

# Key Frame Detection

序号	学号	专业班级	姓名	性别
1	3181010940	数媒1801班	李沛瑶	女

## 1. Project Introduction

### (1) 选题

本次 project 作业，我选择的题目是《视频关键帧提取》。

### (2) 工作简介，即要做什么事情

编写程序进行视频镜头中的关键帧提取（2种以上算法）

程序输入：视频片段（mp4格式）

程序输出：关键帧图像集合（jpg格式）

我选取的算法有：

绝对帧间差法（镜头边界法）

特征转变法

基于聚类的关键帧提取

### (3) 开发环境及系统运行要求，包括所用的开发工具、开发包、开源库、系统运行要求等

开发环境：python3.8

需要用到的包：

numpy 1.13.3

opencv 3.3.1

pillow 8.0.1

scipy 1.5.2

## 2. Technical Details

### (1) 工程实践当中所用到的理论知识阐述

#### 1. 镜头边界法：

在一组镜头中，相邻图像帧之间的特征变化很少，所以整个镜头中图像帧的特征变化也应该不大，因此选择镜头的第一帧和最后一帧可以将镜头内容完全表达出来。

这种方法计算简单，但是它不考虑当前镜头视觉内容的复杂性，这样做并不合理，事实上首帧和尾帧往往并非关键帧，不能精确地得到效果较好的关键帧提取结果。

#### 2. 特征转变法：

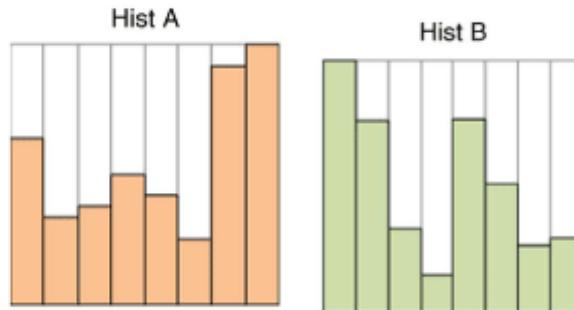
1. 在基于视频图像特征提取关键帧方法中，镜头当前帧与最后一个判断为关键帧的图像比较，如果有较多特征发生改变，则当前帧为新的一个关键帧。
2. 在实际中，可以将视频镜头第一帧作为关键帧，然后比较后面视频帧图像与关键帧的图像特征是否发生了较大变化，逐渐得到后续关键帧。

在实际实验中，使用帧图像的直方图作为帧图像的特征，颜色直方图的差别函数有以下三种，其中实验中选用的颜色直方图的差别函数是直方图的交。

#### 颜色直方图法的差别函数：

1. 通过直方图差度量

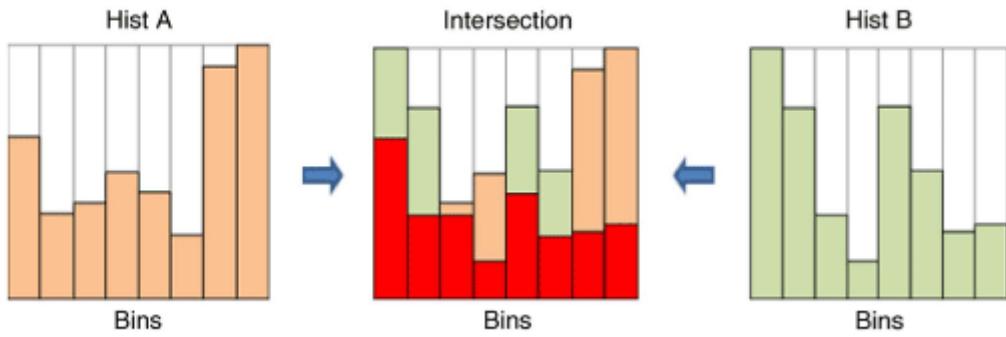
$$d(f, f') = \left| \sum_{j=0}^N H(f, j) - H(f', j) \right|$$



