以3阶Sobel算子计算图像在x，y方向的导数I_x,I_y：

$$f_G = \frac{max(G) - G}{max(G)} = 1 - \frac{G}{max(G)} \tag{3.7}$$

f_D表明了梯度方向，其中$D(\mathbf{p})$和$D'(\mathbf{p})$分别表示$\mathbf{p}$点处的梯度方向的单位向量和该单位向量旋转90度后的向量：

$$f_D(\mathbf{p},\mathbf{q}) = \frac{2}{3\pi}\{acos[d_{\mathbf{p}}(\mathbf{p},\mathbf{q})] + acos[d_{\mathbf{p}}(\mathbf{p},\mathbf{q})]\} \tag{3.8}$$

$$d_{\mathbf{p}}(\mathbf{p},\mathbf{q}) = \mathbf{D}'(\mathbf{p}) \cdot \mathbf{L}(\mathbf{p},\mathbf{q}) \tag{3.9}$$

$$d_{\mathbf{q}}(\mathbf{p},\mathbf{q}) = \mathbf{L}(\mathbf{p},\mathbf{q}) \cdot \mathbf{D}'(\mathbf{q}) \tag{3.10}$$

$$\mathbf{L}(\mathbf{p},\mathbf{q}) = \frac{1}{|\mathbf{p}-\mathbf{q}|} \begin{cases} \mathbf{q}-\mathbf{p}; & if\ \mathbf{D}'(\mathbf{p}) \cdot (\mathbf{q}-\mathbf{p}) \geq 0 \\ \mathbf{p}-\mathbf{q}; & if\ \mathbf{D}'(\mathbf{p}) \cdot (\mathbf{q}-\mathbf{p}) < 0 \end{cases} \tag{3.11}$$

通过这一章的算法分析我们可以看出，实际上Intelligent Scissors除去对于图中初始化需要计算每条边的权值之外，在计算每条边的开销的时候是比较低的。这样保证了在标注每条边的时候用户的延迟等待不会太大，一般来说可以达到实时的相应。

但是另一方面我们可以看出Intelligent Scissors算法寻找的仅仅是较为明显的边缘。正如之前在3.2中提到的一样，当我们需要标注的前景物体上有较多明显的条纹(例如斑马这样的动物)，或者背景杂乱(例如背景是一副轮廓明显的壁画)的时候就较为容易出错。所以在标注如图3.7这样的情况我们就要通过分别加上α，β和γ 这样额外的控制点来达到从这三条边缘中选择一条的目的。

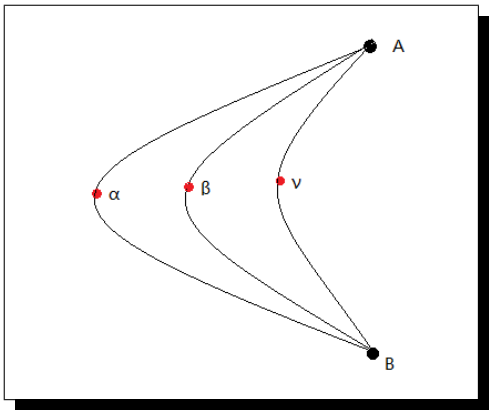

图 3.7: 多条曲线过两点的情况

第四章　标注工具的软件模型设计

在这一章中我将结合软件设计中的UML图来说明整体软件的设计，从软件工程的角度说明软件的设计，构建部分、由于篇幅所限，我将仅仅提及在本软件设计中较为重要的几节，也即：需求分析，用况图文档。同时通过软件的设计分析，我将简明概要地分析本软件的结构，以此来将整个软件的各个模块清晰地介绍出来。

4.1　需求分析及构建图文档

本软件的整体系统主要由以下几个子部分构成：

1. 文件浏览部分：用于浏览平台文件系统上的文件和选择图片。这是整个系统的入口点。
2. 用户标注部分：用于单帧图像的前景分割。这是用户交互最为复杂也最为频繁的地方。
3. 立体图生成部分：人际交互最少而计算量需要最大的部分，将标注结果生成深度图并且完成深度图，视差图，双目立体图。

图4.1是对于子系统的划分，由图4.1可以看出，我们与上文一致地将整个系统划分为了三个子系统。而每个子系统之间最多有两个接口关联。这样在之后使用Android编写代码的时候可以以此为依据划分不同的子Acitivity。需要说明的是由于参与者只有标注用户一个人，之后的分析默认为只有一个参与者。以下给出子系统的需求分析：

