ECE1001A 多媒体技术导论

课程论文&项目报告

学号：__2018633007___

姓名：___姜志成__ _

年级：大二

专业：电子与计算机工程

目录

多媒体通信技术综述

1. 多媒体通信技术综述

1.1 多媒体技术概述

媒体：指信息传递和存储的最基本的技术和手段，即信息的载体。其中媒体可以划分为**5 类**：感觉、表示、显示、存储、传输媒体。媒体具有**三个主要特点**，分别是：信息量巨大、数据类型的多样性和复合性、数据类型间的差别大

多媒体：通常是指感觉媒体的组合，即声音、文字、图像、数据等多种媒体的组合，融合了两种或者两种以上媒体的信息交流和传播

多媒体技术：对多媒体信息进行数字化采集、压缩/解压缩、编辑、存储等加工处理，再以单独或合成形式表现出来的一体化技术。最简单的表现形式就是多媒体计算机

多媒体通信：多媒体信息的获取、存储、处理、交换和传输。是多媒体信息处理技术和组网技术的融合。其中多媒体通信系统包含**四个特征**：集成性、交互性、同步性、实时性和等时性

1.2 数据压缩基本技术

1.2.1 数据压缩基本概念

进行数据压缩，主要的**原因**是多媒体数据量非常大，造成计算机的存储和网络传输负担。而之所以数据能被压缩，主要是因为它存在**冗余**，即数据的信息量小于数据量。常见的冗余包括：空间、时间、结构、知识、视听、信息熵冗余

数据压缩前期的**理论依据**是基于香农的**信息论**，后面主要是根据人眼识别图像的特征构造图像模型。在信息论中，是用信息量$I(x_i)$出来度量信息的。$I(x_i) = -log_2 p(x_i)$。$p(x_i)$ 是符号x_i出现的概率。但在大多时候是用**信息熵**来评价一个信源的，即$H(X) = E\{I(x_i)\} = -\sum_{i=1}^{N} p(x_i) log_2 p(x_i)$, X 为信源符号的概率分布越不对称，信源的熵越大，符号平坦分布时，信源的熵越小。

根据压缩数据还原后，是否与原数据一样可把数据压缩分为**无损压缩**和**有损压缩**。根据压缩原理可划分为：**统计编码**和**变换编码**。而在评价数据压缩的性能时，主要通过：压缩比、重现质量和压缩与解压缩的速度来评价。在编码方式中，无损压缩主要有**熵编码**和**词典编码**，有损压缩主要有 PCM 编码、预测编码、**变**

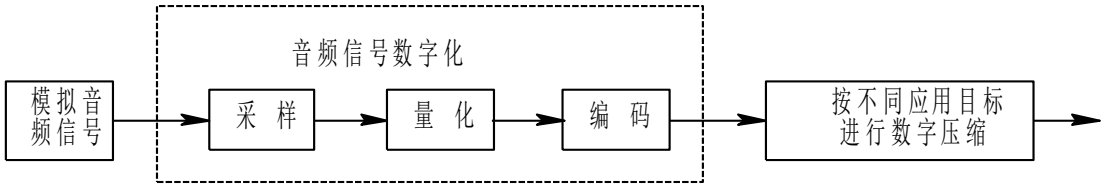
化编码、**子带编码**、混合编码。其中加黑的是生活中比较常用的编码。

霍夫曼编码原理：出现频率高的数据编码长度短。先把符号从出现概率大到小排序然后对出现概率最小的两个符号赋值，大的赋值为 1，小的赋值为 0，合并之后成为一个新符号加入到初始队列中，然后不断循环，直到最后两个符号的概率相加得 1。

算术编码原理：利用 0 和 1 之间的间隔来表示信源序列(也称为消息)的一种方法。把整个消息看作一个单元，用一个半开区间[a,b)来表示一条消息，其中 a 和 b 都是 0 和 1 之间的实数。初始时，区间是[0,1)。随着消息长度的增加，区间的长度将缩小，而用来表示区间的位数将增加。

1.2.2 音频压缩技术

首先，要介绍一个在音频压缩中比较重要的概念---**失真**：主要用来描述重现声音和原来声音的相差程度。在音频信息处理中，主要包含**音频信号的数字化**和**音频信息的压缩**两大技术。其中**音频信号数字化**是为了压缩做准备的。**数字化**其实就是将模拟音频信号转换成有限个数字表示的离散序列。其过程如图所示



音频信息压缩编码技术主要是向基于**波形**和基于**参数**两个方向发展的。从这个角度出发，音频信息编码技术可分为三类：波形编码，参数编码和混合编码。而在现代已经根据这三类编码指定了一些编码标准，如下图所示：

	算 法	名 称	码率/(kb/s)	标准	制定组织	制定时间	应用领域	质量
波形编码	PCM (A/μ)	压扩法	64	G.711	ITU	1972	PSTN ISDN	4.3
	ADPCM	自适应差值量化	32	G.721	ITU	1984		4.1
	SB-ADPCM	子带 ADPCM	64/56/48	G.722	ITU	1988		4.5
参数编码	LPC	线性预测编码	2.4		NSA	1982	保密语音	2.5
混合编码	CELPC	码激励 LPC	4.8		NSA	1989		3.2
	VSELPC	矢量和激励 LPC	8	GIA	CTIA	1989	移动通信 语音信箱	3.8
	RPE-LTP	长时预测规则码激励	13.2	GSM	GSM	1983		3.8
	LD-CELP	低延时码激励 LPC	16	G.728	ITU	1992	ISDN	4.1
	MPEG	多子带感知编码	128	MPEG	ISO	1992	CD	5.0

1.2.3 图像数据压缩编码技术

图像的分类：（1）按**存在形式**分类：实际图像，抽象图像（2）按**亮度等级**不同分类：二值图像，灰度图像（3）按图像**光谱特性**不同分类：黑白图像，彩色图像（4）按图像所**占空间维数**不同分类：二维图像，三维图像

图像信号数字化：将连续图像信号转化为**离散数字信号**。包括取样、量化和编码。**取样**：图像信号空间离散化的过程，即选取图像中的若干点。在过程中所选取的点就是取样点，也被称为**像素。量化**：将取样之后样本的灰度值转化为有限个离散值，并赋予不同码字，再交由数字设备处理运算。**编码**：按照一定的规律把量化后的值用二进制表示

图像压缩编码标准主要有：JPEG, JPEG 2000, 二值图像压缩标准等等

视频压缩编码标准主要有：H.261, H.263, MPEG-1, MPEG-2 和 MEPG-4 等等

1.3 多媒体通信及其相关技术

1.3.1 多媒体通信基础

多媒体通信的特征：集成性，交互性，同步性

多媒体通信框架主要包含：基于服务质量的多媒体通信系统框架，**基于 TCP/IP** 的多媒体通信模型。同时，我们会用 QoS 参数来描述系统行为的性能，主要包括：吞吐量，延时，延时抖动，错误率。一般实时性要求比较高的对 QoS 要求比较高，而 QoS 保证一般可以分为如下三个等级：确定型，统计型，**尽力而为型**。IP 网中最主要运用的是尽力而为型。

通信方式可分为三种：单工通信、半双工通信及全双工通信。**单工通信**是指消息只能单方向传输的工作方式（广播）。**半双工通信**可以实现双向的通信，但不能在两个方向上同时进行，必须轮流交替地进行（对讲机）。**全双工通信**是指通信的双方可以同时发送和接收数据（手机通话）

多媒体同步就是保持和维护各个媒体对象之间和各个媒体对象内部存在的时态关系。主要受六个因素影响：延时抖动，时钟偏差，不同的采样起始时间或不同的延时时间，不同的播放起始时间，数据丢失，网络传输条件的变换。

