Project Name

序号	学号	专业班级	姓名	性别
周五	3140102491	计科 1404	钱旭峰	男

1. Project Introduction

关键帧提取，提取视频中具有表征意义的视频帧作为关键帧，来简洁的表述该视频的内容．主要应用于视频的压缩存储，视频结构化索引等领域．

由于关键帧的选择与人的视觉感受有关，存在一定的主观性，目前，大多数关键帧提取主要是局限某个领域的视频，像体育视频，新闻，电影等．

其中主要有以下几种手段：

绝对帧间差法

图象象素差法

图象数值差法

颜色直方图法

压缩域差法

矩不变量法

边界跟踪法

运动矢量法

这里我主要运用了 opencv 库来完成我的实验，其中的两种算法为

1.颜色直方图法

2.基于运动分析(光流分析)

2. Technical Details

颜色直方图法 ：

颜色直方图是在许多图像检索系统中被广泛采用的颜色特征。它所描述的是不同色彩在整幅图像中所占的比例，而并不关心每种色彩所处的空间位置，即无法描述图像中的对象或物体。颜色直方图特别适于描述那些难以进行自动分割的图像

算法描述：

首先读取视频

之后创建直方图和 image 的空间

从视频中读取每一帧的画面

将画面转为 HS 二维直方图

比较，每一帧与上一帧直方图的差距，若差距大于阈值则是为关键帧保存为图片

具体算法：

```
//如果是第一帧，需要申请内存，并初始化
        if(nFrmNum == 1)
        {
            pBkImg = cvCreateImage(cvSize(pFrame->width, pFrame->height), IPL_DEPTH_8U,1);
            cvSaveImage(PicName, pFrame );//把图像写入文件
            CurrentKeyFrame = cvCreateImage(cvSize(pFrame->width, pFrame->height),  IPL_DEPTH_8U,3);
            cvCopy(pFrame,CurrentKeyFrame, NULL);
        }
        else
        {
            //cvSaveImage(PicName, pFrame );//把图像写入文件
            LoadHist(CurrentKeyFrame,hist1);
            LoadHist(pFrame,hist2);
            HistCompare(hist1,hist2);//比较两个直方图的差距
            if (total >(pFrame->width * pFrame->height)/4)
            {
                cout << total;
                total = 0;
                cvCopy(pFrame,CurrentKeyFrame, NULL);
                cvSaveImage(PicName, pFrame );
            }
```

我主要用了 opencv 库：

cvCreateImage：创建图像空间

cvCvtColor：将 RGB 转为 HSV

cvCvtPixToPlane:将像素转为 3 个平面

cvCreateHist：创建直方图空间

。。。

基于运动分析(光流分析)：

光流的概念是 Gibson 在 1950 年首先提出来的。它是空间运动物体在观察成像平面上的像素运动的瞬时速度，是利用图像序列中像素在时间域上的变化以及相邻帧之间的相关性来找到上一帧跟当前帧之间存在的对应关系，从而计算出相邻帧之间物体的运动信息的一种方法。一般而言，光流是由于场景中前景目标本身的移动、相机的运动，或者两者的共同运动所产生的。

算法描述：

首先读取视频

之后创建目录

提取每一帧，对每一帧的 提取灰度信息 之后 缩放为原来的四分之一 最后 做高斯模糊，计算每一帧的参数和相邻帧的差别保存到字典中（python）

统计每一帧与上一桢的差值，记录最大最小，平均值，标准差等信息

从每一帧中用 k-mean 聚类算法提取多个主要颜色，并返回主要颜色们的质心，存到字典中

根据字典中的信息（主要是平均值和标准差）计算阈值，当相邻的帧之间的差距大于阈值的时候就把这一帧保存为图片

主要算法：

```
def detectScenes(sourcePath, destPath, data, name, verbose=False):
    destDir = os.path.join(destPath, "images")

    # TODO make sd multiplier externally configurable
    #diff_threshold = (data["stats"]["sd"] * 1.85) + data["stats"]["mean"]
```

```
diff_threshold = (data["stats"]["sd"] * 2.05) + (data["stats"]["mean"])


cap = cv2.VideoCapture(sourcePath)
for index, fi in enumerate(data["frame_info"]):
    if fi["diff_count"] < diff_threshold:
        continue


    cap.set(cv2.CAP_PROP_POS_FRAMES, fi["frame_number"])
    ret, frame = cap.read()


    # extract dominant color
    small = resize(frame, 100, 100)
    cols = extract_cols(small, 5)
    data["frame_info"][index]["dominant_cols"] = cols
    if frame != None:
        writeImagePyramid(destDir, name, fi["frame_number"], frame)
```

