北京大学

本科生毕业论文

题目：（中文）面向三维重建的图像标注工具开发

（英文）Development of an Image Annotation Tool for 3D Reconstruction

姓　　名：周辰

学　　号：00748341

院　　系：信息科学技术学院

专　　业：计算机

指导教师：王亦洲

二〇一一年六月十一日

北京大学本科毕业论文导师评阅表

学生姓名	周辰	学生学号	00748341	论文成绩	
学院（系）	信息科学技术学院			学生所在专业	计算机科学技术
导师姓名	王亦洲	导师单位/所在研究所	数字媒体研究所	导师职称	研究员
论文题目（中、英文）	面向三维重建的图像标注工具开发 Development of an Image Annotation Tool for 3D Reconstruction				

导师评语

本论文开题题目是《面向三维重建的图像标注工具开发》，其核心工作是通过简单的手工交互操作，实现对图像的场景结构和深度信息标注。 在完成毕业设计过程中，论文作者根据实验室项目需求，把该工具与“交互式 2D 视频到立体视频转换系统”的单帧（关键帧）场景深度标注模块相结合，成为系统的关键模块之一，并以此功能模块的界面为平台，实现了对系统各环节的串联、集成。选题工作量较大。论文结构完整，写作清楚，达到了学士学位论文要求。

导师签名：

2011 年　6 月　1 日

摘要

随着 3D 电影电视产业的快速发展，3D 片源缺乏的问题日渐突出。在此背景下，研究将传统 2D 视频转化为 3D 视频的技术成为一个既有实用意义，又有研究价值的课题。由于目前全自动 2D 到 3D 视频转换技术存在着较大的局限性，我们希望开发出一个交互式 2D 到 3D 视频转换系统，能以半自动方式将 2D 视频转化为 3D 视频，并在效果良好的前提下尽量提高自动化程度。

在毕业设计的过程中，我以“面向三维重建的图像标注工具”为基础，完成了对视频中单帧进行相关信息标注，并将其转换为立体图像的模块。

关键词：3DTV，2D 到 3D 转换，深度估计，视差转换，前/背景分割

Abstract

The problem of lacking sufficient stereoscopic video resources has emerged with the recent booming of 3DTV industry. Under this circumstance, the technique of converting a traditional 2D video to a stereoscopic one has become a topic with significance both for research and application. As an automatic conversion method is not able to achieve good results on a wide range of video, we're trying to develop an interactive system for stereoscopic video conversion which converts 2D videos to 3D with good visual effect and relatively modest user interaction.

I accomplished the single frame labeling and conversion module in the system as my graduation project.

Keywords: 3DTV, 2D-to-3D conversion, depth estimation, depth to disparity, foreground/background segmentation

目录

第 1 章 项目介绍

1.1 3DTV 发展状况介绍与项目背景

3D 电影，或者立体电影（stereoscopic film），是一种能让观众感知景深，从而营造出强烈立体感的电影技术。虽然 3D 电影的历史至少可追溯到 20 世纪 50 年代（*Bawana Devil (1952)，* 被认为是美国第一部彩色 3D 电影），而且也曾有过短暂的辉煌期，但由于拍摄硬件昂贵，流程复杂和技术上的种种不足，它在电影业界长期处于边缘地位。直到 2000 年以后，3D 电影又逐渐开始流行，而电影《阿凡达》2009 年在全球的火热上映与之后 3D 电影数量的井喷式增长，可以说开启了 3D 电影逐步走向电影产业主流的新纪元。

随着技术的进步与消费者对立体视频接受度的增加，3DTV 也获得了业界重视并正在迅速发展。索尼、三星、松下等国内外各大电视厂商都争相推出各自的 3D 电视；3D 电视频道的开通也成为各大电视台争取下一代电视服务制高点的热点。在海外，大量 3D 电视频道正在筹备或者开播，如英国的 sky 3D；由 ESPN 开办，直播体育赛事的 ESPN 3D；美国由 DirecTV 开办的 Cinema 3D、3net 等等。而在我国，各大中央和地方电视台也计划在近期（2011~2012 年内）开通 3D 频道。

3D 产业快速发展带来的一个问题就是片源缺乏。目前 3D 节目的制作主要依赖立体摄像机拍摄或者三维动画的编辑制作，而原有的传统 2D 电视节目却不能直接的在 3D 频道实现立体播放。在英国，天空电视台（SKY），BBC 在 2010 年也着手开通一个或多个 3D 频道，但某些频道却因为每天非重复播放的 3D 节目达不到至少 6 个小时的要求，而不得不推迟正式开通频道的时间。由此可见，如果能够很好的利用原有的 2D 电视节目，即选择适合的 2D 节目，利用一定的技术手段，转换其成为适合 3D 频道播放的立体视频，那么，上面英国两家电视台遇到的问题便可迎刃而解。所以，2D 到 3D 视频转换技术是 3D 电视产业发展的必要环节。

目前国际上已经有一些公司陆续推出自己的转换技术，可以分为全自动和半自动两类。在全自动转换技术方面，Tridef DDD 推出了自己的软件产品，并与三星合作生产 2D 到 3D 视频转换芯片，集成到了三星的三维电视中；JVC 也推出了一款硬件解决方案 IF-2D3D1。在手动或半自动方面，IMAX 用其手动转换软件，把最新的《哈利波特 7》中的约 20 分的片段转为立体视频。但目前两种手段都还有其缺陷。全自动转换一般为了达到实时效果，只用简单的三维场景先验知识，并简化了视频帧间关系的约束，导致算法估计结果与真实感受相差甚远；半自动或交互式转换效果较好，但也存在效率不高，以及在人工标注时由于不同的人做了有差异的标注动作导致转换结果不理想等问题。

我们的“2D 到 3D 视频的交互式转换系统”项目正是针对上述需求和已有手段的缺陷，希望开发出一个高效、转换效果良好的半自动 2D 到 3D 视频转换系统，能够应用到 3D 产业的实际生产流程中。

1.2 本文所做的工作

在本项目中，我完成的主要是系统中单帧交互式标注、转换模块和系统各部分的集成工作。

需要说明的是，本论文开题题目是《面向三维重建的图像标注工具开发》，其核心工作是通过简单的手工交互操作，实现对图像的空间三维结构标注，进而能够实现从 2D 图像到其对应场景的三维重建。 在毕业设计过程中，由于实验室项目需求，毕业设计目标做了一定调整。目前所开发的标注工具通过标注场景内容（前景物体和背景组成部件）的相对深度，生成立体图像，因此是对图像场景的 2.5D 重建。这与开题题目中的“三维重建”有所差异，是在具体项目需求和应用背景下做的适度调整。

本文后面部分组成如下：第二章将简要介绍立体视频的原理，包括人眼产生立体视觉的机理和目前生成 3D 视频所用的主要技术。第三章将概述系统流程，并详细介绍单帧标注转换模块的分析及设计。第四章是对系统中一些相关算法的总结。第五章展示了一些转换结果。第六章是对本文的总结。

第 2 章 3D 视频原理介绍

2.1 人体立体视觉原理

人类获得立体视觉的线索很多。只通过单目就能获得的线索称为单目线索（monocular cues）,如物体间遮挡透视关系、视线移动时远近物体移动关系（motion parallax）等; 需要双目才能获得的线索称为双目线索（binocular cues），即由于人双眼位置的差异导致同一物体落在双眼的像略有差异（观看视差）而产生的立体感。

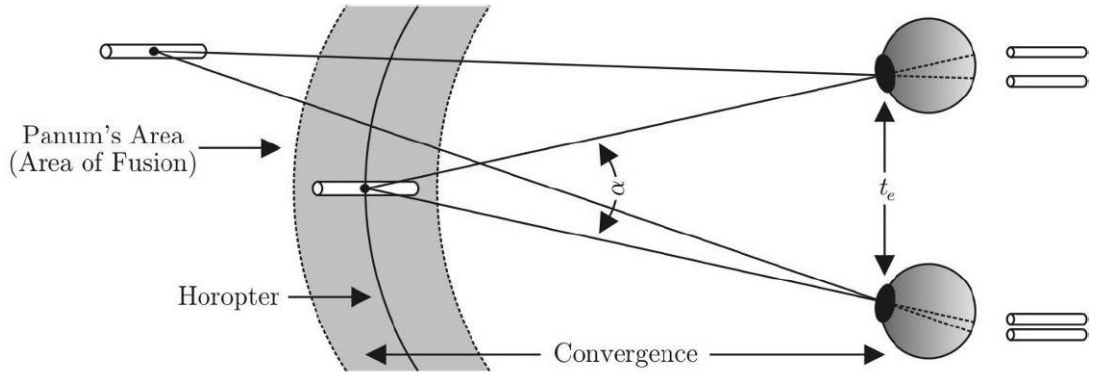


图 2.1-1 人眼立体视觉（正上方俯视图）

由双目线索产生立体视觉的详细描述可见上图[1]。当人试图看清某物体时，会转动眼球，使视线汇聚（converge）在该物体上（如图中灰色区域中的小棍），此时该物体在左、右眼所成的像中的位置相同，视差（disparity）为 0。在图中被称为 Horopter（双眼单视界）的弧线上（真实世界中为一个区域），点的成像同样具有 0 视差；在其他位置上，点因为在左、右两眼所成的像中的位置不同而具有非 0 的视差，离图中 Horopter 越远的点，产生的视差越大[2]。

物体产生的视差在一定范围内时，人的视觉系统能够将双眼所成的略带差异的像融合到一起，并感知物体深度信息，但视差超过一定范围后，就会产生复视（diplopia）现象。不产生复视现象的点的范围被称为 *Panum's Fusional Area*，即图 2.1-1 中的灰色区域。

2.2 立体显示的基本技术

尽管在具体实现上有差异，但目前的立体显示技术基本上都是利用双目线索，通过让观众双眼同时看到略有差异的两幅图像来实现立体效果。图像中对应点的距离称为屏幕视差（注意与上文“观看视差”（指物体在双眼视网膜上所称的像的距离）含义有区别。下文提到的视差均指屏幕视差）。

将视差记为在屏幕上右眼观察到的点与左眼观察到对应点的距离，用 P 表示。双眼看到的图像情况如下图：

a) 右眼观察到的点在右，左眼观察到的点在左，汇聚点形成于屏幕后面，即屏幕空间。此时 P>0。若 P=人眼间距，则在无穷远处形成汇聚点。因此要求 P<人眼间距。
b) 双眼观察到的成像点是同一个点，汇聚点在屏幕上，此时 P=0。
c) 右眼观察到的点在左，左眼观察到的在右，此时汇聚点形成于屏幕前面。P<0。

所以原则上，对于不同景深的物体，通过赋给其不同的视差即可实现立体效果。视差为正/负值时分别可实现入屏和出屏。

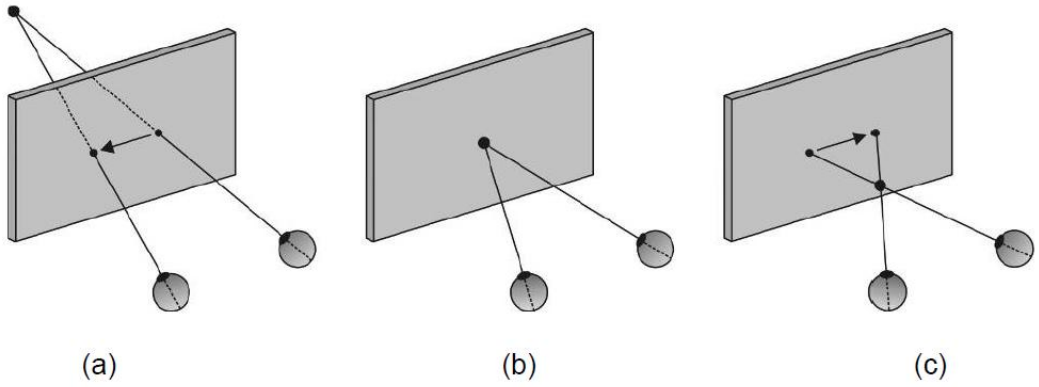


图 2.2-1 立体显示示意图

对于如何让双眼同时看到不同图像的问题，目前主要有如下一些技术：

色分法（Anaglyph）：主要原理是提取左眼图像的红色通道和右眼图像的蓝绿通道合成一副图像，观看时佩戴红青眼镜，使得左眼只看到红色图像（因为蓝绿图像通过红色镜片滤光而成黑色），右眼只看到青色（蓝绿）图像，由此产生立体效果。这种方法的优点是实现简单，设备要求低；缺点是实际上损失了部分颜色信息，容易造成观看者疲劳。