4.1.1　文件浏览子系统

文件浏览子系统是三个子系统中较为简单的一个部分，它的功能需求如下：

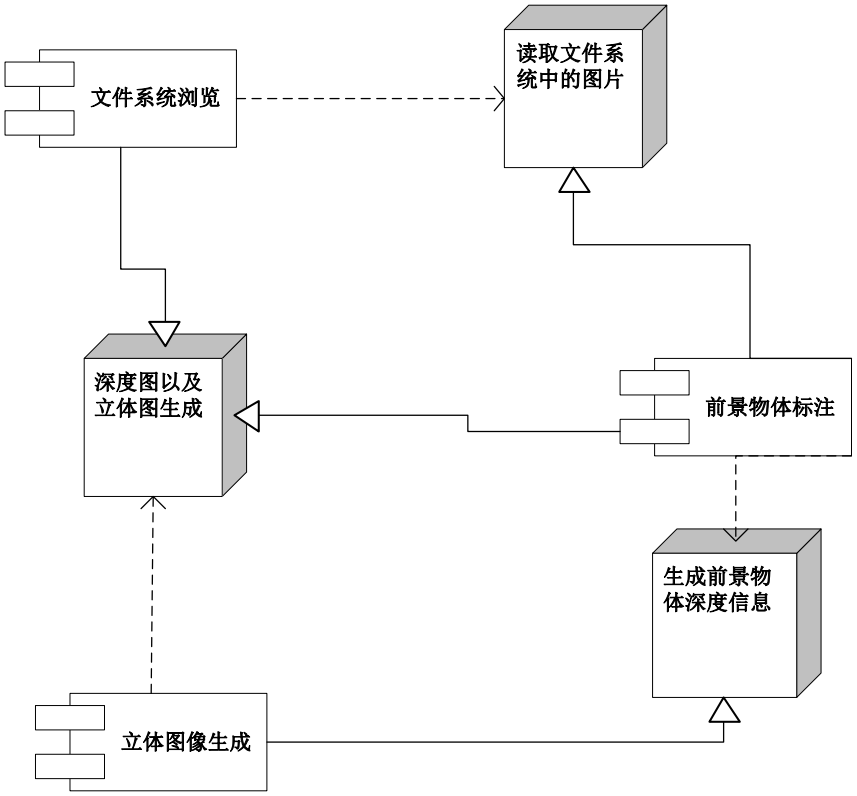

图 4.1: 子系统划分构建图

- 浏览SD卡文件系统上的文件，需要能够遍历所有的具有用户权限的文件夹。由于我们标注的文件一概是图像文件，为了便于用户的选择和浏览，不应该列出不必要的文件如txt文件等。另一方面处于对系统安全性(这里指的是对于整个android系统的安全性) 考虑，不应该让用户拥有浏览系统所在的文件夹的权限，并且实现返回根目录的功能。
- 当用户需要浏览一个文件夹的时候向用户呈现一个图片浏览器，并且让用户能够自由浏览图片。
- 当用户点击图片浏览器上的图片的时候需要将所选择的图片传递给图片标注程序。

据此分析的用况包括：更改当前文件夹，浏览图片，传递图片给用户标注部分。

4.1.2 用户标注子系统

用户标注子系统是三个子系统中交互最复杂的部分，它的功能需求如下：

- 分出三个状态，分别为图片调整，前景物体分割，前景物体调整
- 在图片调整状态的时候允许用户移动图片，并且允许用户调整图片的大小。与此同时原有的图像标记也必须要同步。
- 在前景物体分割状态下，当用户点击屏幕的时候需要能够将屏幕上的坐标转换成实际的图片上的逻辑坐标。这个转换在图片放大和调整了位置之后依旧要有效。
- 用户点击了当前的两点之后，需要生成切割后的边缘，并且将边缘显示在屏幕上。
- 对于一个正在分割中的物体，仅仅开放边缘的两个端点供点击，别的部分则显示为灰色不能够点击，对于用户点击的端点在别的物体内的情况，不能响应以避免物体重叠。
- 当用户将边缘的两个端点相连时要跳出警告，当用户确认确实要封闭曲线之后完成一个前景物体的标注。而此时根据前景物体的深度给物体标上一层有透明度的颜色以便标明已经分割了的前景物体。
- 在前景物体调整的状态下，如果用户点击了一个前景物体弹出一个对话框供用户调整深度，删除当前物体用。

