视频字幕识别项目文档

同济大学

2018 年 3 月 30 日

1 简介

本项目基于 OpenCV 以及 Azure 的 OCR 服务，成功完成了任务。
程序分为两部分:

- 基于 OpenCV 的字幕提取：利用多线程，使得读取视频和提取字幕并行；**在提取字幕方面，我们开发了一种算法，能够很好地剔除重复、无关的帧，提取出含有字幕的帧，并能滤掉背景 (使成黑色)，仅使字幕 (白色) 清晰显示。**
- 基于 Azure 的 OCR：读取上一步得到的图片文件，返回识别结果.

实践证明，我们的算法不仅性能优越，同时也得到了很好的结果。**precision 最高达到了 100%，且 recall 也能接近 99%。处理接近 3 小时的电影，提取字幕仅用时 12 分钟。**

2 使用说明

Windows 环境下在 powershell 下运行
> ./run.exe your_vedio_path subtitle_save_path
需要输入两个路径:

- 视频的存放路径
- 字幕的存放路径

然后调用 OCR.py 启动 Azure 的 OCR 服务。

3 算法原理

3.1 字幕区域识别

我们采用传统的图像处理方法，能够很好的识别并定位出字幕的区域，同样也能判断是否存在字幕。

我们的算法可以用下面伪代码描述 (为了清晰起见，调用函数的名称与 MATLAB 函数或者 OpenCV 函数一致)：

算法 1 字幕区域识别

输入: 视频某帧图片

输出: 判断是否存在字幕，如果存在并返回处理后的含字幕图片

```
1:  function SubtitleLocating(input_img, output_img)
2:      Let temp_img be new matrix
3:      temp_img ← RGB2GRAY(input_img)
4:      temp_img ← IMADJUST(temp_img)
5:      output_img ← IMOPEN(temp_img)
6:      output_img ← IMSUBSTRACT(temp_img, output_img)
7:      output_img ← IMOPEN(output_img)
8:      output_img ← IMCLOSE(output_img)
9:      Let contours be new vector of Points
10:     contours ← FINDCONTOURS(output_img)
11:     has_caption ← False
12:     for i = 0 → contours.size do
13:         rect ← BOUDINGRECT(contours[i])
14:         if the rect is at center then
15:             has_caption ← True
16:             output_img(rect) ← temp_img(rect)
17:         else
18:             output_img(rect) ← Black
19:         end if
20:     end for
21:     return has_caption
22: end function
```

算法的第 3-6 行，是本算法的核心内容。经过第 3 行的灰度化后，第 4 行的 imadjust 图像增强是为了将图片的灰度映射到更紧密的一个范围中去，方便下面的处理，实践证明这一步对结果是有举足轻重的作用的。第 5 行的 imopen 形态学开操作，是该算法的精髓。这一步可以将字幕区域模糊化，甚至可以做到直接融于背景色中。所以通过最后一步，也就是第六行的差分操作，就可以轻松得到清晰的字幕。其实大多数视频帧做到这一步，效果已经非常好了。

算法的 7-21 行，是进一步滤掉背景噪声，得到更为干净的字幕。首先通过算法第 7-8 行的形态学开操作和闭操作，使得字幕的区域能连通成一体。然后通过 9-10 行的一个寻找连通区域的函数，经过筛选，就可以锁定字幕所在的连通区域。我们确立了三条筛选的原则：一是连通区域的平均水平位置是在中央附近 50 像素以内；二是连通区域的底部不高于图片底部 30 像素；三是连通区域大小必须超过 4000. 如果没有任何一个连通区域，或者不满足上述任一条件，那么判定为该图片不存在字幕，同时把不符合要求的区域变成黑色；若存在一个区域能满足上述所有条件，那么满足的区域就是字幕所在区域。这样，就得到了干净的字幕。

3.2 字幕去重

我们采取的提取策略是每 20 帧提取一张图片进行处理，其余的全都忽略。但是很多时候，一条字幕会跨很多帧。这样势必要解决一个去重的问题。

我们的解决策略是计算上一张处理后的字幕图片与当前处理后的字幕图片的余弦相似度，

$$cosine = \frac{dot(A,B)}{norm2(A) * norm2(B)} \tag{1}$$

我们判定，$cosine > 0.5$ 的视为两张图片相同;$cosine < 0.5$ 的视为两张图片不同。因为我们滤掉了背景，所以这种方法对于字幕去重极为有用。那么，每当出现前一张和当前这张图片计算余弦相似度，出现 $cosine < 0.5$ 的情况，那么就保存当前这张图片；反之认为图像一致，不进行保存。这样就完成了字幕的去重。

3.3 生产者-消费者模型

为了提高程序的效率，我们并不是等视频处理函数读取完每帧图片后再进行处理，而是边读取视频，边处理图片。我们采用了著名的生产者- 消费

者模型。

我们设置了一个队列，保存图片的矩阵头信息 (OpenCV 中的 Mat)。同时开了两个线程，一个读取视频流，每二十帧将图片的矩阵头压入队列中，此为生产者；另一个线程进行从队列中取矩阵头，处理图片，此为消费者。

为了防止多线程同时访问共享资源，我们设置了锁。利用 C++11 的 unique_lock 和 condition_variable，可以保证队列线程安全。

4 性能与效果

示例：

图 1: 原图

图 2: 提取图

更多图片请见附件.

性能上，我们对很多视频的字幕提取进行了测速，得到了一个平均值: 每秒钟处理能力为 390 帧左右。6 分钟就能处理完 1h30m 的视频 (配置:i7-6700)。

效果上，除非是由于字幕出现不足 20 帧因而被略过的缘故，一般都能找到所有字幕。但是也会掺杂不是字幕的图片，不过一般这些“杂质”不能识别，因此对识别的结果不会产生影响。

识别结果见附件 output.txt.