快门式（Shutter Method）：显示器以高频率交替显示左右两路图像，分时眼镜同步接收（调整透光状态），形成立体效果。NVidia 的 3D Vision 产品目前采用这种方式。这种方法的优点是立体效果较好，但缺点在于一是目前仍然有一定闪烁问题（因为以 120Hz 交替显示两幅图像的显示器，每只眼睛看到的刷新率只有 60Hz），看久了容易出现疲劳，二是画面会有明显变暗，因为每只眼睛实际只得到一半的光。

光分法（Polarization Method）：在放映时，左右两幅图像分别透过偏振片形成垂直的偏振光，观看者戴上偏振眼镜，使得每只眼睛只看到相应的偏振光图像。这种方法的立体效果好，但设备复杂昂贵，一般只用在电影院中。

裸眼 3D（Autostereoscopy/ glasses-free 3D）：指一些无需佩戴 3D 眼镜就能看到 3D 效果的技术。一种比较流行的技术是 parallax barrier，通过在图像显示设备前放置光栅，使得左右眼分别只能看到图像的一部分像素。这种方式的优点是无需佩戴眼镜，但缺点是观看者必须要在特定角度才能得到较好的 3D 效果，另外由于遮挡，每只眼睛实际上只能看到一半的像素。该技术示意图可见图 2.2-2。

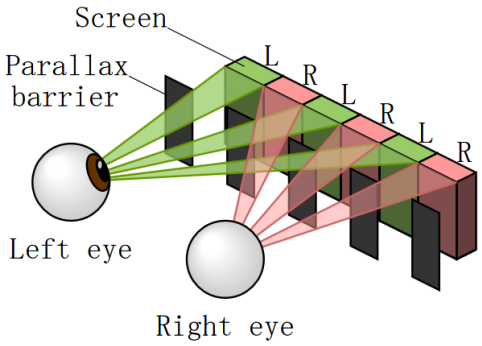


图 2.2-2 Parallax Barrier 示意图

第 3 章 系统设计

3.1 系统总述

3.1.1 系统流程

我们的系统需要将 2D 视频转为立体视频。由第二章介绍的 3D 显示原理可知，其基本流程是从视频的 2D 帧中获得景物的深度信息，并由此计算出每个点的视差，得到视差之后就可以选择任何一种技术显示 3D 化的帧。虽然理论上可以通过用户逐帧纯手工标注得到所需的深度信息，但我们希望在保证效果的前提下尽量提高自动化程度。所以目前系统总体流程设计如下：

1. 针对待转换的 2D 视频，首先对其进行镜头分割和聚类，以便将视频中相似的帧聚合在一起。

2. 针对聚类中的一些关键帧，让用户进行标注。利用标注信息，可得到该帧的视差图并生成立体图。

3. 将关键帧的标注信息传递到同一个聚类中的其他帧上，生成立体图。通过标注信息的帧间传递可以减小用户的工作量。

4. 将得到的各帧立体图压缩成最终的 3D 视频。

3.1.2 模块划分

图 3.1-1 给出了系统的模块划分。系统可分为五个模块：音视频分离（合成）与解码（编码），视频镜头分割及聚类，场景深度估计及视差转换，双目视频生成以及可 2D->3D 转换序列选择模块。

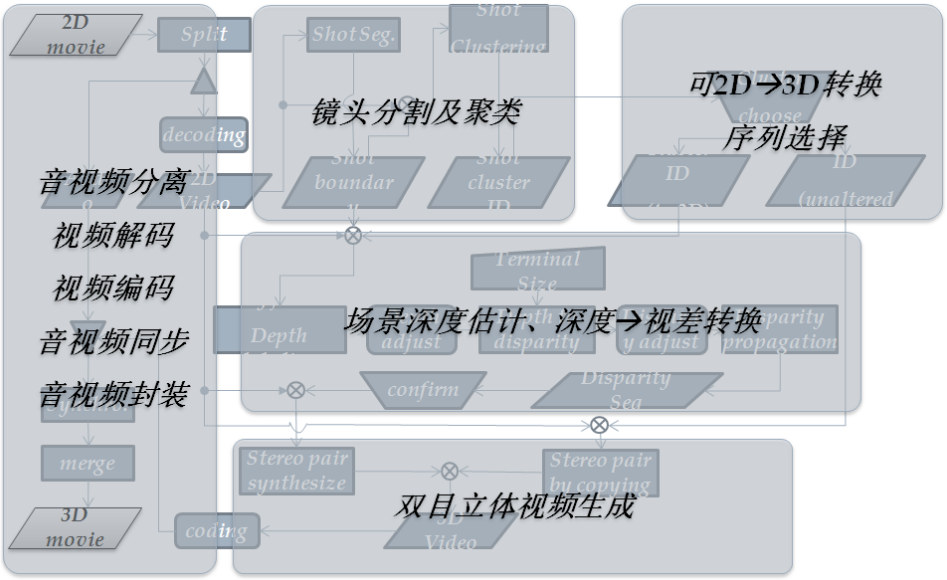


图 3.1-1 系统模块图

这些模块中，“音视频分离（合成）与解码（编码）”和“双目视频生成”模块主要完成数据的预处理和后处理工作，即读入视频进行解码，分离音视频流和最后将转换好的 3D 视频与音频同步，输出有声 3D 视频；“可 2D->3D 转换序列选择”是一个附加模块，用于选出视频中不适合转 3D 的镜头（如单纯的字幕、立体感很弱的场景等），这类镜头无需估计其深度信息，直接将视差设为 0 生成立体图输出即可。

“视频镜头分割及聚类”和“场景深度估计及视差转换”是系统的两个较为核心的模块。视频镜头分割模块将相似的帧聚合在一起，一是让用户在少量关键帧上的标注可以传递到非关键帧上，可以极大减少用户的工作量；二是方便筛选视频中适合转 3D 与不适合转 3D 的镜头。场景深度估计及视差转换则覆盖了用户对关键帧进行标注的过程。

3.1.3 系统界面总览

系统主界面有两个，如图 3.1-2。左边的界面展示镜头分割及聚类的结果，右边主要用于用户单帧标注操作。

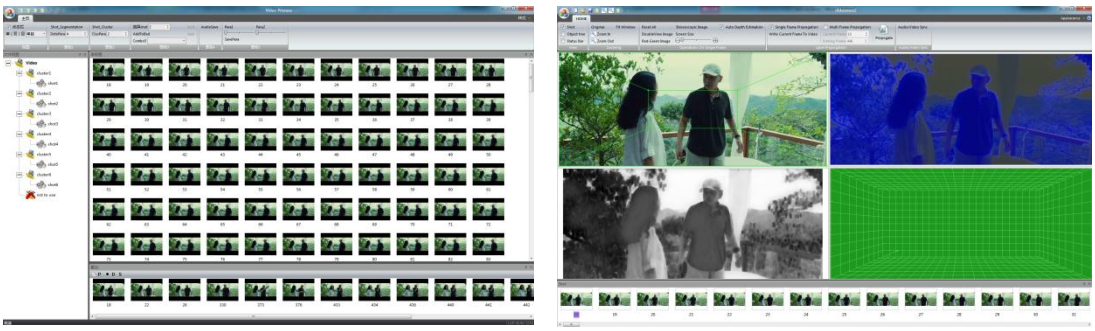


图 3.1-2 系统界面

目前两个界面在同一台显示器上运行，将来可能将系统运行在三显示器的环境中，这样用户可以方便地查看、选择需标注的镜头，进行标注，以及查看转换结果。

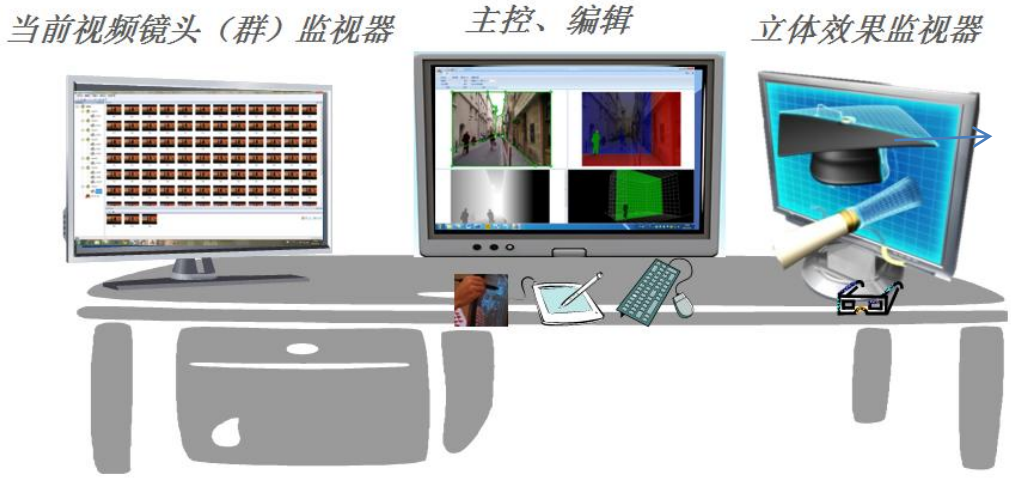


图 3.1-3 三显示器效果图

3.2 场景深度估计及视差转换模块的设计和实现

在整个系统中，我完成的主要是场景深度估计及视差转换模块。以下将详细介绍此模块的设计与实现。

该模块基本需求上文已提过，即用户交互式标注图像深度信息，并利用此信息得到视差图及输出立体图像。考虑程序收集用户标注信息，产生立体图像的工作流程，该模块可再细分为以下模块。

1. 背景标注、前景标注

我们把深度标注的过程分为前景标注和背景标注两部分。图像的前景千变万化，但背景部分往往呈现出相对固定的模式（见 3.2.1 节）。因此对于背景，我们可以以合适的模型对其建模，并以此估计场景深度信息；对于前景，可先将其分割出来，再手工指定其深度，或者根据前景在背景中的位置自动估计其深度。

2．深度图生成

有了前、背景的深度信息之后，程序可用此生成反映整个场景相对深度情况的深度图。

3．屏幕位置标注

深度图可以反映场景深度关系，但为了得到视差图，必须知道场景与屏幕的位置关系。因此需要指定屏幕位置。

4．视差图生成

利用前、背景和屏幕位置标注的结果，生成视差图。

5．立体图像生成

利用视差图和原图像，生成最终立体图像。可以是红青图、双路图像等。

各模块流程关系如下图：

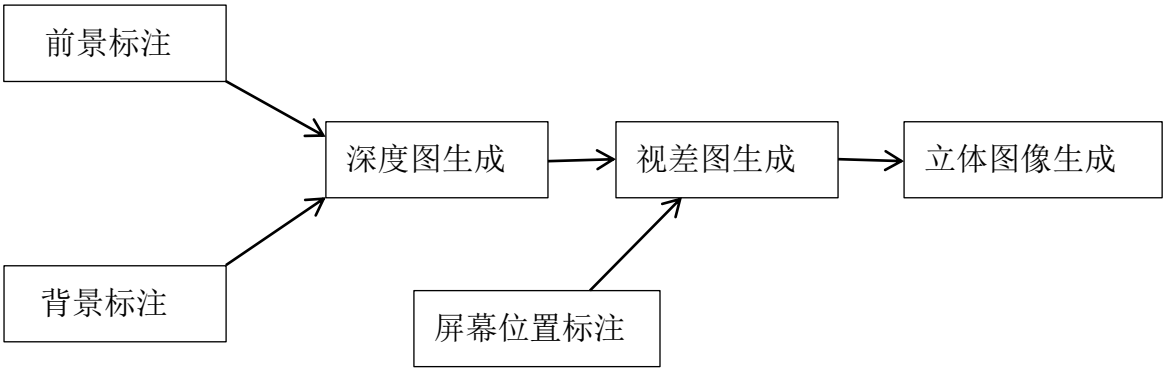


图 3.2-1 各模块流程关系图

3.2.1 节到 3.2.6 节将依次介绍些模块的实现情况。

3.2.1 背景标注

背景标注方面，我们用 Stage Model 给背景建模，并以此估计深度信息。

3.2.1.1 基于 Stage Model 的背景建模

生活中许多常见的场景，都可以用几个平面的组合来粗略描述其几何结构。如室外风景，常常可以用地面-天空两个面或者地面-远山-天空三个面大致表示其背景；室内场景，常常可以划分为地面、墙壁及天花板；街道场景，可以划分为地面、建筑物外壁、天空等等。由于真实世界中场景结构的限制（如建筑物总垂直于地面、平行线交汇于消失点等），这些面的组合一般是比较固定的，可以被划分为有限的几种类别。（图 3.2-2）