这里分析出来的用况比较多，根据子系统的三个状态可以大致分成三个用况：图片调整，前景物体分割，前景物体调整。

4.1.3 立体图像生成子系统

立体图像生成子系统是三个子系统中计算量最大的一个子系统，但是交互很少，它的功能需求如下：

- 图像自动背景估计
- 将用户标注子系统生成的用户标注与自动背景估计相结合，生成整体的图像深度图

- 在预览时返回一个系统深度图
- 将深度图转换为视差图，最后转换为双目立体图

这里得到的用况包括：背景深度自动估计，最终图像深度图生成，最终立体图生成。

4.2 用况分析及用况图文档

图4.2是系统用况图

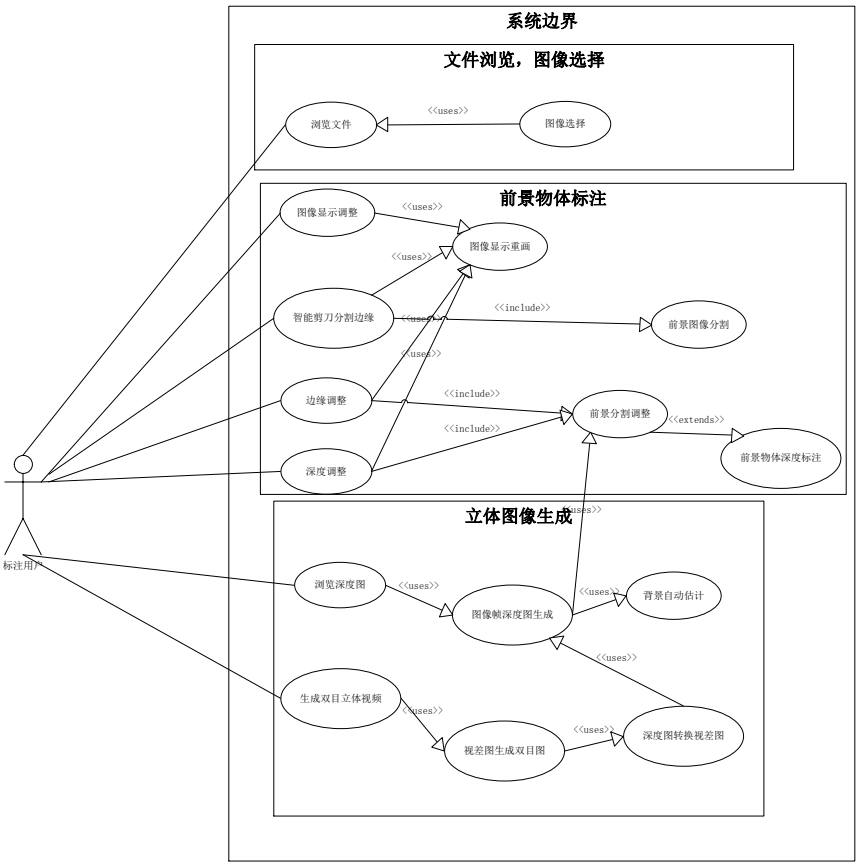

图 4.2: 用况分析图

实际上为了用况图结构的清晰，这里的用况图合并了一些小的用况。我们需要注意到的就是这里子系统的边界比较清晰，并且时间序较为明显。以下是分系

统对于用况的描述：

4.2.1 文件浏览子系统

- 用况图综述：

  文件浏览子系统实现的是从标注工具开启的时刻，用户浏览文件系统并且查找所需图片的过程。在这个过程中需要保证系统文件和隐藏文件的不可见性以确保我们的程序对于操作系统是安全的。

- 参与者描述：

  只有一个参与者即标注用户。不过由于仅仅在这个工具的单独实现中是由用户来选择图片的，而当其加入到整个视频标注系统的时候单人的标注目标图像则是来自于主服务器的分配，所以实际上在处理的过程中我们将其当作是一个单独的参与者。

  相关用况：浏览文件，图像选择。

- 用况描述：
  - 用况：浏览文件

    用户选择子文件夹：如果该子文件夹中有文件或者文件夹，则将当前文件夹切换到子文件夹，并且列出子文件夹和文件夹下的图片。否则提示空文件夹。

    用户选择文件以及选择浏览文件夹：切换到一个图片浏览器窗口，显示当前文件夹下所有图片。

    用户选择返回上一级文件夹：将当前文件夹切换为上一级目录，但是在/mnt/sdcard(即SD卡中用户所拥有权限的最顶层目录)时不响应操作，保证用户对系统目录的不可见。

