

[Python图像处理] 二十九.MoviePy视频编辑库实现抖音短视频剪切合并操作

原创 Eastmount 2020-10-06 21:36:43 6549 收藏 70

编辑 版权

分类专栏: Python图像处理及图像识别 文章标签: 视频处理 Python MoviePy 视频合并 视频剪辑



Python图像处理及图像识别

¥9.90

本专栏主要结合Python语言讲述图像处理相关的知识,从二值图像、灰度图像到RGB图像基础知识,再到常见的图像处理算法,包括:灰度算法、图像锐化、图像分割等知识,最后会结合深度学习和机器...



Eastmount

该系列文章是讲解Python OpenCV图像处理知识,前期主要讲解图像入门、OpenCV基础用法,中期讲解图像处理的各种算法,包括图像锐化算子、图像增强技术、图像分割等,后期结合深度学习研究图像识别、图像分类应用。希望文章对您有所帮助,如果有不足之处,还请海涵~

前面一篇文章详细讲解了OpenCV快速实现人脸检测,涉及图像、视频、摄像头。这篇文章将介绍MoviePy视频编辑库,实现视频的自定义剪切和合并操作,基础性文章,希望对你有所帮助。同时,该部分知识均为杨秀璋查阅资料撰写,转载请署名“CSDN+杨秀璋”及原地址出处,谢谢!!

该扩展包缺点:速度太慢,真的很慢,后面看看有没有更好的替代方法。

该系列在github所有源代码:

- <https://github.com/eastmountyxz/ImageProcessing-Python>
- <https://github.com/eastmountyxz/CSDNBlog-ImageProcessing-Python>

前文参考:

[Python图像处理] 一.图像处理基础知识及OpenCV入门函数
[Python图像处理] 二.OpenCV+Numpy库读取与修改像素
[Python图像处理] 三.获取图像属性、兴趣ROI区域及通道处理
[Python图像处理] 四.图像平滑之均值滤波、方框滤波、高斯滤波及中值滤波
[Python图像处理] 五.图像融合、加法运算及图像类型转换
[Python图像处理] 六.图像缩放、图像旋转、图像翻转与图像平移
[Python图像处理] 七.图像阈值化处理及算法对比
[Python图像处理] 八.图像腐蚀与图像膨胀
[Python图像处理] 九.形态学之图像开运算、闭运算、梯度运算
[Python图像处理] 十.形态学之图像顶帽运算和黑帽运算
[Python图像处理] 十一.灰度直方图概念及OpenCV绘制直方图
[Python图像处理] 十二.图像几何变换之图像仿射变换、图像透视变换和图像校正
[Python图像处理] 十三.基于灰度三维图的图像顶帽运算和黑帽运算
[Python图像处理] 十四.基于OpenCV和像素处理的图像灰度化处理
[Python图像处理] 十五.图像的灰度线性变换
[Python图像处理] 十六.图像的灰度非线性变换之对数变换、伽马变换
[Python图像处理] 十七.图像锐化与边缘检测之Roberts算子、Prewitt算子、Sobel算子和Laplacian算子
[Python图像处理] 十八.图像锐化与边缘检测之Scharr算子、Canny算子和LOG算子
[Python图像处理] 十九.图像分割之基于K-Means聚类的区域分割
[Python图像处理] 二十.图像量化处理和采样处理及局部马赛克特效
[Python图像处理] 二十一.图像金字塔之图像向下取样和向上取样
[Python图像处理] 二十二.Python图像傅里叶变换原理及实现
[Python图像处理] 二十三.傅里叶变换之高通滤波和低通滤波
[Python图像处理] 二十四.图像特效处理之毛玻璃、浮雕和油漆特效
[Python图像处理] 二十五.图像特效处理之素描、怀旧、光照、流年以及滤镜特效
[Python图像处理] 二十六.图像分类原理及基于KNN、朴素贝叶斯算法的图像分类案例
[Python图像处理] 二十七.OpenGL入门及绘制基本图形 (一)