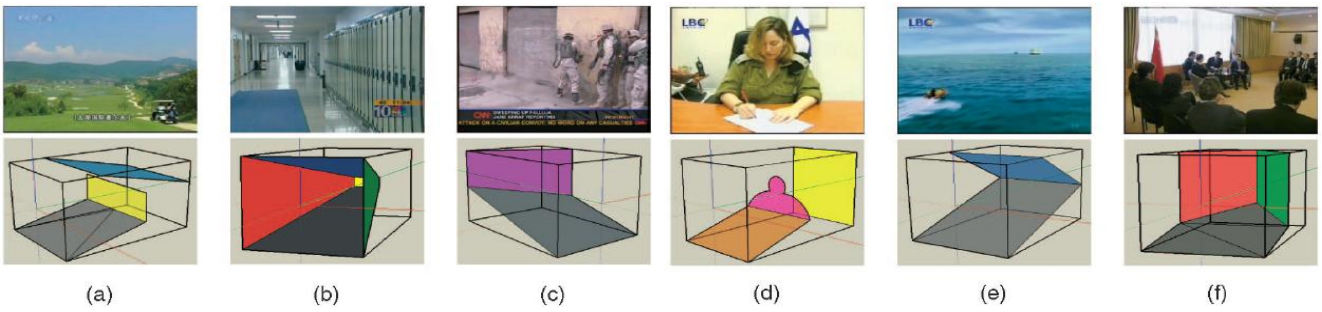


图 3.2-2 stage model 图示

Nedovic[3]等人根据上述观察，将场景的几何结构划分成了 15 类（图 3.2-3），

称为 **stage model**。每种类别称为一种 **stage**。注意如果把类别的切换看成相机在空间中移动并采样（拍摄）的过程，并定义相机的三种基本操作 pan(水平摇动)，tilt（竖直摇动）和 zoom in/out（变焦，放缩），那么除去特殊的 no depth stage 之外：1) 由任何一个 stage，通过上述三种基本操作，都会变换到已知的 stage 之一；2)任何一个 stage，都可以由另外的 stage 通过上述三种操作得到。（图 3.2-4）这也从一定程度上说明这种模型是比较完备的。

如图 3.2-3 中 sky+background+ground 类型，当相机 zoom out 到一定程度，background 面消失，就转换成了 sky+ground 类型；当相机向下转动，使 sky 这个面面消失，就成了第二列的 background+ground 类型；当相机再向下转动使 sky 和 background 面都消失时，就成了 ground 类型。

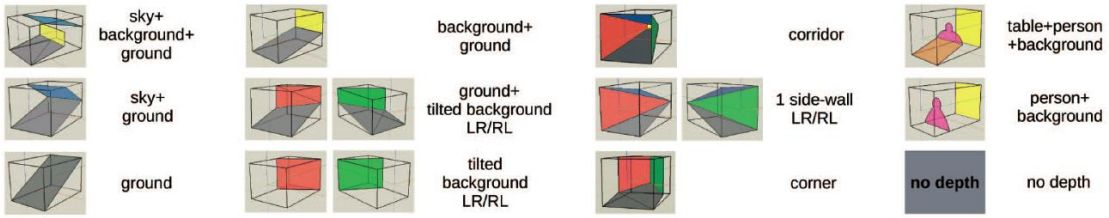


图 3.2-3 stage model: 共 15 类

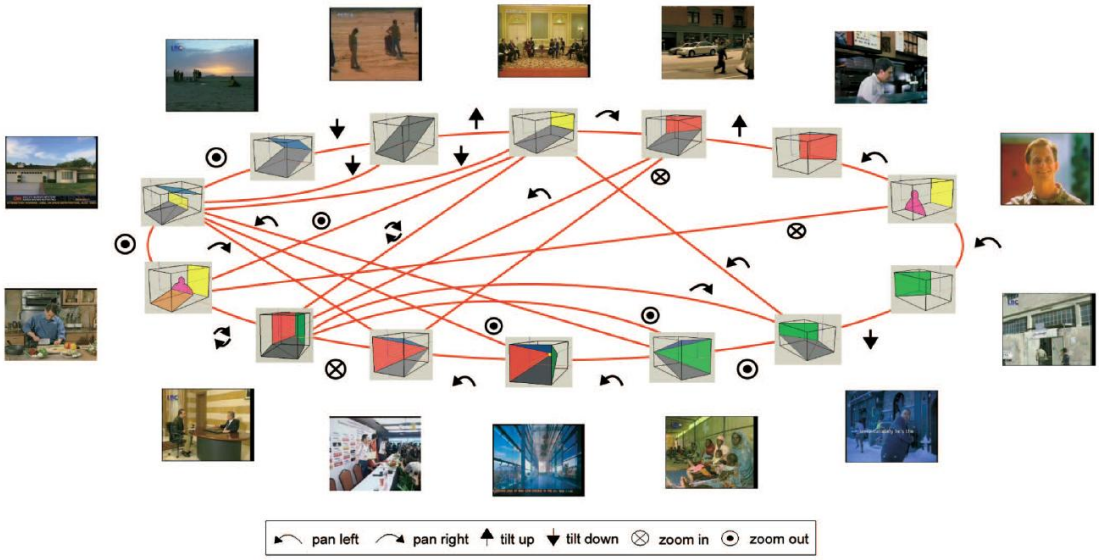


图 3.2-4 类别之间相互转换

注意上述 15 类中有一个特殊类别 no depth。这个类别主要是指图表、地图等无场景几何结构的图片。在我们的项目中，这对应于 3.1.2 节中“2D->3D 转换序列选择”所选出的不适合转成 3D 的帧。

Nedovic[3]等人利用 stage model，对图片按几何构造进行自动分类。而我们则是利用 stage model，将整幅图片划分成为几个面，并由此估计关于图片背景深度的信息。目前 stage model 通过用户手工标注得到。

3.2.1.2 用户交互设计

如前所述，按 stage model，我们需要将原图划分为几个面（如地面、天空、墙壁等）。对任意图片，初始化 stage model 为 5 个面（如图 3.2-5，为了方便观看，5 个面的边和顶点都已加粗显示）。用户可以通过拖动顶点来标注图像中的各个面。另外，可以在边上双击以增加顶点，或者把两个相邻顶点拖到一起来合并这两个顶点，以适应 stage model 的不同情况。（图 3.2-6）

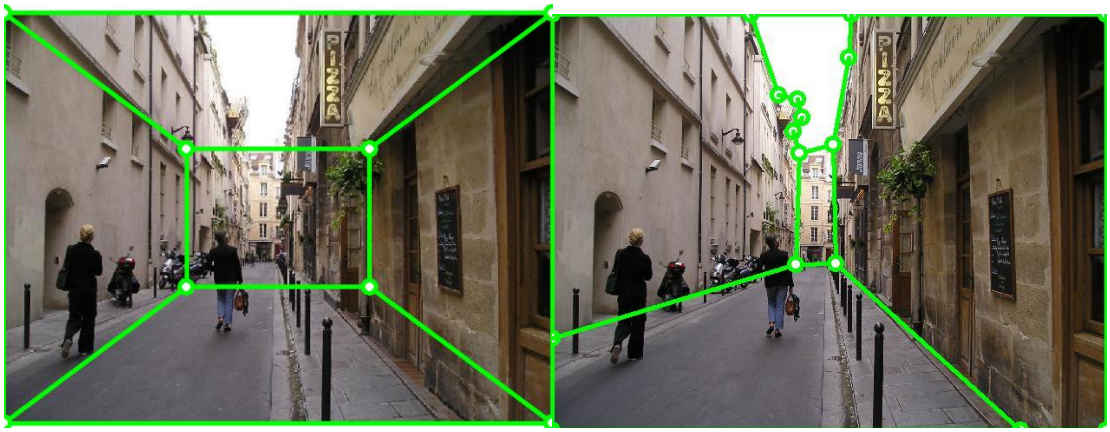


图 3.2-5 左：初始化 右：标注完成

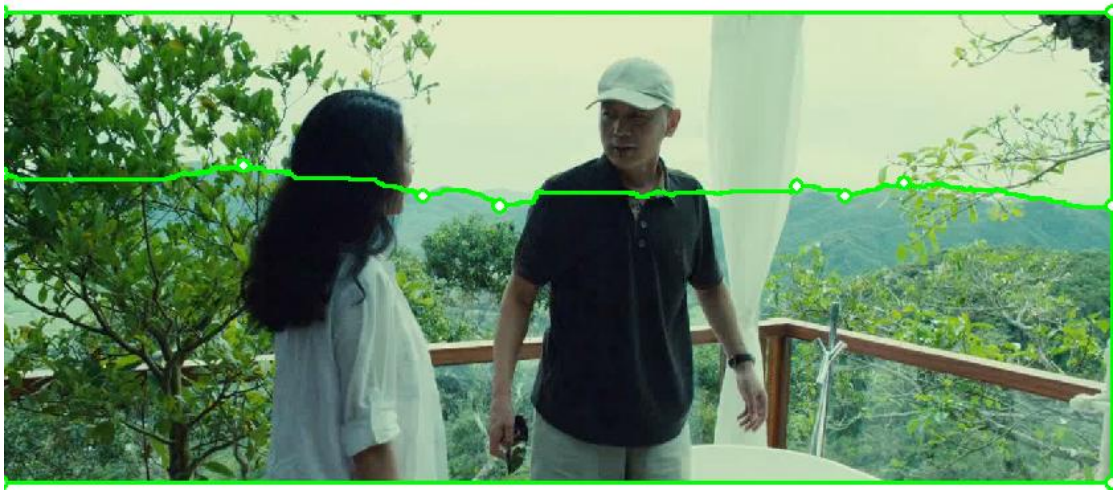


图 3.2-6 两个面的标注（注意顶点之间的线自动贴在了远处山的边缘）

自动贴边：为减少用户操作量，同时获得较精确的 stage model 划分，程序支持通过 intelligent scissors 算法“自动贴边”。即在用户拖动顶点时，划分 stage model 的边能自动贴合到图像中附近的边缘上。如图 3.2-6，注意划分上下两个面的线上，用户添加的顶点之间所连的并不是直线。对 intelligent scissors 算法的介绍可见第 4 章。

3.2.1.3 实现

为支持在边上双击增加顶点、合并相邻顶点等操作，将划分 stage model 的顶点和边看成一个无向图模型。初始化时图的顶点集和边集如图 3.2-7 左边所示。随着用户增加顶点、合并顶点，图的结构作相应改变。

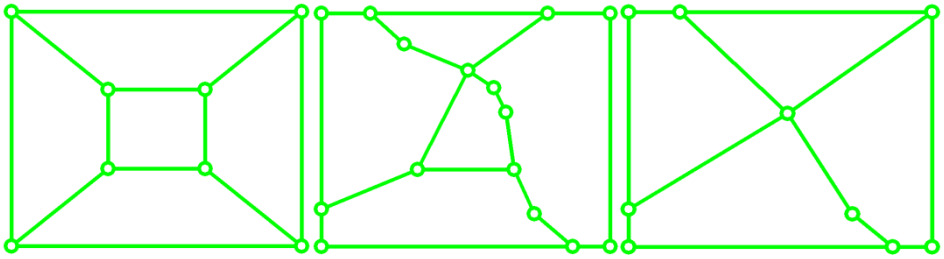


图 3.2-7 左：初始化时图模型 中、右：用户操作后的图模型

数据结构上采用类似于邻接表的方式实现。注意因为允许“自动贴边”，图中两顶点之间连线可以是曲线。另外，为简化图的结构，图中一系列相连的度为 2 的顶点及连接它们的曲线，可作为一条路径保存，该路径作为邻接表的边。只有度大于 2 的顶点才作为邻接表中的顶点。

在实现时，还需要考虑一些细致的问题，如边缘上的点在拖动时不允许其离开边缘；边缘上的点拖到图像角上时，应和角上的点合并，拖离角上时，又应重新插入图像角上的点（图 3.2-7 左、中）等等。