    用户选择退出文件浏览器：退出图片浏览器，并且回到之前的文件夹列表。目录与之前一致。
  - 用况：选择文件

    用户双击图像：将该图像传递给用户标注子系统。

    用户滑动浏览器：切换到下一张或者上一张图片。

4.2.2 前景标注子系统

- 用况图综述：
  前景标注子系统从用户获得了图片之后，将其全屏显示在屏幕上。之后通过前景图像分割得到前景，经过深度调整，轮廓调整等生成完整的前景用户标注。
- 参与者描述：
  这里的参与者在单机和系统集成的两种情况下都是标注用户。相关用况：图像调整显示，智能剪刀分割边缘，边缘调整，深度调整。
- 用况描述：
  - 用况：图像显示调整
    用户点击放大缩小按钮：图像当前中心位置不变，图片整体放大或者缩小。同时调用图像显示重画接口。
    用户调整图片位置：如果当前处于图像调整状态，则更改图片的起始坐标，同时调用图像显示重画接口。
  - 用况：前景物体分割
    [前置条件]:当前状态为前景分割
    用户依次点击两个端点的时候划分出一条由Intelligent Scissors获得的边缘曲线，并且将其增加到当前已有的边缘当中去。同时更新被激活的线段端点。最后调用图像显示重画接口更新屏幕上的内容。
    当用户将现有的两个激活状态的端点相连的时候，提示用户这个操作会导致曲线的闭合。当得到用户的确认的时候为这个标注中的前景加入最后一条边，并且将曲线闭合。之后生成新的前景分割。
  - 用况：边缘调整
    [前置条件]:当前已有分割完成的前景物体并且选定了一个前景物体
    由于前景物体的边缘是由若干条边缘曲线首尾相连组成，而原有断线保留。调整时需要调节对应边缘曲线的端点的位置，并且重新调用调用Intelligent Scissors分割边缘，更新边缘曲线。
  - 用况：深度调整
    [前置条件]:当前已有分割完成的前景物体并且选定了一个前景物体
    用户拖动SeekBar来在0 255范围内调整物体深度，并且根据该深度调整

前景物体的深度，调用重画接口反映在屏幕上。

4.2.3 立体图像生成子系统

- 用况图综述：

  立体图像生成这个子系统较为特殊，它没有参与者。实际上它负责将用户的标注和自动的背景深度估计结合起来并且生成立体双目视频。这个子系统提供的接口主要被之前两个子系统调用。而且在之后集成到我们的整体标注系统后这部分应该由计算能力充裕的主服务器进行。

- 用况描述：
  - 用况：浏览深度图

    [前置条件]：当前已经初始化完成

    一般这个用况由标注界面调用。而背景的深度图在初始化的时候应该已经生成。在每一次调用这个方法的时候都从前景标注子系统中获得用户的标注情况，之后同当前的深度图结合后生成该帧的深度图最后输出在屏幕上。
  - 用况：生成立体图

    [前置条件]：当前深度图已经完成

    生成立体图在用户保存输出文件的情况下完成，这时调用深度图转换视差图模块将生成的深度图转换成视差图。最后调用立体图像生成模块将视差图转换成双目立体图。

4.3 部分类图文档

以下仅以OOD(问题域)部分类图文档中的前景标注子系统为例说明：

其中图4.3是前景标注子系统部分的类图文档。由于前景标注子系统是较为整个系统中较为复杂的部分，所以我们以此作为例子说明我们的类图的设计，并且期望以此来清晰地传达出软件模型设计时的思路。以下是类图中出现的类的说明：