2. 带权重的直方图差

$$d(f, f') = \frac{r}{s} \cdot d(f, f')^{(red)} + \frac{g}{s} \cdot d(f, f')^{(green)} + \frac{b}{s} \cdot d(f, f')^{(blue)}$$

3. 直方图的交



$$d(f, f') = \frac{s(f, f')}{\sum_{j=0}^N H(f, j)}$$

$$s(f, f') = \sum_{j=0}^N \min(H(f, j), H(f', j))$$

按照上述方法，对于不相似的两幅图像进行相似度匹配，会发现相似值很小。而如果对两幅相同的图像，图像相似度为1。

### 3. 基于聚类的关键帧提取

**使用阈值控制聚类密度：**

设某个镜头 $S_i$ 包含 $n$ 个图像帧，可以表示为 $S_i = (F_{i_1}, \dots, F_{i_n})$ 其中 $F_{i_1}$ 为首帧， $F_{i_n}$ 为尾帧。如果相邻两帧之间的相似度定义是这相邻两帧颜色直方图的相似度（也即使直方图特征差别），预定义一个阈值 $\delta$ 控制聚类的密度。

计算当前帧 $F_i$ 与现存某个聚类质心间的距离，如果该值大于 $\delta$ ，则该帧与该聚类之间距离较大，因此 $F_i$ 不能加入该聚类中。如 $F_i$ 与所有现存聚类质心相似度均小于 $\delta$ ，则 $F_i$ 形成一个新的聚类， $F_i$ 成为新聚类的质心。否则将该帧加入到与之相似度最大的聚类中，使该帧与这个聚类的质心之间的距离最小，并如下相应调整该聚类质心：

$$\text{centrod}' = \text{centrod} * F_n / (F_n + 1) + 1 / (F_n + 1) * F_i$$

其中， $\text{centrod}$ 、 $\text{centrod}'$  和 $F_n$ 分别是聚类群原有质心、聚类群更新后质心和该聚类群帧数。

通过上面方法将镜头 $S_i$ 所包含的 $n$ 个图像帧，分别归类到不同聚类后，就可以选择关键帧：从每个聚类中抽取离聚类质心最近的帧作为这个聚类的代表帧，所有聚类的代表键帧就构成了镜头 $S_i$ 的关键帧。

假设镜头 $S_i$ 形成了 $cluster$ 个聚类，那么就可以从镜头 $S_i$ 中提取 $cluster$ 个关键帧。

该聚类算法由阈值 $\delta$ 控制， $\delta$ 越大，形成的聚类数目越多，镜头 $S_i$ 划分越细，选择的关键帧也越多；反之， $\delta$ 越小，所形成的聚类个数越少，镜头 $S_i$ 划分越粗。

### K-means算法（预先设定聚类个数控制聚类）：

聚类是一种对数据分类的方法，聚类和分类最大的不同在于：分类的目标是事先已知的，而聚类则不一样，聚类事先不知道目标变量是什么，类别没有像分类那样被预先定义出来。

K-Means是聚类算法中的最常用的一种，算法最大的特点是简单，好理解，运算速度快，但是只能应用于连续型的数据，并且一定要在聚类前需要手工指定要分成几类。

K-means算法的主要过程如下：

1. 首先输入k的值，即我们希望将数据集经过聚类得到k个分组。
2. 从数据集中随机选择k个数据点作为初始质心
3. 对集合中每一个数据，计算与每一个质心的距离（距离的含义后面会讲），离哪个质心距离近，就加入哪一个类。
4. 这时每一个类中都有很多数据，这时候对每一类中的数据重新计算质心。
5. 如果新质心和之前的质心之间的距离小于某一个设置的阈值（表示重新计算的质心的位置变化不大，趋于稳定，或者说收敛），可以认为我们进行的聚类已经达到期望的结果，算法终止。
6. 如果新质心和之前的质心之间的距离很大，则需要迭代3~5步骤。