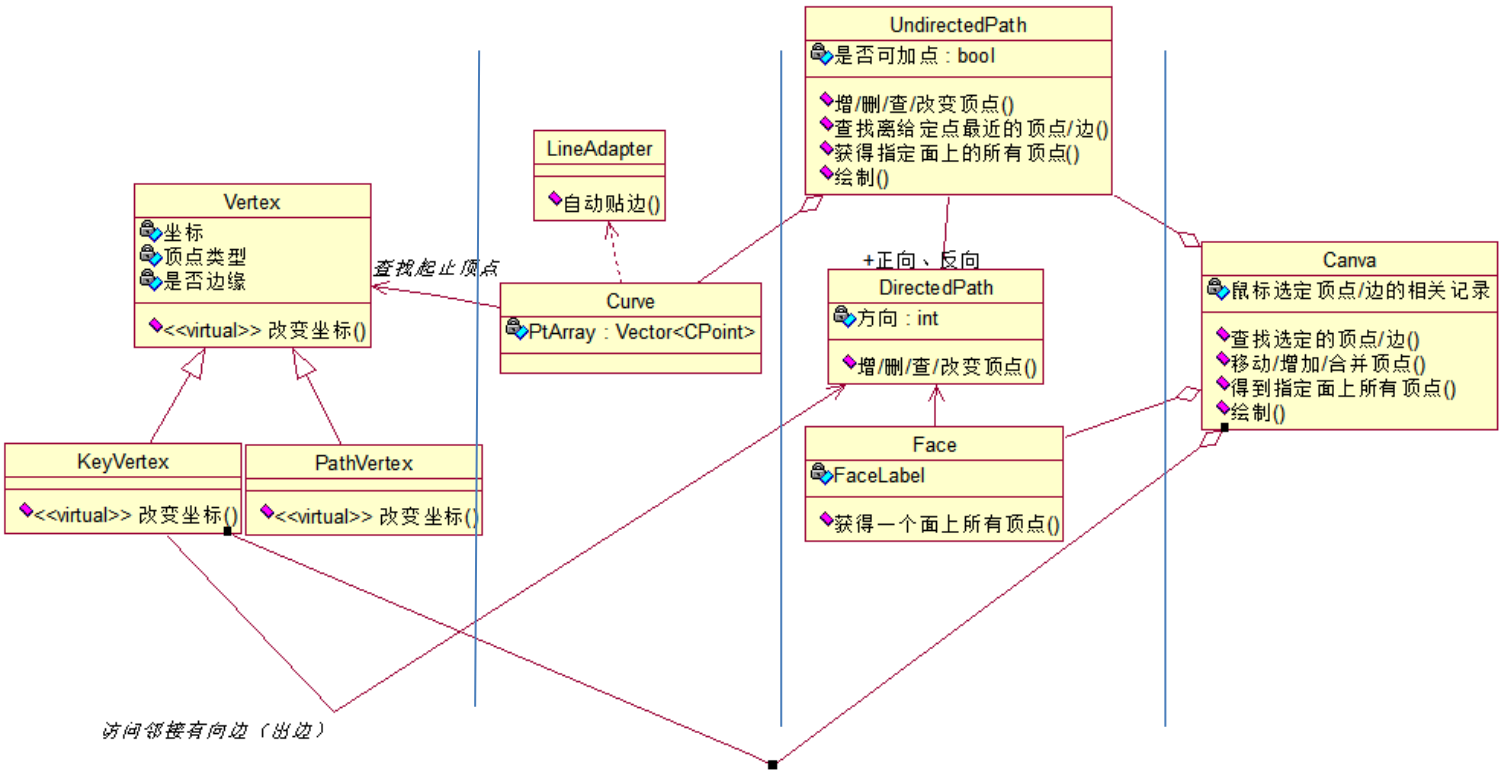


图 3.2-8 背景标注模块类图

因为该模块涉及的类较多，所以给出了该模块的类图（图 3.2-8）。为保持简明只列出了较重要的成员和方法。类图从左至右可分为四个层次。

1. 最左端的 Vertex 及其派生类表示用户可拖动的顶点（如图 3.2-7）。Vertex 是一个抽象基类，存储顶点坐标、是否在边缘上（用于限制用户拖动），以及顶点类型。“顶点类型”用于动态确定对象的派生类类型。KeyVertex 指图中度大于 2 的顶点，由它可以访问由该顶点出发的邻接路径。PathVertex 指图中度为 2 的顶点，它们作为一条路径的一部分储存。“改变坐标”作为虚函数实现的原因主要是因为在“自动贴边”过程中，用户移动 KeyVertex 将导致其邻接的路径变化；而 PathVertex 由 UndirectedPath（保存路径的类）通过 Curve 类管理，移动 PathVertex 所导致的变化由 UndirectedPath 类负责。

2. Curve 类与 Vertex 间有单向关联，即拥有指向开始、结束顶点的指针。它依赖 LineAdapter 类计算贴边过程中连接两点的曲线，PtArray 属性保存该曲线。

3. UndirectedPath 顺序储存一系列 Curve 对象。它保存一条路径的相关信息并负责对路径上的 PathVertex 及 Cruve 执行相关操作。但通过 KeyVertex 访问其邻接路径时，常常需要方向信息（如：从某 KeyVertex 处需访问某条路径上离它最近的 PathVertex。此时储存该路径的 UndirectedPath 需要判断待访问的 PathVertex 是它管理的 PathVertex 中正向遍历的第一个还是反向遍历的第一个）。为了方便这种访问，增加了 DirectedPath 类。它含有指向 UndirectedPath 对象的指针和一个方向标识。KeyVertex 含有指向 DirectedPath 的指针。当 KeyVertex 需进行上述访问的时候，直接调用 DirectedPath 的方法，DirectedPath 以方向为参数调用 UndirectedPath 中相应方法，返回结果。这样可以在每条路径的信息只保存一份的情况下实现有向图。

UndirectedPath 和 DirectedPath 存在双向关联，它们可以相互访问。

Face 类的作用是维护 stage model 中某个面与图上路径的对应关系。它记录了组成该面的 DirectedPath 指针。调用其方法可获得组成一个面的所有顶点。

4. Canva 类是该模块的对外接口。它管理所有 KeyVertex、UndirectedPath 和 Face 对象。外部调用其方法后，Canva 类可以视情况调用 UndirectedPath、Face、KeyVertex 等的相应方法，把任务分派给路径/面/顶点等低层次的类处理，并得到返回结果。

3.2.2 前景标注

对前景标注，我们采用 GrabCut 进行交互式前景分割。该方法通过迭代运行 graph cut，能在用户交互相对较少的情况下达到较好的分割效果。算法原理的介绍见第 4 章。

在用户交互方面，GrabCut 允许用户用一个矩形宽松地标出前景并进行分割，也可以在此基础上指定前景、背景像素等。

图 3.2-9 利用 GrabCut 的前景分割

在实现上，GrabCut 主要是调用 openCV 相关函数实现。设立 FrontObjInfo 和 FrontObjInfoCollection 类来管理和储存前景的相关信息，包括用户标注、前景轮廓等。

此外，对于 GrabCut 结果不太令人满意的地方（如边缘毛刺等），允许用户对边缘轮廓进行微调。

3.2.3 深度图生成

获得前、背景标注后，深度图生成模块可以首先根据背景标注估计背景的深度，然后根据前景在背景中的位置估计前景深度。如果前景深度估计不当，用户可手工进行微调。

背景深度图生成：目前根据背景标注获得深度图的方法非常简单。即首先找出地面的最近与最远点（地面顶点中图像 y 坐标最小与最大点），然后按最小深

图 3.2-10 背景标注图与对应深度图（颜色越深表示深度越小）

度到最大深度均匀对地面上的像素进行线性插值。对于左右两个面上的点，取其地面上对应点（图像中 x 坐标相同）的坐标作为其深度。（图 3.2-10）

这种方法非常简单，仅能估计出场景大致的相对深度关系。以后我们可能会用更精细的模型来对 stage model 标注的场景进行深度估计。另外，作为对这种深度估计方式的补足，我们还集成了自动场景深度估计的模块。这两种方式各有优劣，根据不同场景选择适当方式可以得到更好的效果。（详细介绍见 3.3 节 部分其他模块的集成或实现）

前景深度图生成：目前自动估计前景深度的方法，是将前景着地点的深度作为整个前景的深度。若前景无着地点，由于左、右两个面的深度都是由地面对应点的深度得到，所以仍能得出前景的深度。

图 3.2-11 左：前/背景标注完毕图像 右：总体深度图（颜色越深，深度越小）

当然，这样估计出的前景深度不一定准确。由于观众在观看立体视频时对前景的关注程度远高于背景，因此我们允许用户对前景深度进行手动调整。前景手动调整与“屏幕位置标注”模块一起，在一个 OpenGL 3D 窗口中完成。见下一节“屏幕位置标注”。

3.2.4 屏幕位置标注

在这个模块中，我们希望在 3D 空间中呈现出各个前景和屏幕的相对深度关系，用户可在 3D 场景中直观地查看、修改屏幕位置以及各前景的深度。另外，在调整前景深度的同时，深度图也会相应更新。

确定了屏幕位置和各前景位置，就确定了场景的出屏、入屏状况。由于 3D 视频中场景出入屏关系的重要性，我们用给原图像着色的方式直观地展现这一点。图 3.2-12 左侧即为出入屏关系示意图，其中红色表示出屏，绿色表示恰好在屏幕上，蓝色表示入屏。

图 3.2-13，图 3.2-12 展示了该模块调整出入屏设置的过程。

图 3.2-13 原图像（已标注好）

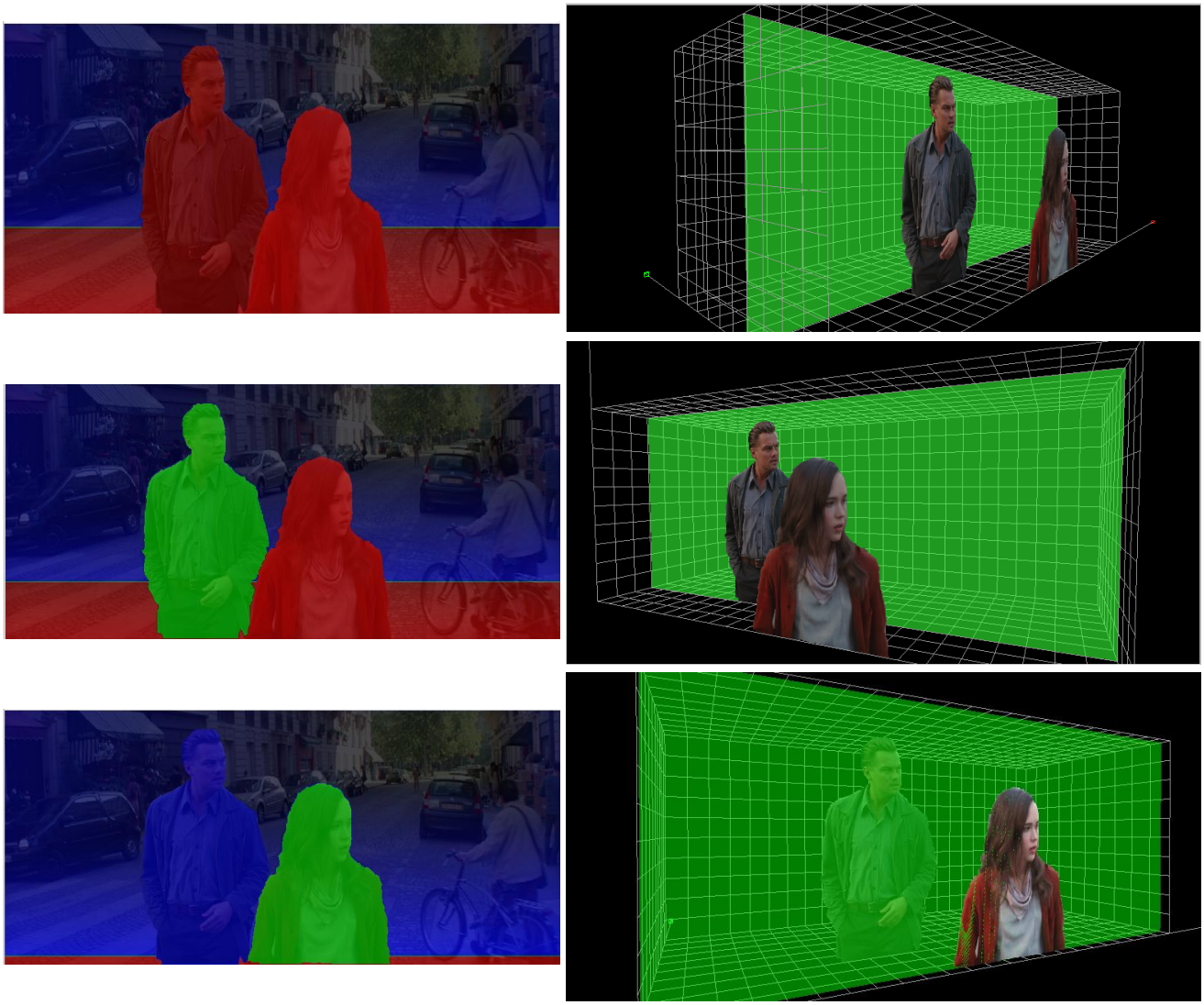


图 3.2-12 屏幕位置调整

左侧的图指示场景出屏/入屏状况。红色表示出屏，绿色表示在屏上，蓝色表示入屏。右侧的图是对应的 OpenGL 3D 空间视图。用户在此视图中调整屏幕及前景深度（半透明绿色平面表示屏幕位置）。三行图片从上至下分别表示前景均出屏、一个在屏上一个出屏，以及一个在屏上，一个入屏的情况。这里没有展示调整前景深度时，程序生成的前景深度图也同步更新的情况。左侧原图是宽屏图像，因为排版原因有一定变形。