- 前景物体

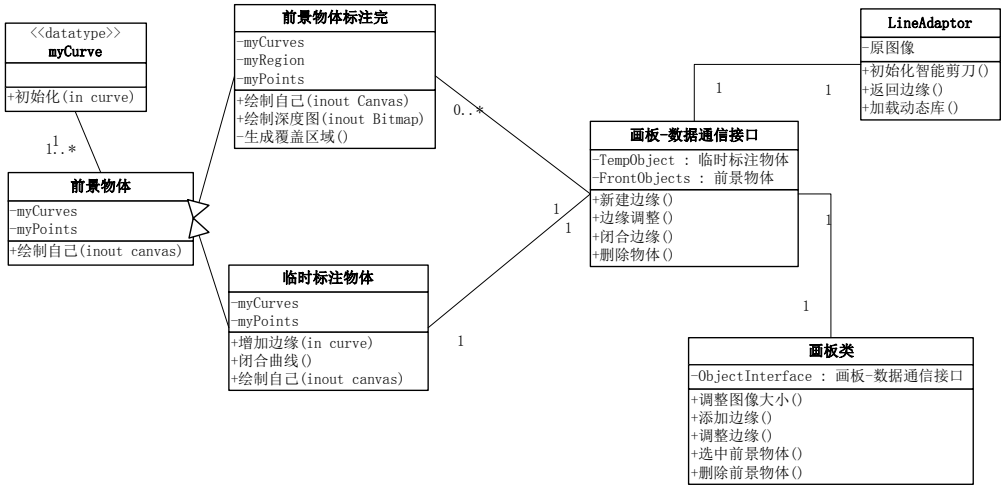

图 4.3: 前景标注部分OOD类图

前景物体这个类定义了一个前景物体所具有的特征，例如边缘曲线以及组成边缘的所有曲线的顶点。并且定义最基本的前景物体将自己绘制在屏幕上的行为。

➢ 前景物体标注完

这个类继承自前景物体，用于定义一个已经分割完毕的前景物体。并且增加了前景物体的一些特征，例如深度和所占据的区域。

➢ 临时标注物体

这个类用于定义一个临时标注的物体，具有代表临时标注物体的一些信息，例如开放供点击扩展的端点。

➢ 画板-数据通信接口

定义了一个从画板的用户操作行为到数据的接口，主要是将用户的抽象行为转换为具体的动作序列。

➢ 画板类

画板类的响应了用户分割的时候主要的操作，并且根据操作的类别来调用各个不同的模块。

➢ LineAdaptor

LineAdaptor是用于实现智能剪刀功能的类。这个类是JAVA虚拟机和本地库

文件的接口。

4.4 软件模型总结

通过本章的分析，我们可以清楚地了解到整个软件的构架。另一方面通过对于软件子系统的划分，我们可以分析出在软件的运行周期中，软件的各个部分对于资源的需求，以及它们互相之间调用的情况。通过这些信息就可以在下一步合理地分配计算资源以及使用合适的方式来实现我们事先定义好的各种算法。

通过在这一章的分析我们可以明显地看到软件各个子系统之间的关系，以及各个子系统之间互相的消息传递。这样我们就可以基于实现平台的特点和用户操作的需求来设计我们软件的界面。

另一方面，通过清晰地梳理出独立的大计算量的模块，让我们对于各个部分对于计算资源的要求层次有了比较清晰的认识。这样就能够针对性地设计一些方法使得计算得到优化。

第五章　标注工具的具体实现

根据上一章的结果，我们得到了整体软件的初步模型。这一章将要在得到的模型的基础上，具体地说明软件实现的细节，并且有重点地介绍一下在编码的过程中使用到的技术。

5.1　软件实现平台

如前所述，我们的工具实现在Android移动平台上。以下是软件实现中的一些具体参数：

操作系统	Google Android®
系统版本	兼容2.1至2.3版本的Android系统
硬件平台	基于ARM体系结构的移动平台
实现语言	JAVA&C

接下来简要地介绍以下我们选择上表作为软件的总体开发环境的理由。

关于平台的选择，Google公司的Android是当前最为普遍的移动平台上的操作系统。另外由于Android可以在各种移动设备诸如手机和平板电脑上稳定的运行，我们在今后选择硬件平台的自由度就会比较大。换句话来说，如果我们选择了一种操作系统，例如Apple 公司的IOS系统，那么很有可能的情况就是最适合我们的硬件设备并不能运行这个操作系统。也就是说，选择Android操作系统能够最大地保障我们在限制成本的情况下选择最合适于我们需求的硬件设备。

关于操作系统的版本，虽然我们只是声明能够兼容2.1至2.3版本的Android系统，但是实际上在最新的Android 4.0.3系统上依旧可以稳定运行。我们选择2.3版本作为我们开发的主要版本是由于市场上最为普遍的Android系列的操作系统就是 Android2.3操作系统。

另外选择过高版本的操作系统会在硬件条件不变的情况下拖慢整体工具的速度，亦或是为了保证在高版本操作系统上依旧能够获得流畅的操作就必须购买更高配置的设备。但是低版本的操作系统则缺失了很多特性，并且不够稳定。所以综合考虑决定以2.3版本作为主要支持的版本。