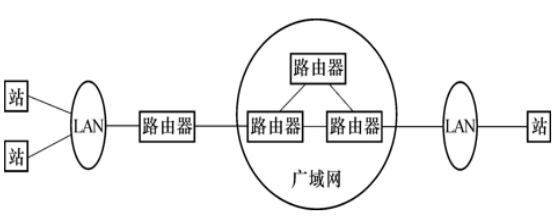
多播是指将信息同时传递给**一组目的地址**。

1.3.2 多媒体通信网络技术

多媒体传输网络主要分为：**电路交换网络**和**存储-转发交换网络**(优缺点)。目

前的通信网络大体上分为三类: 电信网络，如公用电话网（PSTN）、分组交换网（PSPDN）等；计算机网络，如局域网(LAN)、广域网(WAN）等；电视传播网络，如有线电视网（CATV）等。

Internet 网是路由器和专线构成的数据网， 它可以通过电话网、 分组网和局域网接入。其使用的通信协议为 TCP/IP。IP 网络的结构和 TCP/IP 的模型如下



IP网络的结构

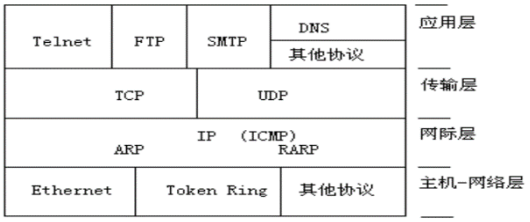


图 TCP/IP参考模型及协议

1.3.3 多媒体通信协议与系统

在这么多年的发展当中，逐渐制定了许多多媒体计算和通信系统的**推荐标准**，以促进各国之间的电信合作。其主要的系列标准如下：

	H. 320	H. 323 (V1/V2)	H. 324
发布时间	1990	1996/1998	1996
应用范围	窄带ISDN	带宽无保证信息包交换网络	PSTN
图像编码	H. 261，H. 263	H. 261，H. 263	H. 261，H. 263
声音编码	G. 711，G. 722，G. 728	G. 711，G. 722，G. 728 G. 723. 1，G. 729	G. 723. 1
多路复合控制	H. 221，H. 230/H. 242	H. 225. 0，H. 245	H. 223，H. 245
多点	H. 231，H. 243	H. 323	
数据	T. 120	T. 120	T. 120

多媒体通信系统是指能够完成多媒体通信业务的系统，包括多媒体通信终端、通信设备、传输通路、多媒体应用服务设备等，它们由通信网络连接在一起

1.3.4 多媒体通信的应用与发展趋势

应用主要包括：可视电话系统、视频会议系统、多媒体电子邮件、视频点播系统 VOD、远程教育系统、虚拟现实

未来发展的趋势主要有：（1）多网合一（2）图像信息处理方面，出现更多新一代的图像压缩编码算法（3）多媒体通信终端体积越来越小，性能越来越强

项目计划和软件介绍书

2. 项目计划和软件介绍书

2.1 项目背景

在人工智能越来越火的时代，很多人都只是听过或者是些许的见过人工智能的应用，但并没有这个能力亲自去体验人工智能。此次项目意在让更多对计算机没有基础但是对人工智能，图像处理很感兴趣的人提供一种能亲身体验并且具有一定交互性的图片处理的网站。

同时，该项目还可以结合**远程教育**，来实现远程的仿真。由于很多同学在家的电脑的配置有限，如果选择了机器学习或者是深度学习之类的课程，可能会因为电脑配置而不能展开仿真实验。而如果把运算平台架设在云端，同时通过网页这样交互式的平台，可以通过不断地修改参数，改变实验结果，从而达到更好的实验效果。

2.2 项目计划

此次的智能图片处理系统大致分为五个部分：

（1） 用网页的形式来实现一个无门槛的操作界面

（2） 实现基础的图片处理功能比如调节亮度，对比度，饱和度，阴影。

（3） 实现各类基础滤波算法，让零基础的人感受到降噪算法的强大之处

（4） 利用 seam carving 算法实现图像压缩，让人们感受到生活钟处处都有的图像压缩

2.3 软件的搜索、比较和选定

2.3.1 开发语言的选择

由于我要使用 OpenCV 库，而 OpenCV 主流支持的语言有 C++和 Python。

实际上 Python 和 C++的 OpenCV 接口几乎一样，PyOpenCV 只是一层封装而已。不同的是用 C++的话，矩阵用的是 cv::Mat，Python 里面用的是 numpy.array，用法和接口都不一样，但文档丰富，使用也非常方便，原理也相似。

效率方面，Python 的接口实际上只是一层 Binding，最终还是调用 libopencv_*.so 里面的函数，所以在 OpenCV 这一层效率与 C++是完全一致的。唯一不同的就是它的 numpy.array 和 cv::Mat。Numpy 底层也是使用 C Extension

的方法写，但相比 C++版的 OpenCV 接口，Python 的接口需要把 Numpy 的数据转化成 OpenCV 的 C 接口可接受的输入。所以 Python 写的程序会慢，但具体慢多少并没有测试数据可支撑。

所以总的来说，因为自己对 Python 比较熟悉和 C++比较难入门。此次项目又主要是想做快速成型验证想法之类的。同时 Python 有着写起来省时省力的优势，毕竟动态类型提供的方便足够多。所以最后还是选择了 Python。

同时，由于我需要制作一个网页来形成交互式的操作界面，选择 Python 毫无疑问是更好的选择。因为 Python 有许多框架可以使用，包括 Flask，Django。而选择 C++的话，一个是开发的时候不方便，其次就是现成的库和框架比较少，所以不选择 C++而选择 Python 进行开发

2.3.2 搭建系统的选择

由于要构建自己的通信框架，利用常见的通信协议来进行网络编程。首先我第一个要选择的就是开发的系统，现在市面上比较流行的系统就是 Linux 和 Windows。下图是两个系统的不同之处

	ubuntu	windows
优点	1、高效的文件管理系统，一般情况下不需要碎片整理 。 2、产生的系统垃圾很少，系统不会随着使用时间的增多而越来越卡 。 3、拷贝文件速度快，Win10达到5M/S，Ubuntu达到20M/S 。 4、强大的命令行，基本上所有操作可在上面执行	1. 软件多，软件多，软件多(重要的事情说三遍) 。 2、支持大量驱动，充分发挥硬件性能。
缺点	1.硬件驱动支持不友好，例如独立显卡，并不能发挥性能优势 2.相比Windows，软件数量少(QQ居然没有Linux版，还要装个CrossOver) 3.学习需要一定的时间成本，长期使用才能较好地驾驭系统	1.执行exe安装包，快速地点next，最后发现PC多了很多不想要的软件 2.使用越久，C盘越大，越来越卡(碎片整理，你值得拥有) 3.莫名奇妙蓝屏，死机奔溃的几率大(特别是更新系统和驱动后)，因为windows是闭源的，貌似重装才是解决问题的唯一办法 4.PC空闲时，系统后台任务促使磁盘利用率莫名奇妙90%