实现上，3D 空间窗口采用 OpenGL 开发。用 OpenGL 的混合特性[4]实现半透明效果。用户可随意旋转、放缩场景。可用鼠标或键盘选中前景、背景或屏幕以便进行调整，被选中的物体会高亮显示。

3.2.5 视差图生成

得到图像的深度图和屏幕位置之后，我们便可以计算每个像素的视差（disparity）了。如 2.2 节所述，各像素的视差描述了像素立体显示时的立体效果。视差为负表示该像素出屏，为正则表示入屏。视差的绝对值体现了像素出/入屏程度的大小。在视差图生成过程中，我们也考虑了显示设备大小对于视差的影响。

用 dp(I)表示原图像 I 的深度图，对图像 I 中的像素 x，其视差 d(x)计算如下：

$$d(x) = s \cdot w \cdot \left(\frac{dp(x) - dp_{\min}(I)}{dp_{\max}(I) - dp_{\min}(I)} - f \right)$$

其中，w 是图像像素宽度，$dp_{\max}(I)$，$dp_{\min}(I)$ 表示深度的最大、最小值。$f(0 < f < 1)$即屏幕深度。屏幕深度越大，则 d(x)就会更多地被移向负值方向，使得像素出屏程度增加。s 是一个用于限制最大视差绝对值的控制因子，观看时较舒适的 s 值与立体设备屏幕大小及人眼到屏幕距离有关。我们目前将显示设备分为了 3 档，分别设置了 s 值。依据经验，对较大的立体显示设备（75 英寸以上），s 设为 0.005；对中等大小显示设备（45-75 英寸之间），s 设为 0.01；对较小显示设备，s 设为 0.025。

3.2.6 立体图像生成

得到视差后，我们可以将原图像转为左眼-右眼双路图像。（Nvidia 3D Vision 可直接显示这种格式）

左、右两路图像 I_r，I_l 中 x 像素的值可由下式计算：

$$I_r(x) = I(x + 0.5 \times d(x))$$
$$I_l(x) = I(x - 0.5 \times d(x))$$

d(x)是上文所得到的视差值。即相当于将像素 I(x)分别向左、右移动了 0.5 倍视差。

目前，对 I_r，I_l 中的每一个像素，都按上述公式到原图像中取值。这样有时会造成图像的变形。以后将进一步集成一些 impainting 的方法以减轻这种现象。

如果需要红青视图，如 2.2 节所述只需取左路图像的红色通道与右路图像的蓝绿通道组合即可。

图 3.2-14 展示了显示设备大小不同时生成的红青视图。可以注意到当显示设备较小时，视差较大。

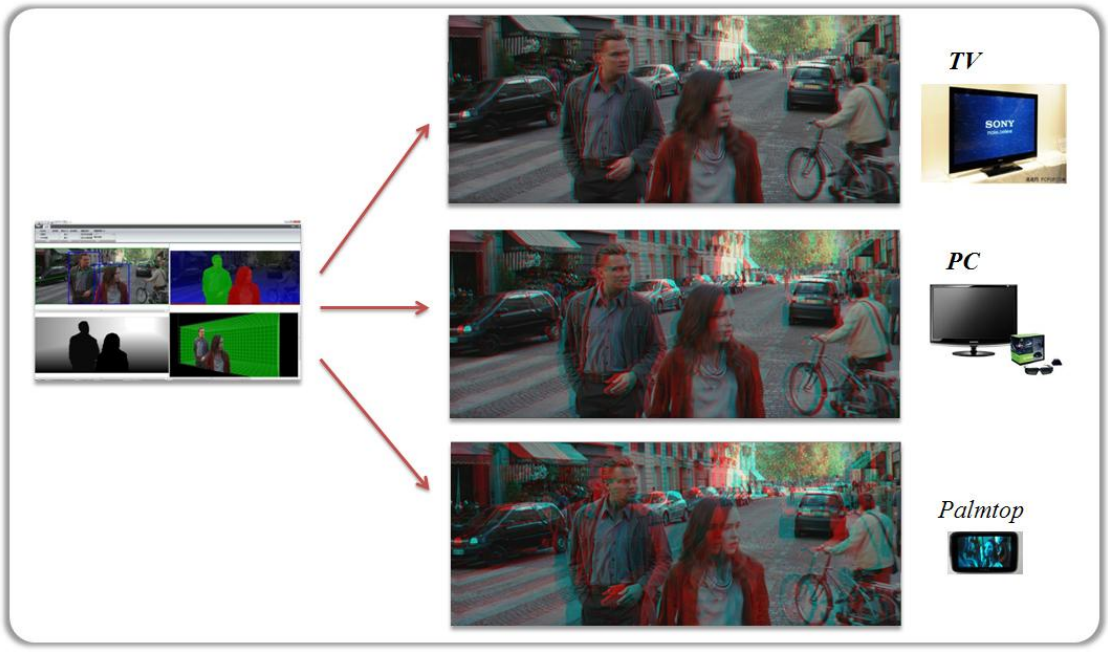


图 3.2-14 不同大小显示设备上的红青视图

3.3 部分其他模块的集成或实现

3.3.1 自动深度估计

在 3.2.3“深度图生成”一节中曾提到，我们在系统中整合了自动深度估计得模块，可以与原先背景标注和深度估计模块相互补足。

图 3.3-1 自动/手动深度估计对比

左上：原图像　　右上：自动估计深度与手工前景标注的深度图

左下：手动前背景标注的深度图　　右下：纯自动估计的深度图

图 3.3-1 说明了这个问题。可以看到，对原图像手工标注后生成的深度图（左下）虽能大致刻画场景深度走向，但对类似于树叶这一类景物，无论将其作为前景或背景进行标注，都很难较准确地刻画其深度。我们整合的自动深度估计模块恰好可以弥补这一缺陷（注意图 3.3-1 右下，自动深度估计模块能够较精细地给出图像左右两侧树叶的深度情况）。

然而，自动深度估计的结果常常会出现一些误差（如图 3.3-1 右下深度图中，左边女性衣服的深度被估计为接近最大深度）。如果这些误差并不严重，且出现在背景上，往往可以接受；但由于观众观看视频时注意力一般集中在前景上，因此如果误差出现在前景上，就会极大影响观感。所以我们可以在自动深度估计的基础上，手工标注前景深度，得到较好的深度图像（图 3.3-1 左上）。

另外，自动深度估计有一定局限性。对某些场景，自动深度估计不能得到正确的结果，此时可以用上文所述的手工标注流程进行转换。

3.3.2 系统其他必要模块

上文完整介绍了对单帧图像，进行手工交互式标注，生成立体图像的流程，也介绍了在此流程中整合自动深度估计模块的情况。以下是对系统一些附属模块的介绍。

文件保存方面，程序内部数据使用 boost 库的 serialization 功能保存，可以保存转换过程中的各种中间结果。

可以与视频镜头分割和聚类模块（另一个程序）通信，得到分割好的镜头并进行标注。生成的立体图像（左-右眼双路图）可以写入到一个视频中。

集成了标注信息帧间传递模块的一个初步版本，但该模块目前还不能实用化。

3.3.3 单帧标注转换界面的设计

作为交互式深度标注和立体图像生成的模块，提供高效、友好的用户界面，方便用户操作和查看结果是非常重要的。由于单帧的标注和立体转换中要求的用户交互较多，需呈现的中间结果也比较繁杂，所以单独用一节来详细介绍这部分界面的设计与实现。

对于用户界面，我们采用了正在逐渐流行 Ribbon（功能区）界面设计。Ribbon（功能区）是 Microsoft Office Fluent UI 的一部分，用来代替传统软件界面中的菜单、工具栏等元素。它首先出现在 Office 2007 及 2010 中，Windows7 的一些附加组件也应用了该界面，现在，越来越多的商业软件也开始应用这种界面（如 SnagIt 等），它也已被越来越多的人接受。

在 Ribbon 程序界面中，程序的所有命令以任务为导向，被分类存放在一组选项卡（Category）中，同一选项卡中的命令被归类组织到不同的面板（Panel）中，并以图标的形式呈现给用户。与传统菜单/工具栏软件相比，用户不需要从

层层级联菜单中寻找命令，可以有效地减少鼠标的点击次数，而且能更容易地找到所需的功能。

下面按单帧的标注过程展示用户界面。

0．用户打开视频，并进行镜头分割和聚类（镜头分割和聚类在另一个程序中进行，界面在这里没有展示）。

1．用户选择一个需标注的镜头，并选择一个关键帧。此时用户界面如图 3.3-2 所示。其中，底部一栏可预览该镜头中所有的帧。

上面 4 个视图的功能如下：

左上角视图：显示用户标注信息，包括前景的轮廓、包围盒以及背景的标注。双击此视图可以进入“**前景标注**”和“**背景标注**”模式（见后），用户也可以直接在此视图上进行一些微调。

左下角视图：显示该帧的深度图。用户可以选择显示以 stage model 方式标注背景、手动标注前景的深度图，或者以自动估计方式标注背景，手动标注前景的深度图。随着用户修改标注，该深度图会同步产生变化。可以看到，图 3.3-2 中目前显示了对原图像自动估计生成的深度图。