5.2 软件实现中用到的优化技术

在我们的软件的立体图像生成的模块计算量较大，而Android提供给我们的API为java语言。虽然java跨平台的特性使得在我们对于不同硬件设备的兼容性不需要考虑到很多。但是另一个方面，随着JVM(Java虚拟机)给我们带来的便利性的同时，我们也要为这种便利付出我们的代价，那就是效率问题。

虽然在J2SE1.4.2发布后Java的速度执行速度有很大的提升，但是在数学计算，内存读取等方面和C/C++的差距还是很明显。于此同时我们的大量开销都花费在图像操作上，这却恰好是设计大量数学计算和内存读取的操作。所以我们需要将基于C语言实现的算法移植到Android上。以下我将会重点两个在移植中大量使用的技术。

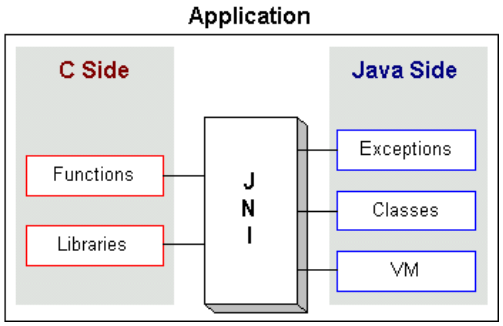

图 5.1: JNI调用的关系

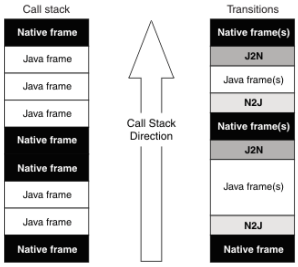

图 5.2: JNI程序调用堆栈

5.2.1 JNI技术

JNI是Java Native Interface的缩写，中文为JAVA本地调用。顾名思义，它是用来调用本地的函数或者方法的接口。

我们知道Java语言都是运行在JVM虚拟机上的，而JVM虚拟机则是运行在本地接口上的。由于JVM的特性，它在大量数学计算和内存读取上相较 C/C++等编译执行的语言来的要慢。但是我们恰恰在进行图像处理的时候需要进行大量的数学计算，这就和我们的要求相违背。所以我们希望能够找到一个折衷的办法，就是如何使Java语言能够在处理这些问题的时候使用C/C++编写的库。

按照Sun公司给出的说明 7，JNI的作用不仅仅是调用C/C++的动态链接库(DLL)，而是"enables the integration of code written in the Java programming language with code written in other languages"。 [1]

如图5.1，我们可以看到，在JAVA端，JVM虚拟机执行字节码文件。通过JNI接口访问C端的函数或者库。实际上，这个库是运行在操作系统本地上(所以称作NATIVE方法)。也就是说它和JAVA代码分别运行在虚拟机和本地操作系统上，所以要明确这个概念：JNI并不是提供在JVM上执行C语言程序的方法，而是提供了一个穿越JVM的接口，可以近似地将它理解成一个系统调用。

根据5.2我们可以清楚地看到java页面和native页面交替执行和通信的过程。不过这种切换总是有开销的[8]，我们尽量降低子系统之间的交互其中有一个原因就是避免过多的转换开销。

5.2.2 NDK技术

我们知道，Android是一个以Linux为基础的操作系统。但是实际上Android截止到4.0.3版本的系统发布为止，仍然只有公布JAVA语言的API，也就是说如果我们需要编写Android操作系统上的第三方程序一般来说就需要使用JAVA语言(实际上借用SL4A等技术可以使用诸如Python等脚本语言，但是这仅仅是一个取巧的方法，并不适合大量使用)。

另一方面，Google从一开始就允许了使用C/C++开发。从Android SDK首次发布的时候Google就已经宣布Dalvik虚拟机是支持 JNI接口的。也就是说，JNI这种技术即使不需要NDK技术的加入也是一直都可以实现的(实际上我们如果翻阅SDK源码，可以发现其中使用了大量Native方法)。但是另一方面，我们的本地动态链接库如何与应用程序打包在一起成为apk并发布？

[1] 感谢Sun公司免费将这本书提供给我们，使得我们不用应付类似在学习C#或者别的技术时面对的问题：花大价钱去买下它或者下载版权有争议的电子书，虽然我们的目的是为了推广这个技术

所以我们使用了NDK技术，使得在编译的时候可以借用NDK进行Android系统上的交叉编译，生成动态链接库.so。最后借用NDK将程序和.so文件打包在一起发布。这使得我们在进行开发的时候省去许多不必要的配置。