## (2) 具体的算法，请用文字、示意图或者是伪代码等形式进行描述（不要贴大段的代码）

### 1. 镜头边界法：

在实际实现时，我们不提前确定镜头划分，而是对一个完整的视频片段，进行帧间差法，计算两个相邻帧的帧间差，通过判断帧间差的大小，确定需要镜头划分的位置，即当帧间差较大时，并把这个位置的帧图像作为一个关键帧图像。

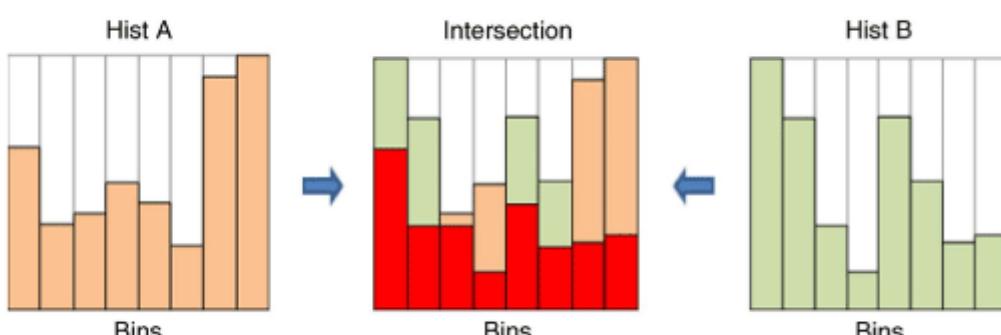
在计算帧间差时，使用的是图像像素数值差，将相邻两帧的帧图像对应点的像素值求差并取绝对值，对整个帧图像的所有像素差求和，用其结果作为两帧相邻图像的帧间差。

$$d(f, f') = \sum_{i=1}^h \sum_{j=1}^w |f_{i,j} - f'_{i,j}|$$

在具体实现时，为了降低物体高速运动带来的影响，不直接对帧间差设置阈值判断，而是把帧间差看成一个关于帧序号的函数曲线，取其中的**极大值点**作为关键帧所在处，为了避免帧间差微小波动带来的极大值过多，事先需要对曲线做一个**平滑操作**，平滑操作之后得到的极大值才可以作为关键帧所在处。

### 2. 特征转变法：

直方图的交：



$$d(f, f') = \frac{s(f, f')}{\sum_{j=0}^N H(f, j)}$$

$$s(f, f') = \sum_{j=0}^N \min(H(f, j), H(f', j))$$

按照上述方法，对于不相似的两幅图像进行相似度匹配，会发现相似值很小。而如果对两幅相同的图像，图像相似度为1。

因此，实验中计算的帧图像特征转变情况可以通过计算直方图的交映射到一个0 – 1的值，1表示两个帧图像直方图完全相同，数值越小，相似度越低。

我们在实验中设置一个阈值，首先选定第一帧作为关键帧，接下来遍历整个视频中的帧图像，如果与之前最后确认的关键帧特征转变过大，使得相似度低于阈值，就可以认定为当前帧为一个新的关键帧，即当前帧成为最后被确认的关键帧。

### 3. 基于聚类的关键帧提取

首先，在实验中我曾经尝试直接对图像信息进行PCA主成分分析降维并聚类，并对降维的结果进行聚类。但是由于视频帧图像信息过多，导致最后生成的矩阵过大，即使使用SVD方法，也无法完成计算，程序运行速度非常慢，因此最后尝试过之后还是放弃了这种方法，而还是选用对直方图进行聚类的方法。

对直方图聚类时，我使用了以下三种方式尝试聚类：

1. 首先使用的是使用阈值控制聚类密度的聚类算法。
2. 普通的 *kmeans* 算法。
3. 之后我又根据对算法及结果的分析，对算法进行了一些改进，比如：
  1. 此时的聚类，并不能在整个数据集中寻找聚类中心，而是需要考虑视频帧图像是具有时间顺序的，所以应该只在帧图像的前后寻找聚类中心。
  2. 初始状态下第一次选择聚类中心时，应该尽量散布在整个视频时序中，才能减轻计算难度，得到更准确的计算结果。否则如果视频的开始和最后有相似的场景，可能会被聚合成一个类而不能被成功分开。
  3. 为了防止迭代次数过多，无法生成结果，应该对迭代次数设置一个阈值。