右下角视图：微调屏幕和前景位置的 3D 窗口。因为此时尚未标注前景，所以窗口中表示屏幕的半透明绿色平面外没有其他内容。

右上角视图：显示场景出入屏状况。当用户在右下角视图中对屏幕位置进行调整时，右上角视图会同步变化。

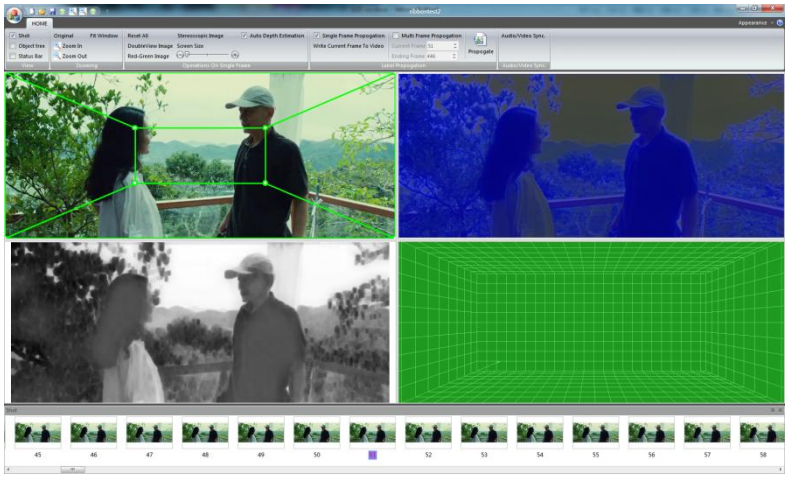


图 3.3-2 选定需标注关键帧后的主界面

2．背景标注：双击主界面左上角视图，可进入前景或背景标注模式。

图 3.3-3 展示了背景标注模式。

3．前景标注：图 3.3-4 展示了前景标注的界面。另外，如果用户对分割出的前景轮廓不满意，可以对轮廓进行逐点的微调。

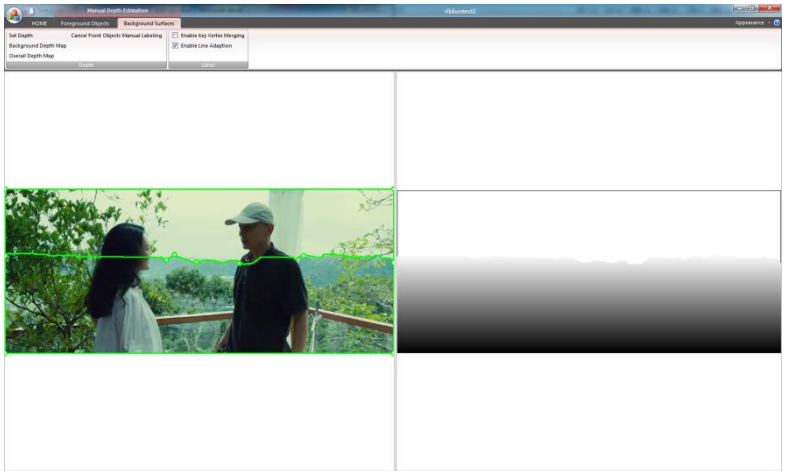


图 3.3-3 背景标注界面

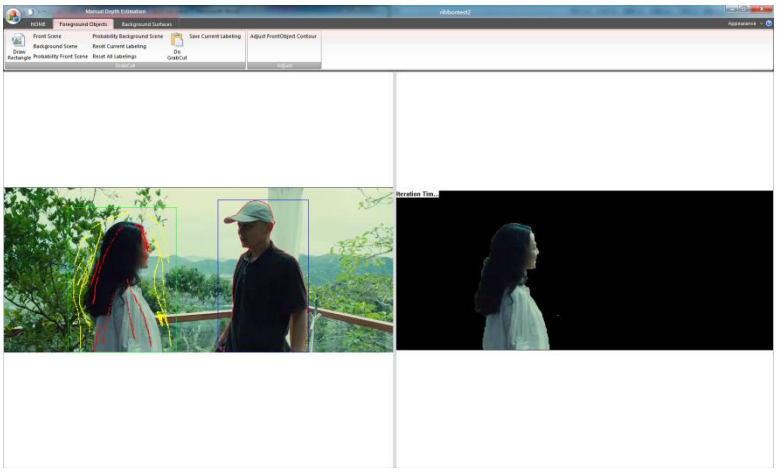


图 3.3-4 前景标注界面

4．完成前背景标注后的主界面：

图 3.3-5 标注完成后的主界面（stage model 估计背景深度）

图 3.3-6 标注完成后的主界面（结合自动估计的背景深度）

图 3.3-7 适当调整前景和屏幕位置后的结果

其中图 3.3-5、图 3.3-6 分别展示了以 Stage model 估计背景深度和以自动算法估计背景深度的结果。图 3.3-7 展示了适当调整前景和屏幕位置后的结果。可以看到前景中女性出屏，男性在屏上，且部分树叶也有出屏效果。

5．双目视图与红青视图

图 3.3-8 左-右双目视图

图 3.3-9 红青视图

图 3.3-8 和图 3.3-9 展示了最终生成的双目视图和红青图 。

第 4 章　系统所用部分算法总结

第 3 章展示了整个系统的情况，并详细介绍了场景深度估计及视差转换模块的设计和实现。其中使用了两个图像分割的算法。基于区域的 GrabCut，在系统中用于前景分割；基于轮廓的 intelligent scissors，用于背景 stage model 标注的“自动贴边”。下面将介绍这两个算法。

4.1　GrabCut

作为近年来较受关注的图像分割算法，GrabCut 的资料和实现都可以从网上找到。由于项目时间要求我们没有自己实现 GrabCut，而是直接使用了 OpenCV 相关函数。下文是对 GrabCut 的总结。

4.1.1 Graph Cuts

Boykov 和 Jolly 于 2001 年提出的 Graph Cuts[5]是 GrabCut 的基础，因此有必要先介绍该算法。在 Graph Cuts 中，用户在图像上分别指定一些前景点和背景点，这些点既对可能的分割做出了限制（用户指定的前/背景点必须被分别划分为前/背景），也提供了前/背景的颜色信息。然后，该算法可以综合利用前/背景颜色信息和图像中的边缘信息，算出分割的结果。

算法基本思路是把图像分割问题形式化为能量函数的最小化问题，然后将能量函数最小化问题转化为求解图的最小割问题。求得的最小割对应能量函数的最小值，从而对应一个“最优”的分割。

Boykov 和 Jolly 2001 年的论文中只能实现对单色图像的分割。

4.1.1.1 能量函数

原图像的一种前/背景分割可用二值向量 $A=(A_1,...,A_n...,A_N)$ 表达。其中下标 n 表示图像的第 n 个像素，N 表示图像中总像素数。$A_n \in \{0，1\}$，0 表示该像素为背景，1 为前景，后文也直接记作”bkg”或”obj”。此时 Graph Cuts 所使用的能量函数可写成：

$$E(A)=R(A)+\gamma \cdot B(A)$$

其中，

$$R(A)=\sum_n R_n(A_n)$$

$$B(A)=\sum_{(m,n)\in C} B_{\{m,n\}}\cdot\delta(A_m,A_n)$$

各项的含义可理解如下：$R_n(A_n)$表示将像素 n 分为前景或背景所产生的惩罚，R(A)是所有像素该惩罚值的和，称为 region term。$B_{\{m,n\}}\cdot\delta(A_m,A_n)$表示将相邻的像素 m，n 划为边界所产生的惩罚值。$\delta(A_m,A_n)$是一个 indicator function，$A_m\neq A_n$时其值为 1，否则其值为 0；求和符号中的 C 表示某种邻域关系，图像处理中一般定义其为 4 连通或 8 连通。B(A)是对所有相邻像素对惩罚值的求和，称为 boundary term。能量函数中γ是一个常数，用于调整 region term 和 boundary term 的相对权重。

我们可以通过适当定义 $R_n(A_n)$和$B_{\{m,n\}}\cdot\delta(A_m,A_n)$，使得令能量函数最小化的 A 对应一个最优的分割。在[5]中，相应项被定义为：

$$R_n(A_n="obj")=-\ln\Pr(I_n\mid O)$$
$$R_n(A_n="bkg")=-\ln\Pr(I_n\mid B)$$
$$B_{\{m,n\}}=\exp(-\frac{(I_m-I_n)^2}{2\sigma^2})\cdot dist(m,n)^{-1}$$

其中 I_n 表示像素 n 的 intensity；$\Pr(I_n\mid O)$和$\Pr(I_n\mid B)$分别表示该颜色在前景/背景中出现的条件概率（通过统计用户标注的前、背景点的直方图即可计算此概率）；$dist(m,n)$表示两像素间的欧式距离；σ控制 $B_{\{m,n\}}$惩罚值的大小，当$|I_m-I_n|<\sigma$，会产生大的惩罚值。

直观看，上述定义使得像素颜色与前/背景颜色相似时，相应划分为前/背景产生的惩罚值小，相邻像素颜色差异较大时，划分为 boundary 产生的惩罚值小。所以按上述定义，得到的分割应是比较好的。

4.1.1.2 转化为 min-cut

上述形式能量函数的最小化问题可以容易地转化为图的最小割问题。如图 4.1-1 所示，图的顶点集 V 为原图中所有像素，加上源点和汇点 S 和 T。设 S 代表"object"，T 代表"background"。边集 E 由两部分组成，相邻像素间所连的边（称为 N-link），另外每个像素与 S、T 各连一条边（称为 T-link）。

边的权值：对 N-link 的权值，边{m, n}权值为$\gamma\cdot B_{\{m,n\}}$。对 T-link，权值如下表所示：

像素 n 的类型	{n, S(obj)}权值	{n,T(bkg)}权值
用户标为前景	K	0
用户标为背景	0	K
未知	$R_n(A_n="bkg")$	$R_n(A_n="obj")$

K 为充分大的数。在[5]中，$K=1+\max_n\sum_{m:\{m,n\}\in C} B_{\{m,n\}}$

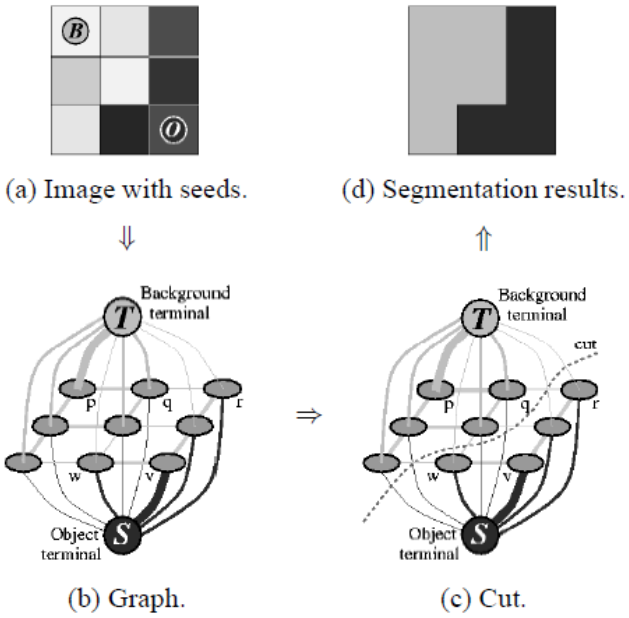


图 4.1-1

可以证明，在按上述方式构造的图上，从源点 S 到汇点 T 的最小割的总权值对应能量函数的最小值。且在去掉了最小割集的边之后，每个像素根据自己 t-link 是连在 S 还是 T 上，可确定自己属于前景还是背景，从而确定一个分割。下面证明这两点。

首先引入可行割集的概念。原图上的可行割集C'是满足如下性质的割：

1．对每个像素 n，C'一定包含 n 的一条 t-link。

2．$\{m,n\}\in C'$，当且仅当 m，n（去掉最小割集后）的 t-link 连到不同的终点。

3．如果 n 为用户标注的前景点，则$\{n,T\}\in C'$(与 background terminal 相连的边被切掉)

4．如果 n 为用户标注的背景点，则$\{n,S\}\in C'$(与 object terminal 相连的边被切掉)

显然，每个可行割都可导出图像上的一个分割。像素根据自己去掉割集中的边后与 S 还是 T 相连，就可确定是属于前景还是背景。

第二，可证明，该图上的最小割集C是可行割集：

1．对每个像素 n，C一定包含 n 的一条 t-link。

显然C至少包含 n 的一条 t-link，否则 S 和 T 就连通了。如果C包含了 n 的两条 t-link，则一定可以从C中去掉其中一条，且使C仍为割（证明该结论：如果去掉{n,S}且C仍为割，则成立。如果去掉{n,S}后C不为割，则说明存在从 n 到 T 的路径(n,T)，使得从C中去掉{n,S}后 S、T 通过 n 连通了。这时我们选择从C中去掉{n,T}。如果去掉{n,T}后C也不为割，同理可说明存在从 n 到 S 的路径(n,S)。则在去掉最小割C的情况下，图中 S 和 T 还可通过路径(n,S)、(n,T)连通，与C是割矛盾）。所以该性质成立。

2．$\{m,n\}\in C$，当且仅当 m，n（去掉最小割集后）的 t-link 连到不同的终点。

必要性：显然，如果 m，n t-link 到不同终点而$\{n,m\}\in C$，则 S，T 连通，与 C 是最小割矛盾。

充分性：如果$\{m,n\}\in C$且 m, n t-link 到相同终点，可证明可以从C中去掉{m,n}后，C仍为割集，与C为最小割矛盾。

3．如果 n 为用户标注的前景点，则$\{n,T\}\in C$(与 background terminal 相连的边被切掉)

由性质 1，若$\{n,T\}\notin C$，则$\{n,S\}\in C$。由于此时{n,S}取足够大的值，可以把C中{n,S}替换为{n,T}以及与 n 相连的所有 N-link。此时 n 除与 S 相连外孤立，因此C仍是割而总权值降低。所以$\{n,T\}\in C$。

4．如果 n 为用户标注的背景点，则$\{n,S\}\in C$(与 object terminal 相连的边被切掉)与性质 3 同理可证。

所以最小割一定对应一个合适的图像分割。

第三，可证明最小割的总权值对应能量函数的最小值。

任意可行割C'的权值可计算如下：

$$|C'|=\sum_{n\notin O\cup B}R_n(A_n)+\sum_{\{m,n\}\in C}B_{\{p,q\}}\cdot\delta(A_m,A_n)$$

注意第二个求和号中的 C 仅表示邻域关系。O 和 B 分别是被用户标为前、背景的点。

显然，

$$E(A)-|C'|=\sum_{n\in O}R_n(A_n)+\sum_{n\in B}R_n(A_n)$$

由于需满足用户标注的限制，$n\in O\cup B$时，A_n为固定值。因此$E(A)-|C'|$为常数。所以原图上的可行割集与满足用户标注限制的能量函数的值有一一对应关系。所以，求出图的最小割集等效于最小化了能量函数。