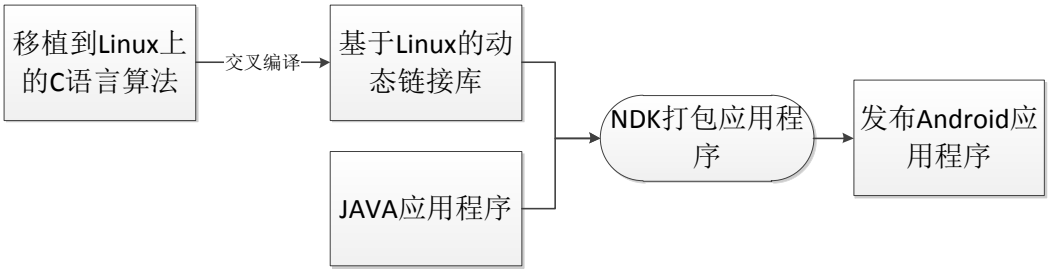

图 5.3: 利用NDK交叉编译C语言代码以及打包的流程

如图5.3所示，我们将在linux上实现的C语言算法交叉编译成适应于Android系统的.so文件，然后将它和Java应用程序一起打包，最后生成可以发布的APK。

5.3 部分程序界面介绍

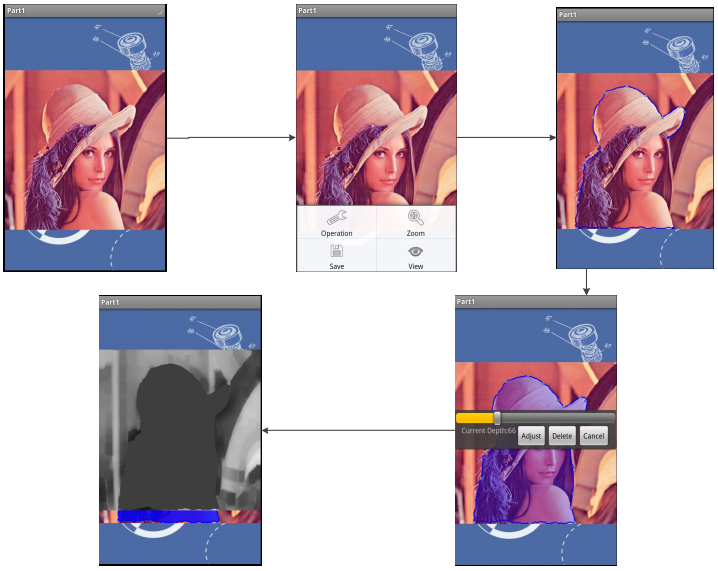

图 5.4: 程序界面流程

如图5.4所示，整个的程序的流程包括初始界面，选择菜单，标注，前景物体调整以及深度图预览几个部分，和我们在图3.1中看到的一样。以下是各个部分的展示。

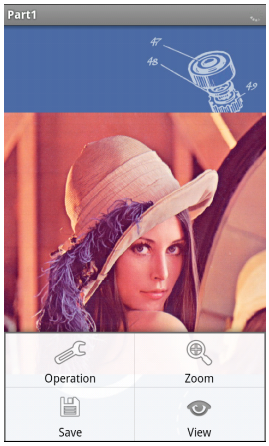

图 5.5: 程序主界面

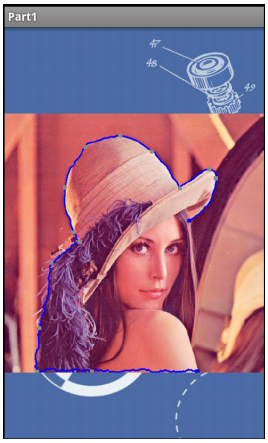

图 5.6: 前景分割

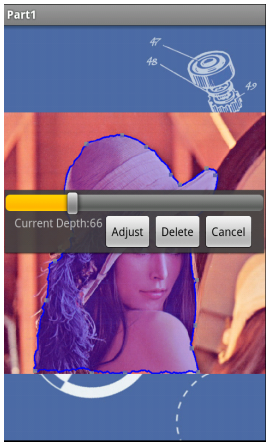

图 5.7: 前景调整

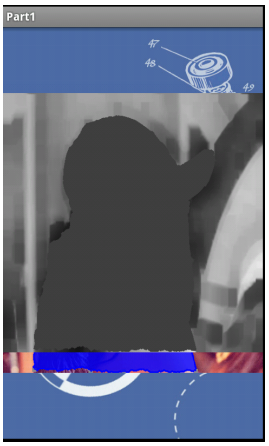

图 5.8: 深度图预览

总结

作为毕业设计的内容完成了基于触摸屏的标注模型的设计以及在Android初步实现了需要的各种功能。将Intelligent Scissors和背景深度估计移植到了Android平台上，并且实现了和JAVA应用程序的衔接。在Android平台上实现了单帧图像的视频标注工具。其中集成了前景分割模块，背景自动深度估计模块，深度图合成模块，深度图转换为视差图以及视差图转换为立体图像模块。