图 2.1 Windows 与 Ubuntu 的对比图

相比于 Windows 我最后还是选择了 Ubuntu。主要有以下几个原因：

1. 如果以后要对这个项目继续扩展的话，就必须在云端搭建一个服务器，而维护服务器的话，Linux 相比于 Windows 会更好。因为（1）**开源**：Linux Server 相较 Windows Server 领先的首要原因是完全免费且可用作开源用途。通过开源方式，您可以轻松查看用于创建 Linux 内核的代码，也可以对代码进行修改和再创作。通过许多编程接口，您甚至可以开发自己的程序并将其添加到 Linux 操作系统中。还可以对 Linux Server 操作系统进行自定义，以满足使用要求，这是 Windows 无法实现的。（2）**稳定性**：Linux 系统一直以其稳定性而闻名，它们可以连续运行多年而不发生任何重大问题。事实上，很多 Linux 用户都从未在自己的环境中遇到过系统崩溃的情况。相对 Windows 而言，挂起和崩溃完全是一种常态。尽管 Windows 也可以很好地执行多任务处理，但 Linux 可以在处理各种任务的同时，仍能提供坚如磐石的性能。当将 Linux 与 Windows 进行对比时，对每项系统配置的更改都需要重启 Windows Server，而 Linux 更改大多数配置时都无需重启服务器即可生效，这也确保了 Linux 服务器最短的停机时间（3）**安全**：Linux 由最初的多用户操作系统开发的 UNIX 操作系统发展而来，在安全方面显然比 Windows 更强。与 Windows 作为病毒和恶意软件攻击的首要目标不同，Linux Server 只有管理员或特定用户才有权访问 Linux 内核，而且 Linux 服务器（相较 Windows 而言）不会经常受到攻击，并且被发现的任何漏洞都会在第一时间由大批 Linux 开发人员修复。
2. 用 Ubuntu 在安装模块的时候直接在命令行用 pip 安装就可以了，Windows 的话还要考虑路径问题。

2.3.3 框架的选择（Flask 与 Django 比较）

- 基本介绍与特色

  Flask 框架：Flask 是一个由 Python 语言写成的轻量级 Web 框架，最早由奥地利人 Armin Ronacher 于 2010 年发布。Flask 最显著的特点是它是一个“微”框架，轻便灵活，但同时又易于扩展。默认情况下，Flask 只相

当于一个内核，不包含数据库抽象层(ORM)、用户认证、表单验证、发送邮件等其它 Web 框架经常包含的功能。Flask 依赖用各种灵活的扩展（比如邮件 Flask-Mail，用户认证 Flask-Login，数据库 Flask-SQLAlchemy）来给 Web 应用添加额外功能。Flask 的这种按需扩展的灵活性是很多程序员喜欢它的地方。Flask 没有指定的数据库，可以用 MySQL，也可以用 NoSQL。

Django 框架：Django 是一个开源的 Python Web 应用框架，采用了 MVT 的框架模式，即模型 M，视图 V 和模版 T，最早于 2005 年发布。Django 被认为是"大而全"的重量级 Web 框架，其自带大量的常用工具和组件（比如数据库 ORM 组件、用户认证、权限管理、分页、缓存), 甚至还自带了管理后台 Admin，适合快速开发功能完善的企业级网站。Django 自带免费的数据 SQLite，同时支持 MySQL 与 PostgreSQL 等多种数据库。

- 流行度

Flask 和 Django 均是当今最流行的 Python Web 框架。截至 2019 年 9 月 2 日，Flask 在 Github 上的星数是 46179 颗，Django 的 Github 星数是 43806 颗，两者几乎难分伯仲，其它 Python Web 框架与 Flask 和 Django 星数相差甚远。考虑到 Django 早发布 5 年，而 Flask 在星数上还领先 2000 多颗，由此可以得知 Flask 当前略微占优。

从应用上来说，Flask 与 Django 均广泛用于 Web 应用开发。利用 Django 开发的著名网站包括 Pinterest, Disqus, Eventbrite, Instagram and Bitbucket。不过最近 Pinterest 改用 Flask 开发它的 API 了。其它利用 Flask 开发的项目包括 Twilio, Netflix, Uber 和 LinkedIn。Django 似乎更多用来开发常规网站，而 Flask 经常用来开发 API（比如 Pinterest 和 Twilio)。（注：这点可以理解。如果只需要开发 Web API，Django 自带的大而全的网站功能很多用不上，比如缓存和管理后台。)

- 性能

两者差不多。有些人会说 Django 会因"大而全"更慢些，其实这是 Python 的锅。网上有人测试对比过 Flask 和 Django 的性能，两者速度差不多，并无本质差异。

- 项目结构

Django 项目的结构布局是刚性的，每个人写的项目结构最后都差不多，你可以清楚地知道在哪个 APP 的哪个文件夹里找到哪个文件(media 目录, static 目录, template 目录，views.py, models.py, forms.py, etc)。在项目结构上，Flask 是很灵活的，你可以随意地组织自己的代码，1000 个 APP 说不定就有有 1000 种组织代码的方式。不同的人之间因为习惯不同可能导致最后项目结构布局差异很大, 造成后期代码难以阅读和维护。如果大家都严格遵循 Flask 推荐的代码布局，那么你会发现最后将得到和 Django 类似的项目结构布局。

- 文档与社区

Django 的文档要更丰富些，目前总的社区人数要更大点。StackOverflow 上 Django 的问题与回答数大概是 Flask 的 3 倍以上。不过 Flask 社区目前也在成长，且速度惊人。

但综合考虑，我还是选择 Flask 框架来开发，因为我只需要开发一个轻量级网站，并用不上 Django 自带的大而全的组件和功能。同时 Flask 更加自由灵活的特点也让我这个小项目更加容易展开。

2.4 项目亮点

此次项目的亮点主要有以下几个：

零门槛，高交互性：由于是用网页来实现的一个交互系统，所以可以把服务器架设到云端之后，使得即使是零基础的人都可以直接通过运行之后的图片来得到视觉的反馈，从而激起对学习的兴趣

能实现远程教育的目的：由于在以后的改进中，GPU 还有服务器都是架设在云端上。所以即使是在家，老师都可以叫学生直接用自己电脑，通过网页来达到

辅助教学的目的

准确性：相比于传统的图像压缩算法，这里使用 Seam Carving 算法，使得图像中的关键目标、感兴趣目标不失真

计划性强：运用了大一《工程设计导论》中学习到的甘特图。通过甘特图这种更易于时间管理的工具，来进行项目的规划及管理

2.5 计划进度表

A	B	C
计划工作情况	开始日期	持续天数
项目寻找灵感	5月2日	3
综述1st（多媒体的概念）	5月2日	3
综述2nd（压缩技术）	5月5日	5
学习图像处理基本概念	5月5日	8
学习python的flask框架	5月16日	5
综述3rd（通信框架）	5月16日	4
综述4th（感想）	5月20日	2
编写整个项目的代码	5月23日	6
完成项目的软件介绍书，可行性分析	5月29日	2
进行论文最后的整理	6月5日	3