文章目录

- 一.MoviePy简介及安装
- 二.MoviePy基础用法
- 三.MoviePy抖音短视频剪切
- 四.MoviePy抖音短视频合并
- 五.总结

一.MoviePy简介及安装

MoviePy是一个用于视频编辑的Python模块，包括切割、连接、标题插入、视频合成、非线性编辑，视频处理等功能，甚至可以用它增加一些自定义的高级特效。此外，MoviePy可以读写绝大多数常见的音频和视频格式，包括GIF格式，并且可以在Windows / Mac / Linux上运行，这也意味着，MoviePy项目可以部署到服务端，在服务端进行视频处理。该扩展包带有Python 2.7+和Python3版本，是一个视频编辑良好扩展包（ffmpeg亦推荐学习）。



下载地址：

- <https://pypi.org/project/moviepy/>

Github：

- <https://github.com/Zulko/moviepy>

Project description

pypi package 1.0.3 chat on gitter build passing build passing coverage 68%

MoviePy (full [documentation](#)) is a Python library for video editing: cutting, concatenations, title insertions, video compositing (a.k.a. non-linear editing), video processing, and creation of custom effects. See the [gallery](#) for some examples of use.

MoviePy can read and write all the most common audio and video formats, including GIF, and runs on Windows/Mac/Linux, with Python 2.7+ and 3 (or only Python 3.4+ from v.1.0). Here it is in action in an IPython notebook:



本文主要通过MoviePy实现短视频的剪切和合并操作。随着自媒体公司频繁在各个平台进行视频搬运，对于视频原创性的要求越来越高，用MoviePy可以批量实现视频编辑，结合MoviePy的跨平台特性，可以实现在服务端视频采集，自动处理，发布的流水线作业。推荐ucsheep老师的部分文章：

- [MoviePy - 中文文档\(一个专业的python视频编辑库\)教程](#)

MoviePy安装如下图所示:

- pip install moviepy



```
C:\WINDOWS\system32\cmd.exe - pip install moviepy

C:\Users\xiuzhang>cd..

C:\Users>cd..

C:\>cd C:\Software\Program Software\Python37\Scripts

C:\Software\Program Software\Python37\Scripts>pip install moviepy

Collecting moviepy
  Downloading moviepy-1.0.3.tar.gz (388 kB)
    |#####| 388 kB 139 kB/s
Requirement already satisfied: decorator<5.0,>=4.0.2 in c:\software\program software\python37\lib\site-packages (from moviepy) (4.4.1)
Requirement already satisfied: tqdm<5.0,>=4.11.2 in c:\software\program software\python37\lib\site-packages (from moviepy) (4.48.2)
Requirement already satisfied: requests<3.0,>=2.8.1 in c:\software\program software\python37\lib\site-packages (from moviepy) (2.22.0)
Collecting proglog<=1.0.0
  Downloading proglog-0.1.9.tar.gz (10 kB)
Requirement already satisfied: numpy<=1.17.3 in c:\software\program software\python37\lib\site-packages (from moviepy) (1.17.4)
Collecting imageio<3.0,>=2.5
  Downloading imageio-2.9.0-py3-none-any.whl (3.3 MB)
    |#####| 3.3 MB 1.3 MB/s
Collecting imageio-ffmpeg<=0.2.0
  Downloading imageio-ffmpeg-0.4.2-py3-none-win32.whl (19.6 MB)
    |#####| 19.6 MB 1.3 MB/s eta 0:00:10

Requirement already satisfied: urllib3<1.25.0,!<1.25.1,<1.26,>=1.21.1 in c:\software\program software\python37\lib\site-packages (from requests<3.0,>=2.8.1->moviepy) (1.25.7)
Requirement already satisfied: certifi<=2017.4.17 in c:\software\program software\python37\lib\site-packages (from requests<3.0,>=2.8.1->moviepy) (2019.11.28)
Requirement already satisfied: idna<2.9,>=2.5 in c:\software\program software\python37\lib\site-packages (from requests<3.0,>=2.8.1->moviepy) (2.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\software\program software\python37\lib\site-packages (from requests<3.0,>=2.8.1->moviepy) (3.0.4)
Requirement already satisfied: pillow in c:\software\program software\python37\lib\site-packages (from imageio<3.0,>=2.5->moviepy) (7.0.0)
Building wheels for collected packages: moviepy, proglog
  Building wheel for moviepy (setup.py) ... done
    Created wheel for moviepy: filename=moviepy-1.0.3-py3-none-any.whl size=110732 sha256=056c1864541f31b86e5309f00c6475a3a35628933994b6730561b2d5eb7849be
    Stored in directory: c:\users\xiuzhang\appdata\local\pip\cache\wheels\56\dc\2b\9cd600d483c04af3353d66623056fc03faed76b7518fae4df
  Building wheel for proglog (setup.py) ... done
    Created wheel for proglog: filename=proglog-0.1.9-py3-none-any.whl size=6153 sha256=29d7d5a79557838159a1152d54e4b94b7c86a053ef38ba60723c407ed9ead
    Stored in directory: c:\users\xiuzhang\appdata\local\pip\cache\wheels\12\36\1f\dc61e6ac10781d63cf6fa045eb09fa613a667384e12cb6e6e0
Successfully built moviepy proglog
Installing collected packages: proglog, imageio, imageio-ffmpeg, moviepy
Successfully installed imageio-2.9.0 imageio-ffmpeg-0.4.2 moviepy-1.0.3 proglog-0.1.9
WARNING: You are using pip version 20.2.2, however, version 20.2.3 is available.
You should consider upgrading via the 'c:\software\program software\python37\python.exe -m pip install --upgrade pip' command.
```