针对当前阶段已有工作的效果进行评估，我们认为在Android移动设备或者其他基于触摸屏的设备上进一步完善这个工具并且加入新的功能是十分可行的。但是离真正的推向实用化还有一定的距离，系统的架构还需要我们进一步的优化，并且优化对于一些资源的使用，避免计算资源和存储资源的浪费。如果有可能的话可以考虑增加使用C语言实现的比重，这样使得我们的系统速度更快。

下一步的工作，一是要对于占用过多资源的部分做出进一步的优化。其次是完善标注功能，使得使用更为简易和方便。最后需要加入和服务器的通讯，将部分繁重的计算量转移到服务器端上。

参考文献

[1] V. Nedovic, A. W. M. Smeulders, A. Redert, J. M. Geusebroek. Stages As Models of Scene Geometry[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010, **32**(9):1673–1687

[2] Y.Y. Boykov, M.P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images[C]. Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. IEEE, 2001, vol. 1, 105–112

[3] Eric N. Mortensen, William A. Barrett. Intelligent scissors for image composition[C]. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. SIGGRAPH '95, New York, NY, USA: ACM, 1995, 191–198. URL http://doi.acm.org/10.1145/218380.218442

[4] T. H. Cormen. Introduction to algorithms[M], third . The MIT press, 2009

[5] C. Rother, V. Kolmogorov, A. Blake. GrabCut -Interactive Foreground Extraction using Iterated Graph Cuts[C]. Proceedings of the 31nd annual conference on Computer graphics and interactive techniques. SIGGRAPH '04, ACM, 2004, 309–314

[6] Wikipedia. Dijkstra's algorithm[Z], May 2012. URL http://en.wikipedia.org/wiki/Dijkstra's_algorithm

[7] S. Liang. The Java native interface: programmer's guide and specification[M]. Addison-Wesley Professional, 1999

[8] Andy Thomas-Cramer. What makes JNI calls slow[Z], Oct 2011. URL http://stackoverflow.com/questions/7699020/what-makes-jni-calls-slow

致谢

首先诚挚地感谢我的指导老师王亦洲研究员。王老师在整个毕业设计的完成过程中，常常给与我关键性的指导，令我能够把握正确的研究方向，同时对于相关领域的先进工作保持清晰的了解。王老师对于学术问题一丝不苟的态度和严谨的作风，以及在细节问题上的严格要求让我通过这次毕业设计对于一个科学研究人员应该有的科学素养有了更明白的认识。

感谢我的师兄张哲斌博士，在他的协助下我确定了明确的设计方向。感谢周辰师兄，他是我们标注工具的PC版的作者，感谢他在自身事务繁忙的情况下仍然不厌其烦地为我答疑解惑。感谢梁艳慧师姐和辛博师兄，在实验室的工作中我得到了很多来自于他们的帮助。感谢实验室和大学生活中遇到的老师和同学们，从他们身上我学到了很多。

感谢我以前的室友韦屹同学，他在我最后完成编码工作的时候给予了我很多美工上的帮助。感谢StackOverflow和CSDN以及别的技术论坛上的网友们，感谢他们多次热情地帮助我解决了编码工作上的问题。感谢《Hello，Android》的作者Ed Burnette先生，他写的这本书是我学习Android编程的主要途径，没有这本书我可能不会入门地如此顺利。

感谢张翔、丁宁、范笑妍、潘虹、耿士藜、欧阳辰、李鹏飞、马政武以及其他许多的我的朋友，感谢他们不断给予我生活上的关心和帮助。感谢他们给我带来的生活的温暖。

最后感谢我的父亲和母亲，感谢他们一直以来对我的支持，感谢他们对于我各种错误和胡闹的容忍，感谢他们每每在我偏离正确的方向的时候给与我的纠正，感谢他们从小到大给我灌输的正确的价值观和人生观，让我能够无论是在学术还是生活上都能以努力向着成为一个正直，诚实的人为目标前进。感谢他们一直以来给予我的关怀和亲情。