图 2.2 计划工作图

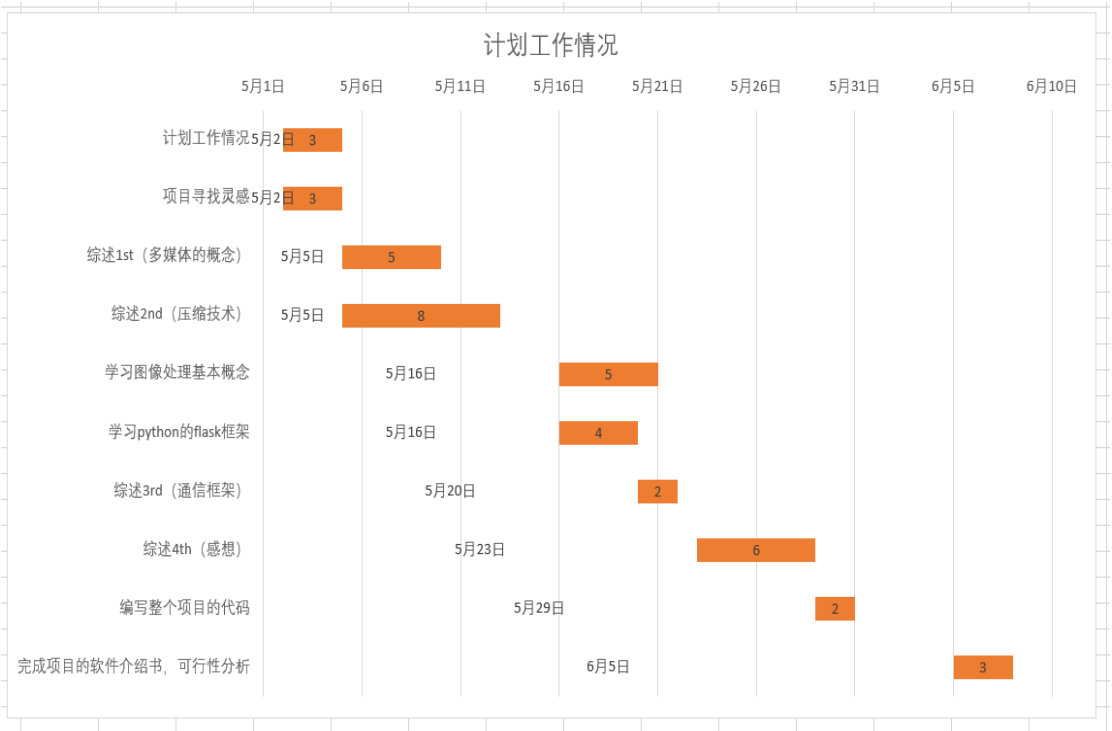


图 2.3 计划工作甘特图

实验可行性报告书

3. 实验可行性报告书

3.1 通信系统构建框架及说明

3.1.1 通信系统框架介绍

由于 Internet 同时具有双向、单向、实时、非实时的交互功能，支持文本、图形、音频、视频等多种文件类型的传输，能独立构成远程教学的完整信息回路，因此，被广泛采用，成为远程教育最基本的网络平台。然而，受带宽因素影响，Internet 对实时音频、视频信息传输效果不够理想，无法满足远程教育的要求。

因此，在大多数远程教育系统网络平台中都选择了一种或几种适于高速度、高质量、 高稳定性音频、 视频信息传输的网络平台，来弥补课堂视频教学方面的不足。这些平台主要有：基于卫星的单向视频广播；基于有线电视的单向视频广播；基于 ISDN 的双向视频会议；基于 DDN 的双向视频会议；基于卫星的双向视频会议等等。

而此次，我搭建的智能图片处理系统就是一种用于远程教育的一种**辅助**，他是建立在基于 ISDN 的双向视频会议的基础之上的。搭建图如下所示

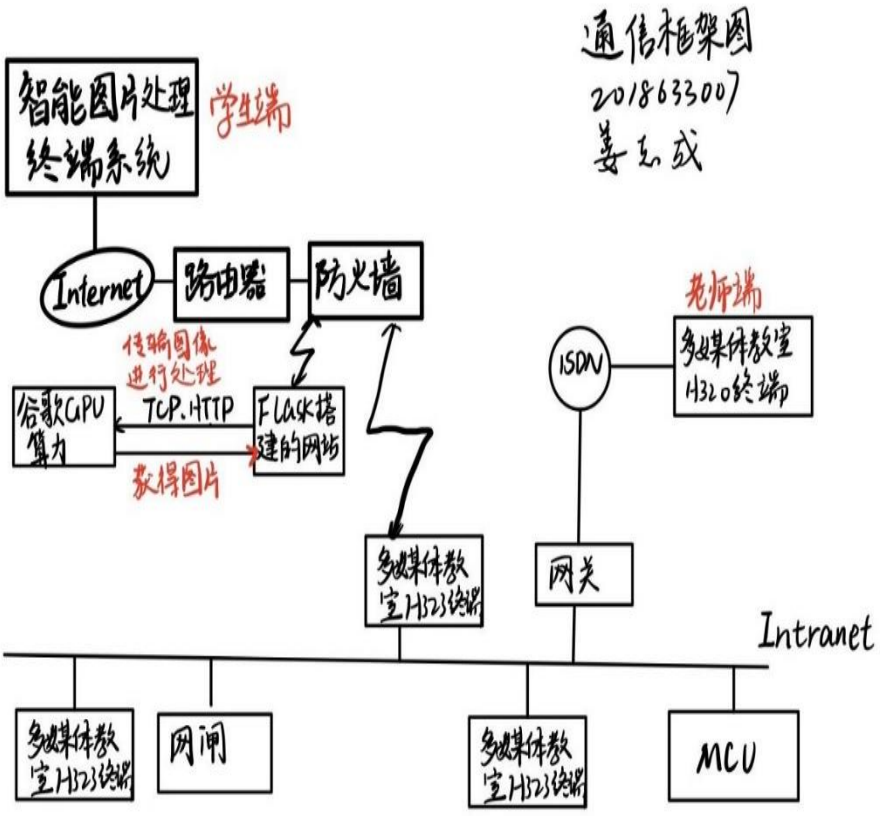


图 3.1 通信框架图

3.1.2 通信系统框架说明

基于 ISDN 网络的视频会议系统用于远程教育的最大特点是双向实时交互功能，主要用于建立主课堂与校外教学中心分教室之间的多媒体实时授听课系统。那么在我加上的智能图片处理系统之后，老师在指导进行仿真实验的时候可以更加好的展开。

在框架中，主要分成三个部分：

1. 学生端

（1） 学生端主要是通过 Internet 来访问我用 flask 搭建的网站来进行交互式的实验

（2） 通过 Internet 加入多媒体教室进行在线教育，老师在线指导进行实

验的进行

2. 云端提供的免费算力

   学生端通过对云端（例如谷歌云，亚马逊 AWS 等算力平台）发出 TCP，HTTP 请求，利用云端提供的 GPU 进行计算，然后云端再把运算后的结果返回到学生端，让学生端看到实验的结果

3. 老师端

   老师通过基于 ISDN 网络的视频会议系统来上课，在线指导同学们进行仿真实验

3.2 媒体压缩技术应用及测试

3.2.1 Seam carving 算法介绍

传统的图像压缩算法在 OpenCV 中主要是通过 resize()方法来实现的。其压缩原理主要是基于采样和插值，但是通过这种方法图像内容会产生拉伸或压缩，比如图像中有人脸等内容，通常产生糟糕的效果。Seam carving 的基本思想是：图像中不同的位置“能量”不同，有重要内容的地方能量大，反之，可有可无的位置能量就小。这个能量函数可以是梯度。这样假设的原因主要是：梯度大的地方亮度变化剧烈、纹理丰富，一般是重要内容；梯度小的位置亮度变化小，比如蓝天，删除一条缝隙对图像影响较小。考虑横向或者纵向寻找一条接缝，怎么样寻找这样一条能量最小的接缝？可以通过动态规划（Dynamic Programing）的方法实现

3.2.2 Seam Carving 算法原理

Seam Carving 是通过对原图删除（增加）像素的方式来 resize 图像的。假设输入图片为I是$n \times m$的图像。定义能量方程为X方向梯度幅值加上Y方向梯度幅值（论文中也给出了其他的能量方程定义方式，但实验表明结果差不多）：

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}| \qquad (1)$$

图 3.2 能量方程定义

一条**垂直 seam** 定义为：

$$\mathbf{s}^{\mathbf{x}} = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1, \quad (2)$$

图 3.3 垂直 seam 定义

其中x是映射 $x : [1, \ldots, n] \rightarrow [1, \ldots, m]$ 。也就是说，**垂直 seam** 是图像中像素从上到下的 8 连通路径，在图像的每一行中仅包含一个像素。

类似的可以定义**水平 seam**：

$$\mathbf{s}^{\mathbf{y}} = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, \text{ s.t. } \forall j |y(j) - y(j-1)| \leq 1. \quad (3)$$