**改进后的K-Means算法：**

```

1 // 差图像的均值
2 for i ← 0 to frames_number:
3     difference ← frame[i+1] - frame[i]
4     mid_d ← mean(difference)
5     mid_d ← sum(mid_d)
6
7
8 // 选定最初的聚类中心，每隔一定间距选定一个关键帧
9 now ← 0
10 for i ← 0 to k:
11     cluster[k] ← (frame[now])
12     now ← now + frames_number/(k-1)

```

```

13 // 迭代
14 while true:
15     now = 0
16     clear the set: cluster_set
17     for i in range(k):
18         set_cluster[i].clear() // 先清空每个集合
19
20     // 为每一帧选定所在的聚类
21     for i ← 0 to frames_number:
22         ldiff ← frame[i] - Cluster[now]
23         rdiff ← frame[i] - Cluster[now+1]
24
25         if ldiff < rdiff:
26             add i to cluster_set[now]
27         else:
28             add i to cluster_set[now]
29             now ← now + 1
30
31     // 重新计算聚类中心
32     ok = True
33     for i ← 0 to k:
34         for x in cluster_set[i]:
35             new_cluster ← mean(frame[x])
36
37             if difference(Cluster[i] - new_cluster) < threshold:
38                 continue
39             else:
40                 ok ← False
41                 Cluster[i] ← new_cluster
42             // 所有的中心都稳定下来, 停止迭代
43             if ok:
44                 break
45

```

### (3) 程序开发中重要的技术细节

**使用的重要函数:**

#### 1. VideoCapture() 函数

该函数来自opencv库，主要用于处理摄像头或者文件中的视频流

- VideoCapture()中参数是0，表示打开笔记本的内置摄像头
- 参数是视频文件路径则打开视频，如cap = cv2.VideoCapture("../test.mp4")

#### 2、calcHist()函数

该函数来源于opencv库，主要用于计算图像的直方图

函数原型：

```
1 | cv.calcHist(images, nimages, channels, mask, hist, dims, histSize, ranges,
2 |         uniform=true, accumulate=false)
```

函数详解：

- images: 输入图像
- nimages: 输入图像的个数
- channels: 需要统计直方图的第几通道
- mask: 掩膜
- hist: 输出的直方图数组
- dims: 需要统计直方图通道的个数
- histSize: 指的是直方图分成多少个区间
- ranges: 统计像素值的区间
- uniform=true: 是否对得到的直方图数组进行归一化处理
- accumulate=false: 在多个图像时，是否累计计算像素值的个数

### 3、np.linalg.norm()函数

来源于numpy库，主要用于求范数，本实验中计算欧式距离时使用

函数原型：

```
1 | np.linalg.norm(x, ord=None, axis=None, keepdims=False)
```

函数详解：

- x: 输入矩阵
- ord: 取值1, 2, np.inf分别表示1范式, 2范式, 无穷范数
- axis: 0, 1, none默认, 0代表按列处理, 1代表按行处理
- keepdims: 是否保持维数

### 4、np.convolve()函数

```
1 | numpy.convolve(a, v, mode='full')
```

来源于numpy库，这是numpy中的卷积函数库，主要用于计算卷积，在本实验中用于对帧间差曲线做平滑操作

函数详解：

- a (长度为N) : 输入的一维数组
- b (长度为M) : 输入的第二个一维数组
- mode: 参数可选 {'full', 'valid', 'same'}
  - 'full' 默认值，返回每一个卷积值，长度是N+M-1,在卷积的边缘处，信号不重叠，存在边际效应。
  - 'same' 返回的数组长度为max(M, N)，边际效应依旧存在。
  - 'valid' 返回的数组长度为max(M,N)-min(M,N)+1，此时返回的是完全重叠的点。边缘的点无效。