所以综上可知最小割对应于原图像一个合适的分割，而且可以使能量函数最小化。

对于求解最小割的过程，原作者给出了一个比较高效的代码。我暂时还没有细致地总结。

4.1.2 GrabCut

GrabCut[6]对 Graph Cuts 的改进主要有两点。第一是将能量函数中 Region term 换成了 GMM，以便描述彩色图像像素的颜色分布（将 Graph Cuts 中所用的 Histogram 直接扩展到 RGB 空间中，将有 2^{24} 个 bins，显然不合适），相应的，Boundary term 中衡量颜色距离的$(I_m-I_n)^2$换成了颜色空间中的欧式距离。第二是

使用 EM 的思想，交替使用统计出的颜色分布做 Graph Cuts，又用 Graph Cuts 结果修正颜色分布，如此迭代直到收敛。

第一点改进使 Graph Cuts 可以应用于彩色图像。第二点改进使 GrabCut 允许不完整的标注，减少了用户交互量。

以下详细介绍算法的步骤。

4.1.2.1 Gaussian Mixture Model (GMM)

GMM 是做无监督学习的常用模型。这里我们用它来描述前/背景像素颜色的分布情况。

在 GMM 模型中，样本 x 的概率密度用下式表示：

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k)$$
$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

可以看到，p(x)表示为 K 个正态密度函数的加权和。在 K 足够大的情况下，理论上 GMM 可以拟合任何分布。上式中，第 k 个正态密度函数称为第 k 个分量。权重 p(k)或 π_k 可理解为出现某分量的先验概率。故 π_k 对所有 k 求和为 1。

GMM 的训练：

现在的问题是，如果分量数 K 已知，给定的样本数据集合 X 已知，而 GMM 各分量的权重 π_k，均值 μ_k，协方差矩阵 Σ_k 均未知，如何估计出合适的 π_k, μ_k, Σ_k。

1. EM 方法[7,8]：

[7][8]两本教材中描述了这种方法。这里我们希望求出能使产生样本数据集合 X 的概率最大的 π_k, μ_k, Σ_k，即最大似然估计。

用 π, μ, Σ 分别表示三种参数的向量。X 的对数似然函数可用下式表示：

$$\ln p(X \mid \pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathrm{N}(x_n \mid \mu_k, \Sigma_k) \right)$$

其中 n 从 1 到 N 求和是对所有样本求和。

记第 i 个样本属于第 k 类的后验概率为

$$\gamma(i,k) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

加上 π_k 对 k 求和为 1 的条件，用拉格朗日乘子法求解上面对数似然函数可得：

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(i,k) x_i$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(i,k)(x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k = N_k / N$$

其中，$N_k = \sum_{i=1}^{N} \gamma(i,k)$ 是所有样本属于第 k 类的后验概率的和。

上述式子的数学形式虽然稍复杂，但都可以从直观上解释。后验概率 $\gamma(i,k)$ 可看作样本 i 属于分量 k 的“程度”。则均值 μ_k 就是所有样本按属于 k 分量程度的加权平均。协方差矩阵有类似解释。而 N_k 可看作是概率意义上属于第 k 类样本的数目，π_k 也由此可直观理解。

注意以上形式中左右两端都含有参数。一般来说都是用 EM 方法来寻求一个局部最值。即：对参数先设定一个初值，然后计算其后验概率 $\gamma(i,k)$；得到后验概率后再重估 π_k, μ_k, Σ_k。

注意使用 EM 方法求解 GMM 时可能遇到奇异（singularity）情况。即：当某分量均值恰好落在某样本上时，随着该分量方差趋于 0，整个似然函数值趋于无限大。当遇到这种情况时，一种解决方案是可以重新设定该分量的参数并继续迭代过程。

2. 其他

在 Justin F. Talbot.[9]提供的 GrabCut 实现中，使用了一种分层次聚类的方法得到 GMM 的参数。该算法来自[ORCHARD, M. T., AND BOUMAN, 1991][10]，基本思路是首先把所有样本点估计为一个高斯密度函数，然后找到该密度函数方差最大的方向（对应最大特征值的特征向量），以一个垂直该方向的超平面将样本分成两部分，再对这两部分样本分别估计高斯密度，以后不断从各高斯密度函数中选出方差最大的方向并进行分裂，作分层聚类。[9]中表示这样做的理由是为使前景和背景的密度分布尽量分开，高斯密度的方差应尽可能小。

4.1.2.2 GrabCut

GrabCut 算法可分为初始化和迭代优化两部分。

初始化：

1. 用户初始化一个 Trimap（将原图像像素标记为前景\背景\未知的矩阵）。与 graph cuts 不同的是可以只提供 T_B（背景标记）（也可以相反，只提供前景标记，后续步骤也做相应改变）。其他像素都标为 T_U（未知）
   注意，如果用户没有 refine 或者 edit 的话，在迭代过程中 trimap 是不变的。它的主要作用就是记录用户标记作为分割的限制条件。
2. 初始化 GrabCut 分割结果 α。将像素 $n \in T_B$ 都标为 α_n=0（背景），$n \in T_U$ 标为 α_n=1（前景）
3. 利用初始化的分割结果 α_n 训练前、背景 GMM。根据[6]，分量数 K 可取为 5。

迭代：

先介绍 GrabCut 的能量函数。GrabCut 以迭代过程寻求该函数的极小值。

GrabCut 所用能量函数基本形式与 Graph Cuts 完全相同。只是 Region Term 和 Boundary Term 的定义有所改变。

对像素 n，其 Region Term 为：

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2}\log\det\Sigma(\alpha_n, k_n)$$
$$+\frac{1}{2}[z_n - \mu(\alpha_n, k_n)]^{\top}\Sigma(\alpha_n, k_n)^{-1}[z_n - \mu(\alpha_n, k_n)]$$

其中 $\alpha_n, k_n, \underline{\theta}, z_n$ 表示决定 D 的参数。α_n 表示 n 被标注为前景还是背景；k_n 表示 n 对应的 GMM 分量（见下文步骤 4）；$\underline{\theta}$ 指该分量的参数；z_n 指向素颜色值。虽然表达式形式上较复杂，但其实就是当 n 取前/背景时，前/背景 GMM 的 k_n 分量产生该颜色的概率取负对数。

Boundary Term 为：

$$\mathrm{B}(m,n) = \frac{\gamma}{dist(m,n)} e^{-\beta\|z_m - z_n\|^2}$$

与 Graph Cuts 相比几乎没有变化，只是替换了颜色的距离度量。$\| Z_m - Z_n \|^2$ 表示颜色空间中两点的欧式距离。γ 仍是衡量 Region Term 和 Boundary Term 权重的常数，[6]的建议值是 50。

$\beta = \frac{1}{2\langle \|z_m - z_n\|^2 \rangle}$，$\langle \bullet \rangle$ 表示求期望。其中 $\langle \bullet \rangle$ 的内容就是 graph cuts 公式中的 σ^2。

接下来是迭代步骤（接初始化）：

4. 每个像素（前景或背景）被分配给相应 GMM 似然概率最大的分量。
5. 前/背景 GMM 的每个分量，利用分配给它的样本更新其参数。均值和方差被简单估计为样本均值和方差，权重估计为分量样本数/GMM 样本总数。
6. 做 Graph Cut。
7. 返回 4 迭代，直到收敛。

注意在整个迭代过程中，Boundary Term 并不改变。迭代过程中的 4，5，6 步可看作分别求极小化 Region Term 的 $k_n, \underline{\theta}$ 和 α_n。多次迭代后整个函数应（至少）收敛到一个局部极小值。

因为我们的 GrabCut 算法是使用 OpenCV 的相关函数，所以这里没有展示更多用该算法的分割结果。该算法结果的展示可见图 3.2-9 和图 3.3-4。

4.2 Intelligent Scissors

[MORTENSEN, E., AND BARRETT. 1995][11]提出的智能剪刀算法，在系统中被用来实现背景 Stage Model 标注时的“自动贴边”功能。

该算法的思路是将“自动贴边”的问题转化为用户给出起点、终点后，在图上寻找最短路径的问题。希望这条最短路径恰好对应图像的边缘。

算法将原图像的每个像素看作图的顶点，图的边集则是按 8-连通规则，在相邻像素之间连接的有向边。要使最短路径能对应到图像边缘，关键在于定义合适的边的权值。

在[11]中，主要利用了三种图像特征来定义有向边的权值，如下表所示：

图像特征	表示
Laplacian zero crossing	f_z
梯度方向	f_D
梯度大小	f_G

一条从 p 到 q 的有向边的权重 l(p, q)，被定义为三种特征的加权和：

$$l(p,q) = \omega_Z \cdot f_Z(q) + \omega_G \cdot f_G(q) + \omega_D \cdot f_D(p,q)$$

其中 ω_Z，ω_G，ω_D 是三种特征的权重，在[11]中，分别取为 0.43, 0.43, 0.14。显然，我们需要合适定义 f_z，f_G，f_D，使得沿图像边缘方向的边，其权值能达到最低。

下面介绍 f_z，f_G，f_D 的详细定义：

1. Laplacian Zero Crossing：

拉普拉斯算子形式如下：

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

它是原图像 x 方向和 y 方向二阶导数的和。

实践中，经常以下面两种卷积核进行拟合：

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

当然，也可以用更高阶的卷积核计算。

laplace 算子的结果刻画了原图像的二阶导数，因此此时结果中穿过 0 的点对应于一阶导数中的“波峰”，因而应对应图像的边缘（一维的示意图见图 4.2-1）。

在实际使用中，图像颜色小的波动也可能导致出现 laplace zero crossing，因此在做边缘检测时经常附加上其一阶导数（或 zero crossing 两边像素值差异）大于一定阈值的条件。另外，对于 zero crossing 出现在两相邻像素之间的情况，取

绝对值小的像素作为 0 点。

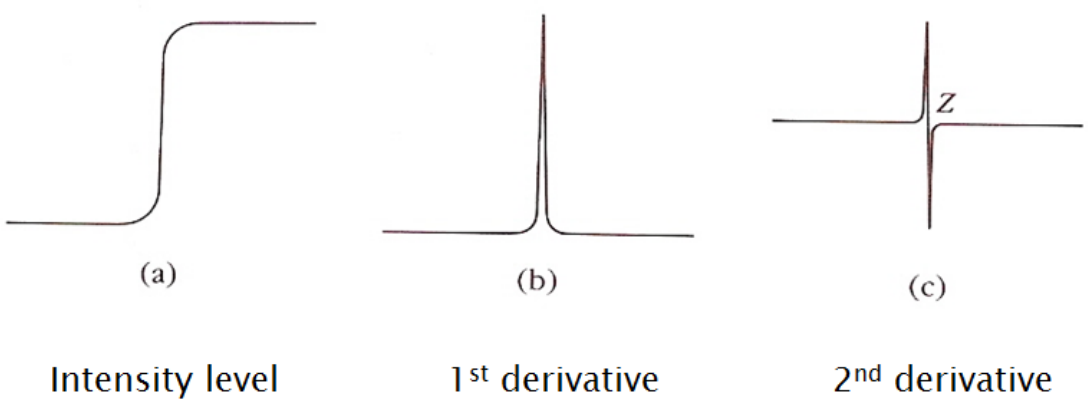


图 4.2-1 一维 laplacian zero crossing 示意

在[11]中，laplacian zero crossing 作为反映边缘是否存在的特征。若有边缘存在，$f_z(q)$应为 0。因此，$f_z(q)$定义为：

$$f_Z(\boldsymbol{q}) = \begin{cases} 0; & \text{if } I_L(\boldsymbol{q}) = 0 \\ 1; & \text{if } I_L(\boldsymbol{q}) \neq 0 \end{cases}$$

其中$I_L(q)$是用 laplacian operator 卷积后像素 q 处的值。

2. 梯度大小：

以 3 阶 Sobel 算子计算图像在 x，y 方向的导数I_x，I_y。梯度大小为$\sqrt{I_x^2 + I_y^2}$。

我们希望梯度越大，权值越小。所以$f_G(q)$定义为：

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

梯度大小这一特征用来反映边缘的强度。

3. 梯度方向

梯度方向即由 x，y 方向导数I_x，I_y组成的二维向量的方向。令$D(p)$为表示 p 点梯度方向的单位向量，$D'(p)$表示$D(p)$顺时针旋转 90 度后的向量（即$D(p)=[I_x, I_y]$，则$D'(p)=[I_y, -I_x]$），则$f_D(p,q)$定义如下：

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

其中，

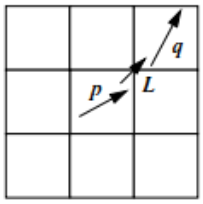
$$d_p(p,q) = D'(p) \cdot L(p,q)$$
$$d_q(p,q) = L(p,q) \cdot D'(q)$$