图 3.4 水平 seam 定义

注意(2),(3)式中的小于等于 1 可以改为小于等于 k，根据实际情况设定 k 值。

一条 seam 的能量定义为：

$$E(\mathbf{s}) = E(\mathbf{I}_{\mathbf{s}}) = \sum_{i=1}^n e(\mathbf{I}(s_i))$$

图 3.5 一条 seam 的能量定义

图片当前状态最优 seam 定义为s^*,s^*为能量最小的 seam：

$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^n e(\mathbf{I}(s_i)) \quad (4)$$

图 3.6 能量最小的 seam 定义

Seam Carving 算法的思想：

1. **为每个像素分配一个能量值**
2. **找到能量值最小的像素的八连通路径(最优 seam)**
3. **删除或复制路径中的所有像素**
4. **重复前面 1-3 步，直到删除或复制的行/列数量达到理想状态**

最优 seam 可以通过**动态规划（Dynamic Programming）**的方式寻找，从第二行（列）开始，计算每个像素(i,j)到第一行（列）的最优路径：

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

图 3.7 动态规划方程定义

依次计算到最后一行（列）就可得到最优 seam。

在缩小图片时，每次删除当前转态图片的最优 seam 即可。

在放大图片时，我们对最优 seam 以及其相邻像素计算平均值，然后将这个平均值复制到最优 seam 旁来增加一行（列）。以此来放大图像。同时为了防止每次都复制同一个 seam（每次都是那个 seam 最优），可以先将原图中最优的 K 个 seam 找出来，然后一次性复制这 K 个 seam，从而增加 K 行（列）。

3.2.3 代码实现

```
1. # -*- coding: utf-8 -*-
2. from __future__ import unicode_literals
3. from imgProcessConfig import *
4. from imgProcessTools import *
```

```
5.  from imgProcessTools import (Axes3D, FormatStrFormatter, Fun, HIS_BACKGROUND_COLOR, HIS_MAX, LinearLocator, MAX_FILTER_R, SHOW_SHAPE, View, allFilter, applyColorTable, arange, array, avgFilter, base64, base64Img, bg, bilateralFilter, blackToWhiteVidoEffects, c, cProfile, cm, crun, cv2, da, draw3dSurface, filterr, g, gaussCore, getColorTables, getFilterImg, getLine, gussFilter, ind, interpolate, io, log, loga, mapp, math, maxFilter, mdeianFilter, normalizing, np, os, plt, polt3dSurface, py3, pyv, r, random, roundInt, run, saveAvi, scipy, show, sk, skimage, smallImg, stand, step, sys, tan, unicode_literals)
6.  from skimage import filters
7.  from skimage import transform
8.  import numpy as np
9.  cvSobel = filters.sobel
10. 
11. r = random(3,5)
12. 
13. #show([img,mag])
14. def getSobel(rgb,mask=None):
15.     img = rgb.astype(int)
16.     tmp = np.mean(np.abs(img[:-1,:-1]-img[1:,:-1])+np.abs(img[:-1,:-1]-img[:-1,1:]),2)
17.     mag = np.append(tmp,tmp[-1:],0)
18.     mag = np.append(mag,mag[:,-1:],1)
19.     if mask is not None:
20.         mag+= mask
21.     return mag.astype(int)
22. #mag = mySobel(img)
23. #show([mag,magg])
24. 
25. def reduceOneLine(img,mag,mask=None):
26.     m, n = mag.shape
27.     expand = lambda row: np.append(np.append(row[:1],row),row[-1:])
28.     accu = [mag[0]]
```

```
29.     for ind in range(1,m):
30.         tmp = np.zeros((n,3),mag.dtype)
31.         row = expand(accu[ind-1])
32.         tmp[...,0] = row[:-2]
33.         tmp[...,1] = row[1:-1]
34.         tmp[...,2] = row[2:]
35.         add = np.min(tmp,1)
36.         accu += [add+mag[ind]]
37.
38.     accu = np.array(accu)
39.     accui = np.append([[accu.max()+1]]*m,accu,1)
40.
41.     _ind = np.argmin(accui[-1])
42.     ind = _ind - 1
43.     pop = lambda ind,row: np.append(row[:ind],row[ind+1:],0).astype(row.dtype)
44.     newimg = [pop(ind,img[-1])]
45.     if mask is not None:
46.         newmask = [pop(ind,mask[-1])]
47.     for r in range(m-1)[::-1]:
48.         d = np.argmin(accui[r][ind:ind+3])-1
49.         ind = ind+d
50.         newimg = [pop(ind,img[r])]+newimg
51.         if mask is not None:
52.             newmask = [pop(ind,mask[r])]+newmask
53.     newimg = np.array(newimg)
54.     if mask is not None:
55. #       print newmask
56.         return newimg.astype(img.dtype),np.array(newmask)
57.     return newimg.astype(img.dtype),None
58. def seamCarve(img,deln,mask=None,each=False,f=None,row=None):
```

```
    if mask is not None:
        if isinstance(mask[0][0],float):
            mask -= skimage.filters.threshold_li(mask)
            mask *= 512
            mask.astype(int)
        else:
            mask[mask==1]=512
            mask[mask==-1]=-512
    if row:
        img=np.transpose(img,[1,0,2])
        mask = None if mask is None else mask.T
    newimg = img
    for i in range(deln):
        mag = getSobel(newimg,mask)
        newimg,mask = reduceOneLine(newimg,mag,mask)
        if f and i % each==0:
            f(newimg,i+1,row)
    if row:
        newimg=np.transpose(newimg,[1,0,2])
    return newimg


def reduceOneLineWithHed(img,mag,mask=None):
    m, n = mag.shape
    expand = lambda row: np.append(np.append(row[:1],row),row[-1:])
    accu = [mag[0]]
    for ind in range(1,m):
        tmp = np.zeros((n,3),mag.dtype)
        row = expand(accu[ind-1])
        tmp[...,0] = row[:-2]
```

```
89.         tmp[...,1] = row[1:-1]
90.         tmp[...,2] = row[2:]
91.         add = np.min(tmp,1)
92.         accu += [add+mag[ind]]
93.
94.     accu = np.array(accu)
95.     accui = np.append([[accu.max()+1]]*m,accu,1)
96.
97.     _ind = np.argmin(accui[-1])
98.     ind = _ind - 1
99.     pop = lambda ind,row: np.append(row[:ind],row[ind+1:],0).astype(row.dtype)
100.    newimg = [pop(ind,img[-1])]
101.    hed = mag
102.    hed = [pop(ind,mag[-1])]
103.    if mask is not None:
104.        newmask = [pop(ind,mask[-1])]
105.    for r in range(m-1)[::-1]:
106.        d = np.argmin(accui[r][ind:ind+3])-1
107.        ind = ind+d
108.        newimg = [pop(ind,img[r])]+newimg
109.        hed = [pop(ind,mag[r])]+hed
110.        if mask is not None:
111.            newmask = [pop(ind,mask[r])]+newmask
112.    newimg = np.array(newimg)
113.    hed = np.array(hed)
114.    if mask is not None:
115. #        print newmask
116.        return newimg.astype(img.dtype),hed,np.array(newmask)
117.    return newimg.astype(img.dtype),hed,None
118.
```

```
119. def seamCarveWithHed(img,deln,mask=None,each=False,f=None,row=None,hed=None):
120.     if mask is not None:
121.         if isinstance(mask[0][0],float):
122.             mask -= skimage.filters.threshold_li(mask)
123.             mask *= 512
124.             mask.astype(int)
125.         else:
126.             mask[mask==1]=512
127.             mask[mask==-1]=-512
128.     if row:
129.         img=np.transpose(img,[1,0,2])
130.         mask = None if mask is None else mask.T
131.     newimg = img
132.     mag = hed.T if row else hed
133.     for i in range(deln):
134.         newimg,mag,mask = reduceOneLineWithHed(newimg,mag,mask)
135.         if f and i % each==0:
136.             f(newimg,i+1,row)
137.     if row:
138.         newimg=np.transpose(newimg,[1,0,2])
139.     return newimg
140.
141. def addBlack(img,i,row):
142.     new = np.append(img,[[[0,0,0]]*i]*img.shape[0],1).astype(img.dtype)
143. #     new = np.append(img,[[[0,0,0] for _ in range(i)] for __ in range(img.shape[0])],1)
144.     new = np.transpose(new,[1,0,2])  if row else new
145.     return new
146. scShow = lambda img,i,row: show(addBlack(img,i,row))
147.
148. def seamCarveBlack(img,deln,mask=None,each=False,f=None,row=None,):
```

```
149.     imgNoBlack = seamCarve(img,deln,mask=mask,each=each,f=f,row=row,)
150.     imgNoBlack = np.transpose(imgNoBlack,[1,0,2]) if row else imgNoBlack
151.     newimg = addBlack(imgNoBlack,deln,row)
152.     return newimg
153. 
154. 
155. 
156. #sc = transform.seam_carve(rgb,mag,'vertical',deln)
157. #show(sc)
158. #%%
159. if __name__ == '__main__':
160.     from imgProcess import *
161.     rgb = da.astronaut()
162.     img = rgb.copy()
163.     img = g['img']
164.     sal = g['sal']
165.     deln = 100
166.     mask = np.zeros(img.shape[:2],int)
167. #    mask[:150] = -1
168. #    mask = None
169.     mask  = sal
170.     row = True
171.     newimg = seamCarveBlack(img,deln,mask,deln//3,scShow,row)
172.     hed = g['hed']
173. #    newimg2 = seamCarveWithHed(img,deln,mask,deln//3,scShow,row,hed=hed)
174.     show(newimg)
175. #    show(newimg2)
176.     show([hed,getSobel(img)])
177.     pass
178. 
```