### 3. Experiment Results

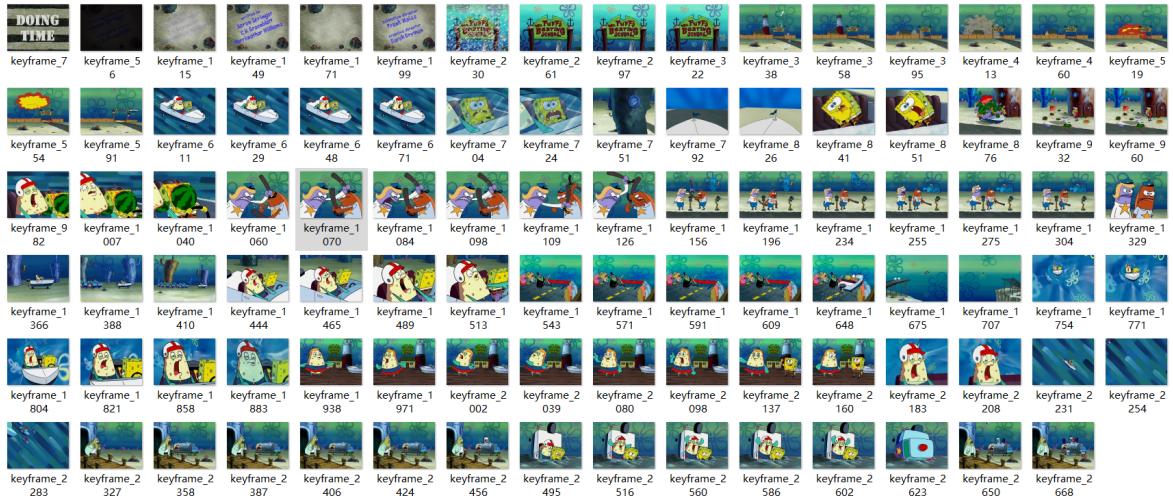
首先，使用的测试视频是test.mp4，该视频一共有2691帧。我首先将所有的帧图像输出，如图：



经过人眼自行判断和筛选，选出了80个关键帧，结果如图，保存在keyframe文件夹下。



#### 1. 帧间差法（镜头切割法）



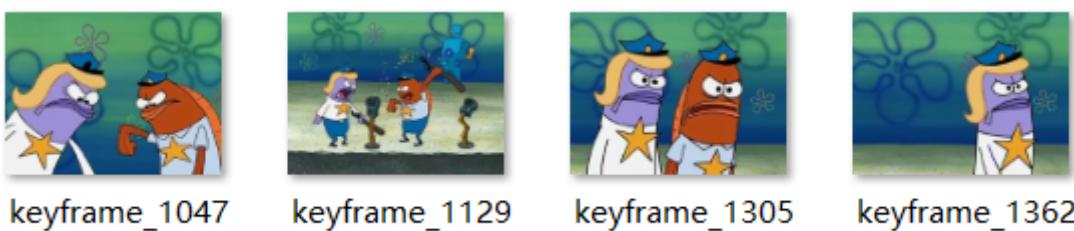
如图，是基于帧间差法得到的关键帧提取结果，通过与自行提取的关键帧对比，可以发现，对于变化明显的相邻帧，该算法基本能够完全检测出应该出现的关键帧，但是由于是直接对对应的像素值取差值，该算法对物体运动十分敏感，因此图像中物体的位移和人物表情动作的变换也会使得该算法将其判断为一个新的关键帧，导致关键帧重复性高，如下图：



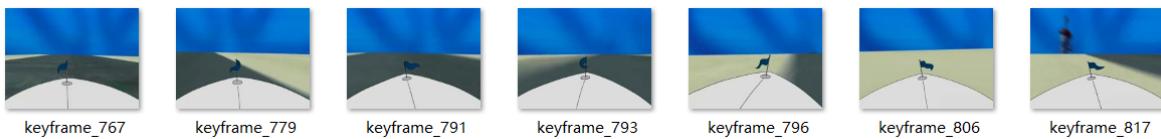
## 2. 特征转变法



1. 如图，使用该算法的识别结果，对于较为剧烈的场景转换，识别效果也基本比较完整，主要的场景转换都能被识别到，但是该算法由于是对直方图的特征变化进行检测，因此对颜色变化的敏感度非常高。但是对对象的微小运动的敏感性降低了许多。如下图：