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

这个定义虽然看起来比较复杂，但也有直观的解释。$D'(p)$是梯度向量旋转 90 度后的向量，可以看作像素 p 处边缘的走向；$d_p(p,q)$为 p 处边缘走向与{p,q}边的内积，acos($d_p(p,q)$)即两者夹角，acos($d_q(p,q)$)的含义类似，是{p,q}边与 q 处边缘走向的夹角；最后 L(p, q)的定义只是用来保证$d_p(p,q)$>0（即 acos($d_p(p,q)$) <=π/2）。

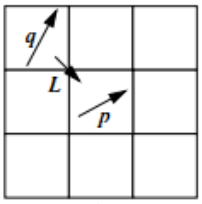
所以 $f_D(p,q)$即 p，q 处边缘走向和{p,q}边夹角之和。直观上，当 p，q 边缘走向与{p,q}方向相似时，$f_D(p,q)$最小；p，q 边缘走向与{p,q}均差异较大时，$f_D(p,q)$较大。

图 4.2-2 来自[12][12]，说明了上述解释。



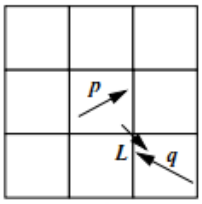
(a)

$D'(p) = [0.870, 0.492]$
$D'(q) = [0.473, 0.881]$
$L(p,q) = [0.707, 0.707]$
$d_p(p,q) = D'(p) \cdot L(p,q) = 0.964$
$d_q(p,q) = L(p,q) \cdot D'(q) = 0.957$
$f_D(p,q) = \frac{2}{3\pi}\{\text{acos}[d_p(p,q)] + \text{acos}[d_q(p,q)]\} = 0.120$



(b)

$D'(p) = [0.870, 0.492]$
$D'(q) = [0.473, 0.881]$
$L(p,q) = [0.707, -0.707]$
$d_p(p,q) = 0.267$
$d_q(p,q) = -0.288$
$f_D(p,q) = 0.671$



(c)

$D'(p) = [0.870, 0.492]$
$D'(q) = [-0.881, 0.473]$
$L(p,q) = [0.707, -0.707]$
$d_p(p,q) = 0.267$
$d_q(p,q) = 0.957$
$f_D(p,q) = 0.880$

图 4.2-2 f_D 定义的示意图 注意最后一栏 $d_p(p,q)=0.957$ 应改为-0.957

以上介绍了定义边权值所用的三种特征。在得到所有边的权值后，可以用 Dijkstra 等算法求最短路径的方法求出最优路径。此最优路径就是贴合到图像轮廓上的路径。

图 4.2-3 和图 4.2-4 展示了启用自动贴边的效果。

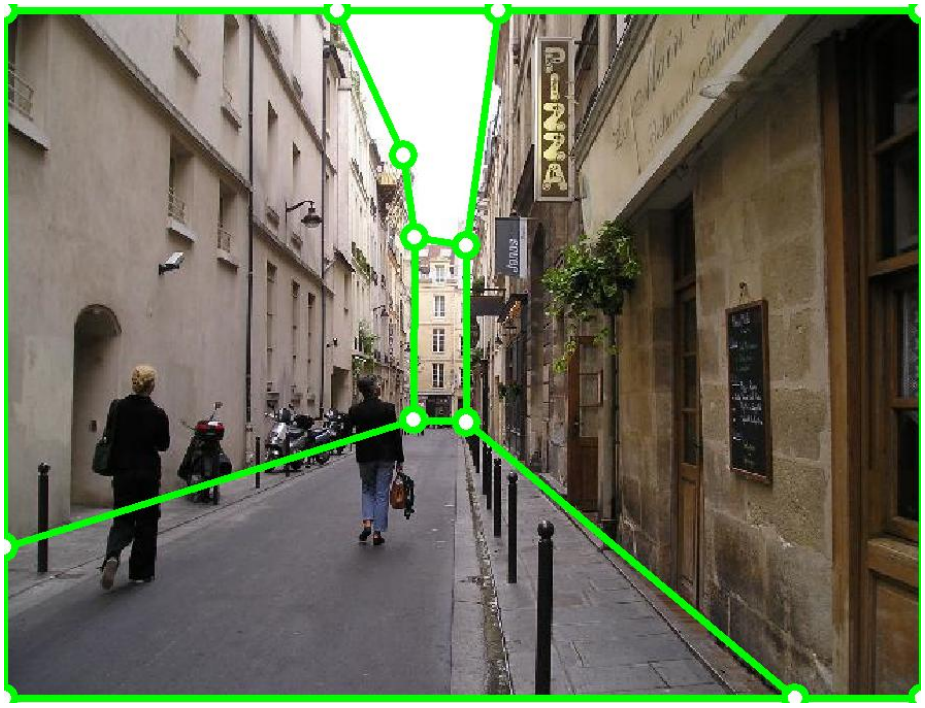
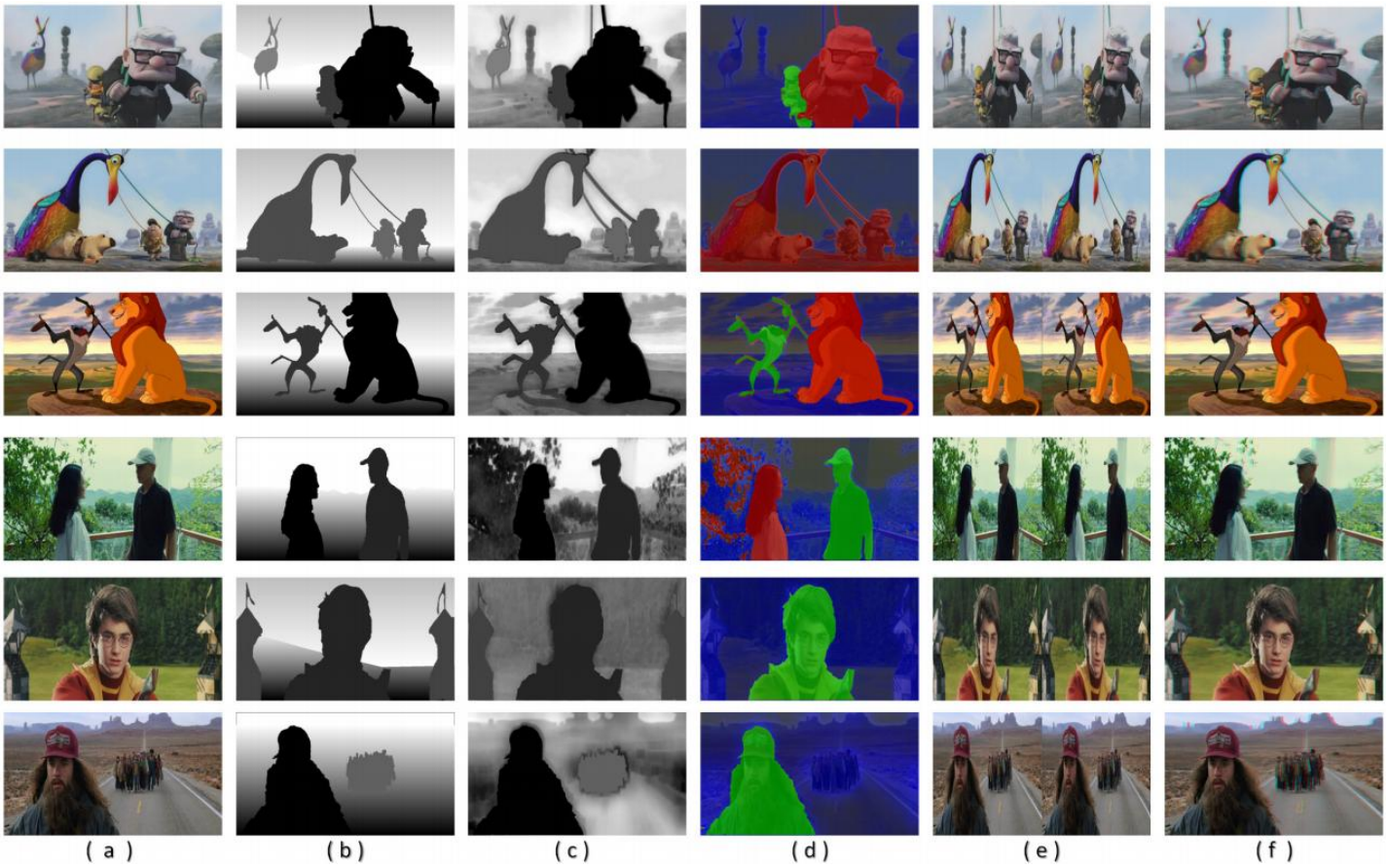


图 4.2-3

图 4.2-4

第 5 章　实验结果

图表 5-1 显示了部分转换的结果。



图表 5-1 部分转换结果

上图中(a)列是原始图像，(b)列为手工标注的深度图，(c)列为自动估计与手动前景标注结合生成的深度图，(d)为出/入屏示意图，(e)为双目视图，(f)为红青图

由于目前缺少 ground truth 数据，暂时只能直观地分析结果。可以看到，目前的单帧标注转换模块对很多图片有较好的转换效果，同时在前景深度和出入屏设置上也给了用户较大的自由度。当然，目前该模块也有一些缺陷，存在着手动标注不能很精确地反映场景结构，而自动深度估计又出现错误，导致不能得到很好的转换结果的情况。对此，我们还需要进一步改进。

第 6 章 总结

作为毕业设计的内容，完成了 2D 到 3D 视频转换系统中对单帧进行标注合转换的模块和系统的一些集成工作。其中单帧标注转换模块可分为背景标注、前景标注、深度图生成、屏幕位置标注、视差图生成及立体图像生成几个子模块；除此之外，还集成了自动深度估计模块，以便与手工标注相互补充。

目前已经有了初步的成果。对于单帧 2D 图像，能够通过用户的一些交互式标注，将其转化为立体图像；对于 2D 视频，目前可以通过逐帧手工标注将其转化为 3D 视频。但整个系统离实用化的目标还有一定距离。下一步的工作，一是对单帧标注转换模块做进一步改进，如提高程序的易用性、在深度估计中采用更加精确的模型等等；二是完成或集成系统其他模块，如标注信息的帧间传递等。总之，要真正开发出一个效果良好，自动化程度较高的交互式 2D 到 3D 视频转换系统，还有大量的工作需要在现在的基础上完成。

致谢

首先诚挚地感谢我的指导老师王亦洲研究员。在完成毕业设计的过程中，老师常常与我讨论并指点我正确的方向，令我获益良多。老师对学术一丝不苟的态度也是我学习的榜样。

感谢博士生张哲斌师兄，他是项目的总体设计者，在项目和毕业论文的完成过程中，师兄都给予了我很大的帮助。感谢梁艳慧师姐，在我遇到问题时，不厌其烦地为我答疑解惑。感谢辛博同学，他是项目中“镜头分割与聚类”模块的完成者，在与他合作和讨论的过程中，我学到了很多东西。最后，感谢在大学生活中给予我关心和帮助的同学和老师们。

参考文献

[1] Kauff P., Muller M. et al. ICT – 215075 3D4YOU. Deliverable D2.1.2: Requirements on Post-production and Formats Conversion. Aug 2008.

[2] Carlos R. Ponce. et al. Stereopsis. Current Biology, 2008.

[3] Nedovic, V., Smeulders, A., Redert, A., Geusebroek, J.: Stages as models of scene geometry. In: PAMI (2010)

[4] Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis: OpenGL Programming Guide 6th edtion

[5] BOYKOV, Y., AND JOLLY, M.-P. 2001. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In ICCV, 105–112.

[6] C. Rother, V. Kolmogorov, and A. Blake. "grabcut" –interactive foreground extraction using iterated graph cuts, 2004. Siggraph.

[7] Richard O. Duda, Peter E. Hart, David G. Stork. Pattern Classification

[8] Christopher M. Bishop. Pattern Recognition and Machine Learning

[9] Justin F. Talbot, Xiaoqian Xu Implementing GrabCut. Brigham Young University,. Revised: April 7, 2006.

[10] ORCHARD, M. T., AND BOUMAN, C. A. 1991. Color Quantization of Images. IEEE Transactions on Signal Processing 39, 12, 2677–2690.

[11] MORTENSEN, E., AND BARRETT, W. 1995. Intelligent scissors for image composition. Proc. ACM Siggraph, 191–198.

[12] MORTENSEN, E., AND BARRETT, W. Interactive segmentation with intelligent scissors. Graphical Models and Image Processing, 1998 - Elsevier