3.2.4 Seam Carving 测试

图 3.8 原始图

图 3.9 3.10 左侧为使用 resize，右侧是 crop 剪切

图 3.11 使用 seam carving 算法

在这个图片中，感兴趣的目标为图像中间的圣诞老人和桌子。但从上面的各图中可以看到，图 3.9 在 Opencv 提供的 resize 方法压缩之后，那个圣诞老人和桌子明显进行极大的压缩，显得非常的不自然，效果非常差。但是使用了 Seam Carving 算法后（图 3.11）不仅能达到剪切（图 3.10）的效果，即使得感兴趣的目标最大程度的保存下来，同时还达到了图片压缩的效果。

由此可见，使用 Seam Carving 算法进行图像压缩比一般的图像压缩效果好很多

3.3 仿真测试可行性分析

3.3.1 用网页的形式来实现一个无门槛的操作界面

调用模块：flask, flask_socketio, json, time, os, base64

实现功能：通过 flask 框架搭建一个网页后端，通过网络编程来实现图片上传调用，返回图片的功能。同时，通过改变 ip 和端口可以把运算功能搭建在云端算力上面

分析：

Flask, flask_socket: 现在 python 比较流行的网页后端框架，利用这个框架可

以直接搭建起一个网站，然后直接通过 setting.py 文件进行 admin 管理。允许使用者上传图片，这样可以直接在本地端上传图片，来直接进行仿真

Base64: 直接把图像的内容嵌入网页里面，减少页面的请求。

Time: 在 http 和 tcp 请求的时候，要进行等待，如果超时的话要进行重传。同时，如果使用者觉得很久都没有出来结果，可以通过这个模块来进行重新上传。

Json：进行轻量级的数据交换，把一些 css 和 html 元素传递到网页上

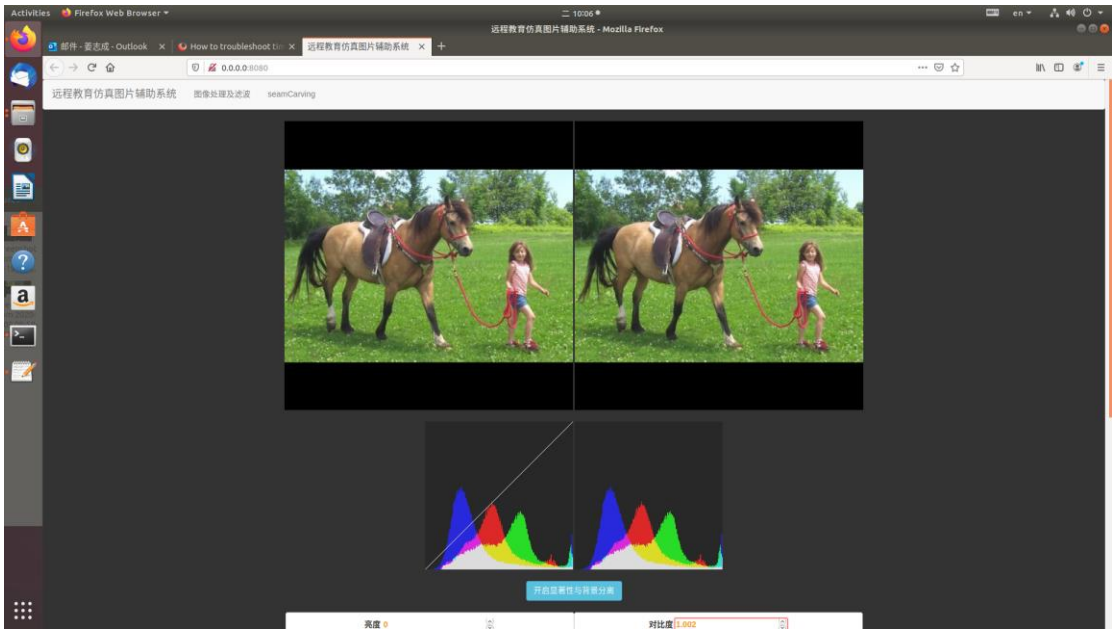


图 3.12 网站主页

3.3.2 实现基础的图片处理功能比如调节亮度，对比度，饱和度，阴影

调用模块：Opencv

实现功能：通过调用 opencv 里面的基本定义，即像素的定义，然后通过改变数值来达到改变调节亮度，对比度，饱和度，阴影等功能

分析：

1. 亮度与对比度

   在图像像素公式 g(x)=a*f(x)+b 其中定义：

   参数 f(x)表示源图像像素。

   参数 g(x) 表示输出图像像素。

   参数 a（需要满足 a>0）被称为增益（gain），常常被用来控制图像的对比度。

参数 b 通常被称为偏置（bias），常常被用来控制图像的亮度。

为了访问图像的每一个像素，我们使用这样的语法： image.at<Vec3b>(y,x)[c] 这里的 a 也就是对比度，通过改变 a 和 b 的值可以改变图像的亮度和对比度