2. 帧图像主要场景没有变换，但由于颜色较少，色块占比较大，色块的平移就会导致直方图发生较大的变化，被识别成不同的关键帧：



3. 对两个场景的平缓过度识别效果较差，特别是黑场过渡，因为在过渡的过程中，直方图会经历较亮处占比较大，到全是黑色，再到较亮处占比较大的过程，在这个过程中会有多个阶段的直方图被识别成为关键帧。



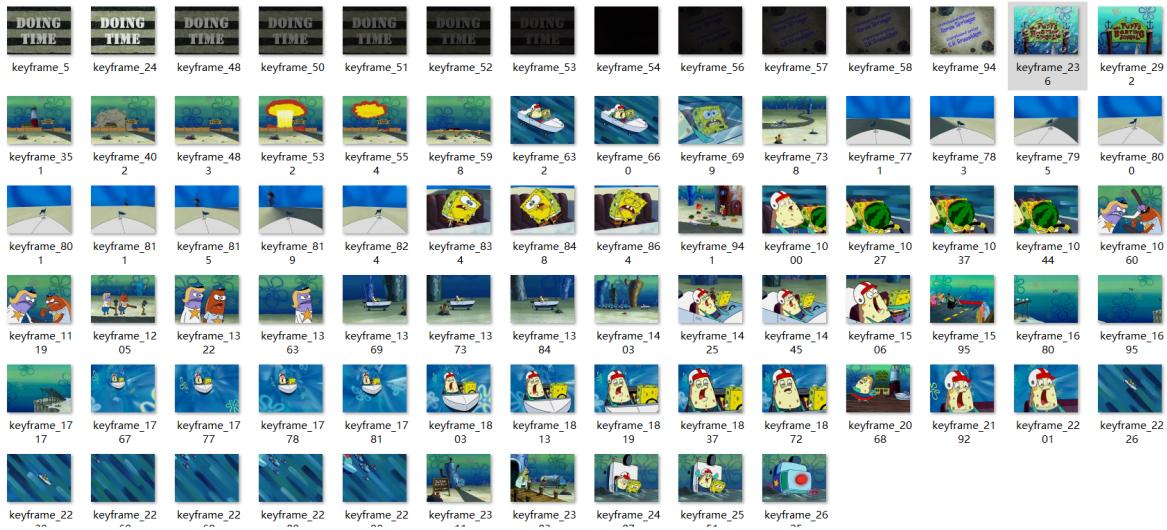
4. 帧图像的主色调没有发生变化，但仅仅是由于其中2-3种颜色的占比在不停波动，就被识别成不同的关键帧，导致关键帧识别重复。对场景中对象的缩放较为敏感，如图，背景主色调为蓝色，而对象主色调为黄色和白色，对象的放缩就会对直方图的影响较大，从而使得在缩放过程中有多个图像被识别为关键帧。