我主要用了 python 中的 opencv 库：

VideoCapture：获取视频

resize：重定义图片大小

cvtColor:提取灰度信息

GaussianBlur：做高斯模糊

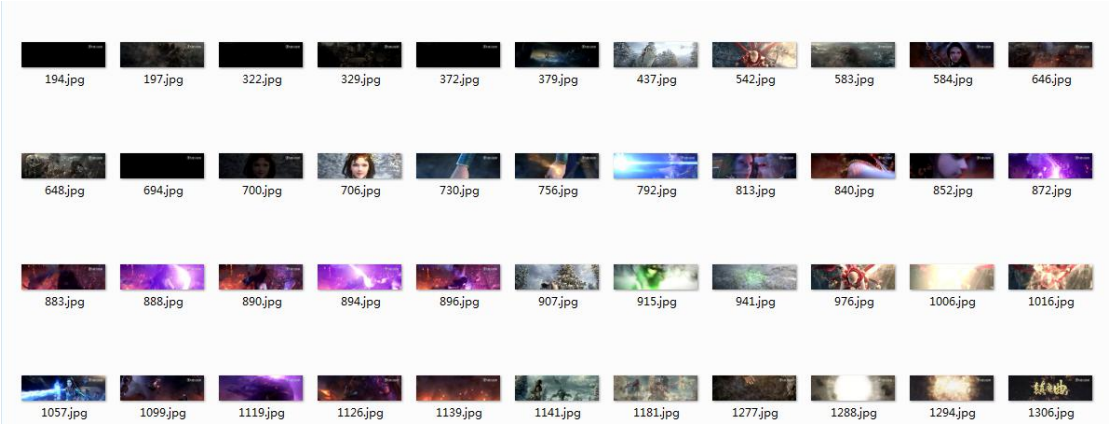
。。。

3. Experiment Results

我用了一个一分钟左右的视频做检测，这个视频中镜头切换频繁，所以最后的关键帧书比较多，大概 50-60 张

颜色直方图法 ：

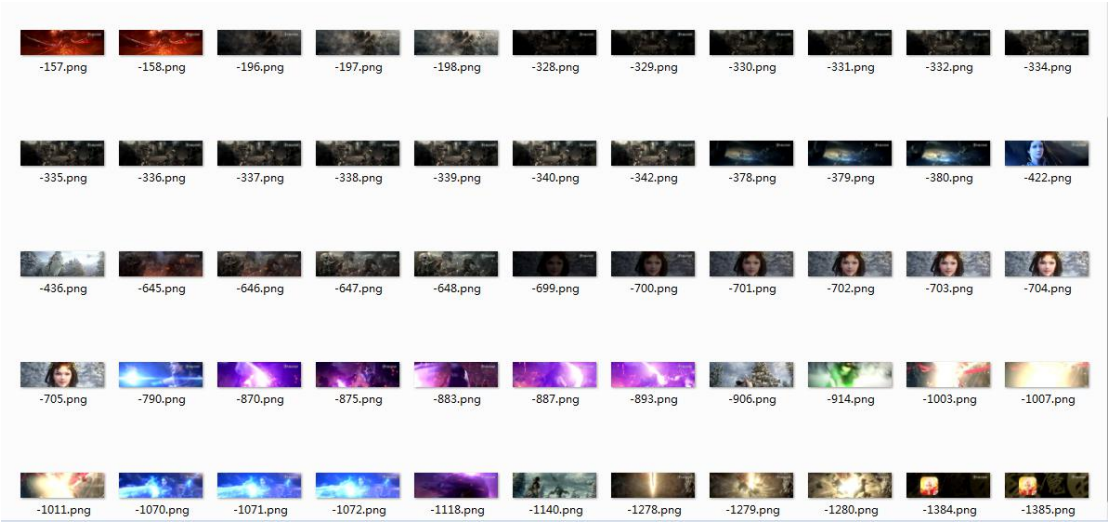
这个方法效果比较好



几乎所有的关键帧都有捕捉到，但是有点不足的就是在开头的那几部分，有存在关键帧重复捕捉的现象，再调参数的时候，当参数过大，关键帧不怎么会重复但有可能会漏掉几个关键帧，当参数过小的时候，关键帧完全捕捉但关键帧会重复

基于运动分析(光流分析)：

这个算法相对于上面那个算法效果较差：



可以看到这个算法的关键帧重复较为严重，而且捕捉不全面

References:

一种基于视频聚类的关键帧提取方法

http://wenku.baidu.com/link?url=UpbpmTSGyH3V7rzTlHg2OhlujVwuGDaHQsYbDJ9hj_sG0l5FPhSDBCwBExteuJEZOS-ieqaUDOH940o8CDNfenTVj0FfUJ1sNP

MUUm8NXjS