2. 改变饱和度，阴影

OpenCV 用 cvtColor 函数把 BGR 图像转换成另一种色彩空间，用色调、饱和度和亮度表示颜色，这里使用转换代码 CV_BGR2HSV

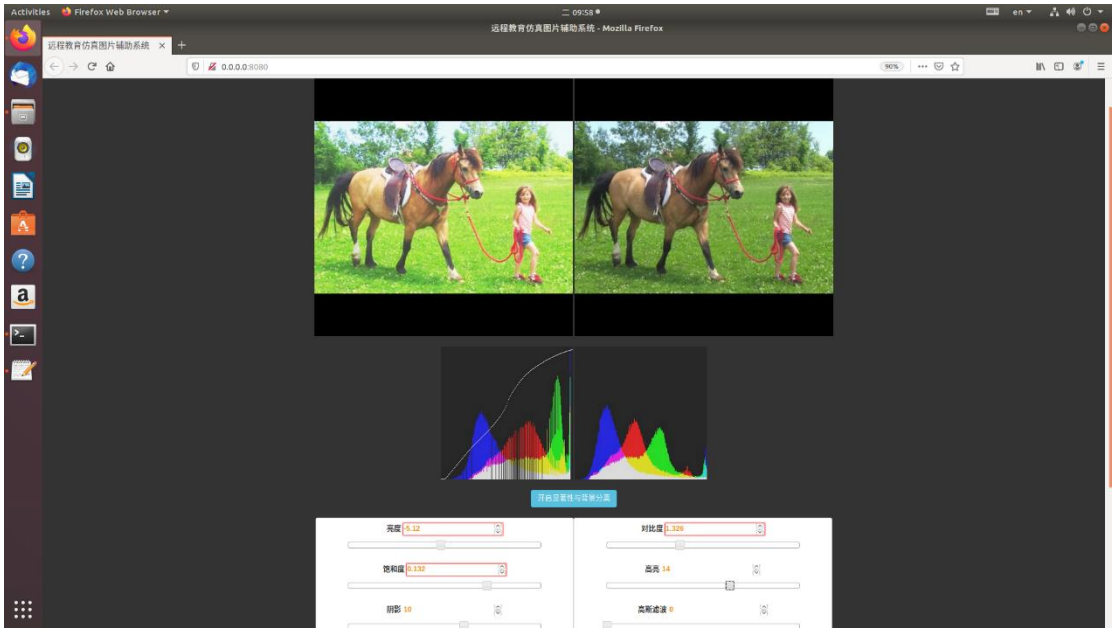


图 3.13 调节了饱和度亮度和对比度

3.3.3 实现各类基础滤波算法

调用模块：Opencv

分析：这一部分我没有自己实现，主要是通过调用 opencv 里面的函数，例如 blur(), GaussianBlur(), medianBlur(), bilateralFilter()等滤波函数。

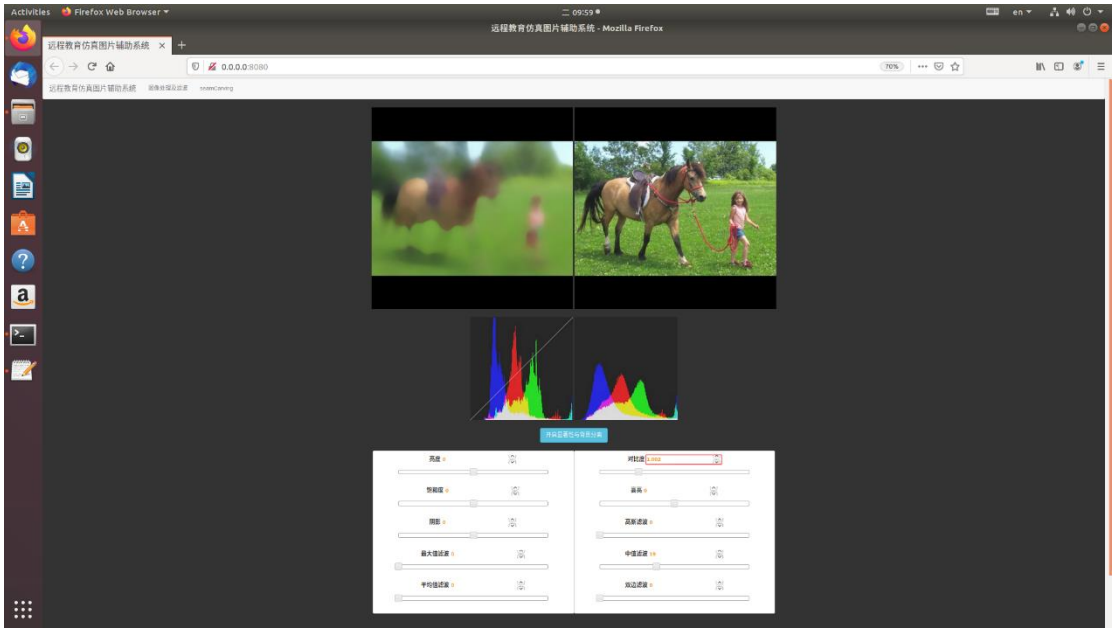


图 3.14 以中值滤波为例

3.3.4 利用 seam carving 算法实现图像压缩

在 3.2 中有进行详细的算法介绍和分析

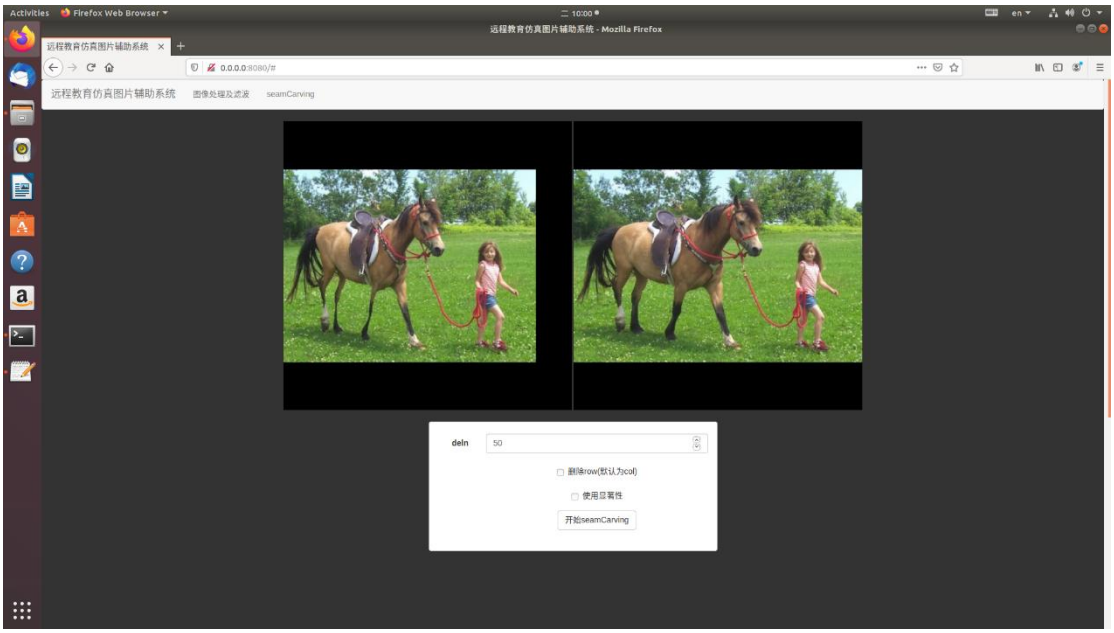


图 3.15 使用了 Seam carving 算法进行图片压缩

3.4 实践收获和感想

虽然这门课是一门导论课，但是老师也在这门课中为我们普及了许多有关多媒体，压缩技术，通信协议及系统的知识。一开头看这些知识还不算特别陌生，

因为在上这门课之前我已经学过了计算机网络和图像处理，所以对很多知识还是有印象的。但是老师却从另外一个角度来对这些知识点进行阐述，同时也介绍了很多这方面的应用，之前我从来没有关心过这方面的知识。老师也让我们做了一次 pre，在 pre 中，我介绍了 IPTV。在进行搜索资料之前，其实我对电视的印象还是停留在以前的天线接收信号，但是现在的电视基本上都是三网合一，不仅能提供电视服务，还会提供网络服务。而且在之前的电视中只有直播，如果错过了就很难追回来，但是现在由于技术的提升，还提供了点播回放等功能。所以在平时的时候还是要去多关注生活当中的各种技术，多去思考现在的技术和以前的技术的进步体现在哪里，是什么原因导致技术进步的。