### 3. 基于聚类的关键帧提取

#### (1) 使用普通的kmeans算法：

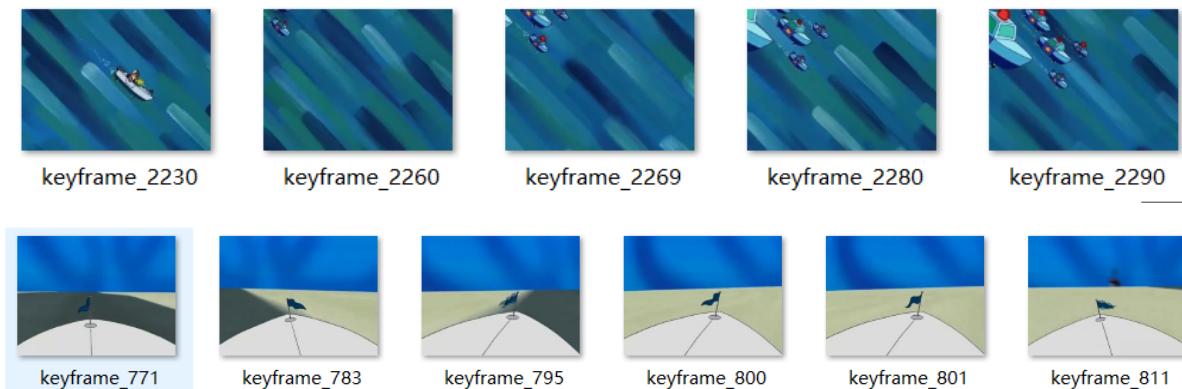
不考虑视频时序相关性



1. 如图，可以看出，由于聚类时会对整个视频序列进行搜索，因此，关键帧重复检测的现象降低了许多，而只会在场景中发生较大的对象增减时检测为关键帧，



2. 但是由于还是对直方图数据进行聚类，因此2中的情况仍然轻微存在，但已经有了一定的改善。



像上图这种大面积色块变化的现象，只要是采用直方图的方法进行计算，就很难避免。

3. 另外，该算法对画面平滑过渡的检测效果明显得到了改善，下面第一张图是第二种算法对于一种变形过渡的检测结果，可以看出由于时序的限制，导致很多过渡过程帧被识别为关键帧：



而第三种算法的识别效果：由于对直方图进行了聚类，非常明显的避免了以上情况，因为过渡帧都可以被聚合到前后两种关键帧中。



keyframe\_1819



keyframe\_1837



keyframe\_1872



keyframe\_2068

但是对于黑场过渡的关键帧重复识别还是无法避免：



## (2) 使用改进的kmeans算法：

考虑视频时序特征，在进行聚类时，对于每一帧图像，只与前后两个关键帧中心比较，决定应该放入哪一个聚类集合，而不是在全局进行搜索。



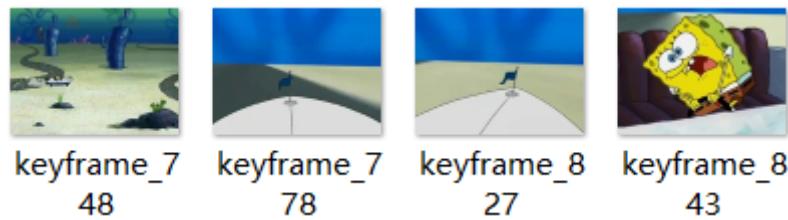
1. 如图，该算法对于黑场过渡的关键帧提取展示出了非常优秀的结果，通过对帧图像的聚类几乎完全摒弃了黑场特效：



2. 该算法对于前后场景相似但被其他场景隔开的情况也能够正确识别。



3. 该算法还非常优秀的展示出了对大片色块平移变化的关键帧提取，如图，在这种情况下，重复的关键帧数目大大减少，显示出非常优秀的检测效果。



4. 但是该算法仍然有一个缺点，那就是一开始选取初始关键帧的时候是对时序序列帧图像均分选取的，虽然在多次的迭代中，重复的图像会趋向于聚合为一类，但还是出现了：在变化剧烈的视频部分选取的关键帧不够多，而在变化平缓的视频部分选取的关键帧又会重复，程序无法根据变化剧烈程度动态选择初始关键帧。

## References:

---

- Kmeans算法：  
[https://blog.csdn.net/sinat\\_36710456/article/details/88019323](https://blog.csdn.net/sinat_36710456/article/details/88019323)
- 关键帧提取：  
[https://blog.csdn.net/caiji\\_is\\_studying/article/details/103718527](https://blog.csdn.net/caiji_is_studying/article/details/103718527)  
<https://blog.csdn.net/Ailberty/article/details/109581016>
- PCA降维  
<https://blog.csdn.net/zouxiaolv/article/details/100590725>