在自己写这个项目中也收获了很多。主要有两点：

1. 通过详细的目标和时间计划表，能完整地完成一个项目。在大二之前的学习中，主要是学一些零零碎碎的东西，很难有这样一个机会把知识整合起来。比如把改变亮度、饱和度等图片属性，通过 json 控件的形式来调控，通过 flask 搭建一个简单的网页后端实现图片的上传和下载。而且通过实现指定的时间计划表能更有效率地展开工作。当然，在制定时间表的时候，一定要把任务划分好区域，比如把写综述分开任务的话，就不要说分配几天去写综述，而是说写综述的第几个部分用几天。在以后的话，也要尝试去把不同的学科融合起来，把不同学科的知识融会贯通。现在学科分得比较细，所以学的东西比较分散，只有把各科的知识融会贯通起来，才能把每一个学科的知识学的更加深入。
2. 成功地把理论投入了实践中。在当初学习计算机网络和图像处理的时候，只知道事物的原理，而不知道怎么把它通过代码的形式展示出来。就比如说通信协议的实现，图像压缩算法的实现，像素点的定义等等。尽管自己在知识层面学的还不错，但只有投入没有产出的话，其实还是没有真正地学会这些知识。尤其是我们电子和计算机专业的学生，必须学会把自己学会的知识通过代码，图片等形式展示出来，同时与别人分享交流，才能不断地提升自己的能力，才能适应互联网社会的发展。

当然，在这次的项目中也有很多不足：

1. 在指定计划的时候，只考虑到了计划成功的时间，而不去考虑如果没做成这

个任务应该怎么办。这样会导致当天的任务没做完，但是明天又有新的任务，就不知道怎么处理。所以在安排任务的时候，不仅要安排成功完成任务的时间，还要给到任务一个小的期限，不能把任务规划的太死，否则在后面展开工作的时候会非常沮丧和懊悔，每次都在想自己为什么没有完成之前的任务，而没有把精力投入到今天的任务当中。

2. 过多地依赖于第三方库。由于时间的原因，同时也受限于自己的能力。很多东西我都是直接调用现成的库，现成的框架。这样对于一个计算机和电子方面的学生不是太好的。诚然，有些轮子不需要我们重新制作，但是一些非常重要的内容还是需要自己慢慢地实现，不能一味地依赖现成的库。否则在以后的工作和学习当中，很容易被淘汰。因为我们只是比较熟悉怎么使用这些库这些框架，但是对这里面的原理却一概不知。所以在以后的学习当中，调用现成的库的时候，可以多去看一下官方源码，或者是网络上面博客的解析，让自己对这些库有着更深的理解。这样做，不仅能让我们更加熟练地掌握这些库的使用，当出现错误的时候知道是哪里出错并去相应的地方改正，而且还能加深自己对学科知识的掌握，因为这些代码都是书本上知识的实现。通过阅读别人的代码，能更加清楚书本知识的原理
3. 还有一部分功能没写完，比如说可以结合机器学习，深度学习等知识来实现语义分割然后实现智能去除图片里面的目标。

工作进度表

4. 工作进度表

实际工作情况	开始日期	持续天数
项目寻找灵感	5月2日	3
综述1st （多媒体的概念）	5月2日	3
综述2nd （压缩技术）	5月5日	5
学习图像处理基本概念	5月5日	2
学习seam carving图像压缩算法	5月8日	2
了解显著性检测并学习算法	5月11日	3
学习语义分割并实现demo	5月15日	3
学习python的flask框架	5月16日	4
综述3rd （通信框架）	5月16日	4
综述4th （感想）	5月20日	2
完成整个项目	5月23日	5
完成项目的软件介绍书	5月29日	2
完成项目的可行性分析	6月1日	3
进行论文最后的整理	6月6日	3

图 4.1 实际工作图

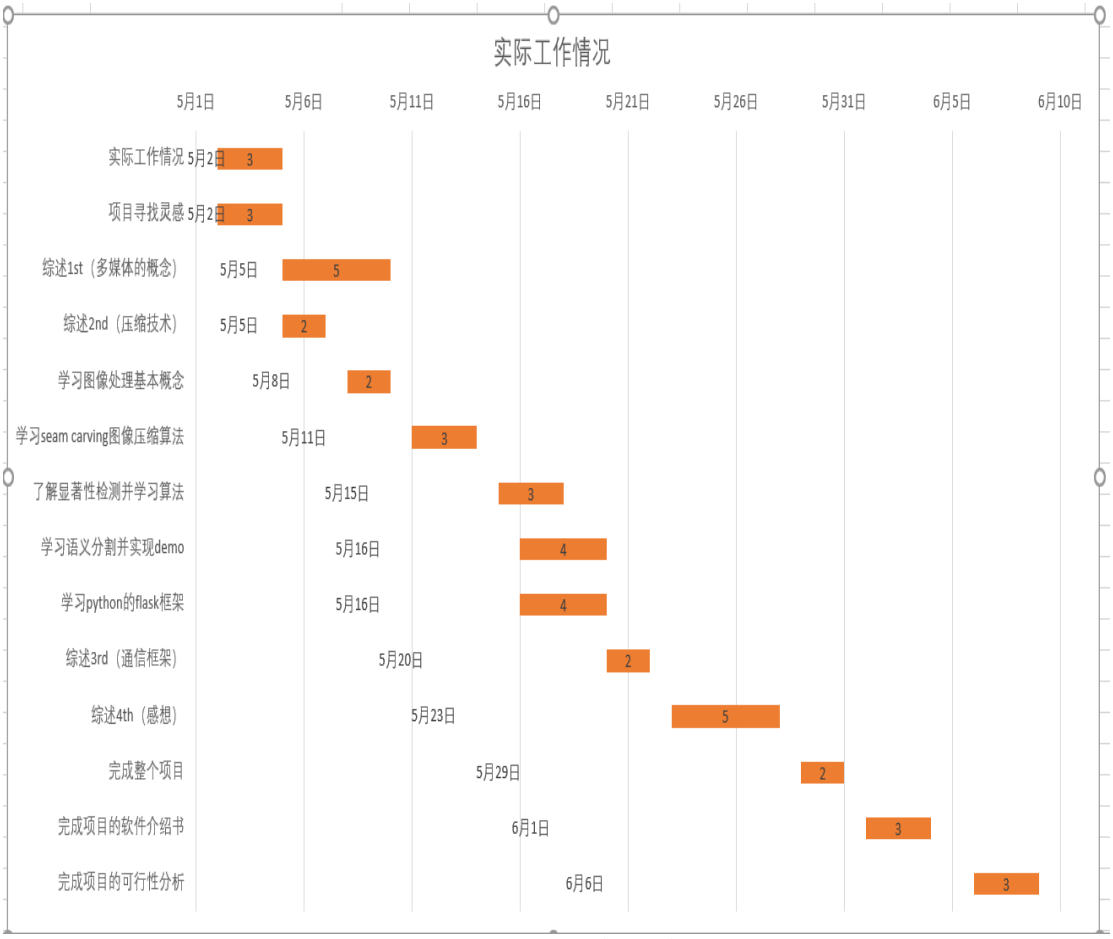


图 4.2 实际工作甘特图

参考文献

5. 参考文献

https://blog.csdn.net/qq_26078953/article/details/92421424

https://zhuanlan.zhihu.com/p/80899085

https://stxnext.com/blog/2019/05/31/flask-vs-django-comparison/

https://www.jianshu.com/p/9960a9667a5c

https://blog.csdn.net/qq_35753140/article/details/90233884

https://stxnext.com/blog/2019/05/31/flask-vs-django-comparison/

https://blog.csdn.net/czl389/article/details/53746668

https://blog.csdn.net/xiaolongrenstep/article/details/78666104

http://www.faculty.idc.ac.il/arik/SCWeb/imret/