常见错误

安装MoviePy运行代码如果报错缺少ImageMagick或decode转码错误,“This error can be due to the fact that ImageMagick is not installed on your computer”,这是需要安装ImageMagic可执行文件。ImageMagic是用在视频中填入文本信息的工具,需要单独下载exe程序安装。可以参考下面这篇文章解决。

- https://blog.csdn.net/weixin_42081389/article/details/104322629
- <https://imagemagick.org/script/download.php#windows>

安装过程注意勾选Install development headers and libraries for C and C++。


```
#修改为刚刚ImageMagic的安装路径
IMAGEMAGICK_BINARY = r"C:\Program Files\ImageMagick-7.0.10-Q16-HDRI\magick.exe"
```

二.MoviePy基础用法

案例一：视频翻转

下面首先看一个简单的官方例子，它将视频翻转。

```
from moviepy.editor import *

#视频旋转180度
clip = VideoFileClip("cat01.mp4").rotate(180)

#The size of the clip, (width,height) in pixels
print(clip.size) #(720, 1280)

#播放视频
#clip.ipython_display(width=100)

#写入视频
clip.write_videofile("cat01_rotate.mp4")
```

输出结果如下图所示，成功将视频翻转，但是视频加载的时间较长。



案例二：自定义logo

再来看一个示例，它将在视频5s和10s之间的子剪辑屏幕中央添加一个标题，然后将结果写入一个新文件。

```
from moviepy.editor import *

#加载视频5-10片段
clip = VideoFileClip("cat02.mp4").subclip(5,10)
print(type(clip))

#降低音量x0.8
clip = clip.volumex(0.8)

#生成文本 自定义颜色字体
```



```

times_list = [i * (times / n) for i in range(n + 1)]
#[0.0, 1.25, 2.5, 3.75, 5.0]
print(times_list)
logos = []

#生成文本 自定义颜色字体
for i in range(n):
    txt_clip = TextClip("CSDN Eastmount 2020", fontsize=50, font='Simhei', color='blue')

    #显示时间 位置
    txt_clip = (txt_clip.set_start(times_list[i]).set_end(times_list[i + 1])
                .set_pos((random.randint(0, video.w), random.randint(0, video.h))))
    logos.append(txt_clip)

#Overlay text on video
result = CompositeVideoClip([video, *logos])

#写入文件 mp4文件默认用Libx264编码 比特率单位bps
result.write_videofile("cat03_edited.mp4", codec="libx264", bitrate="10000000")

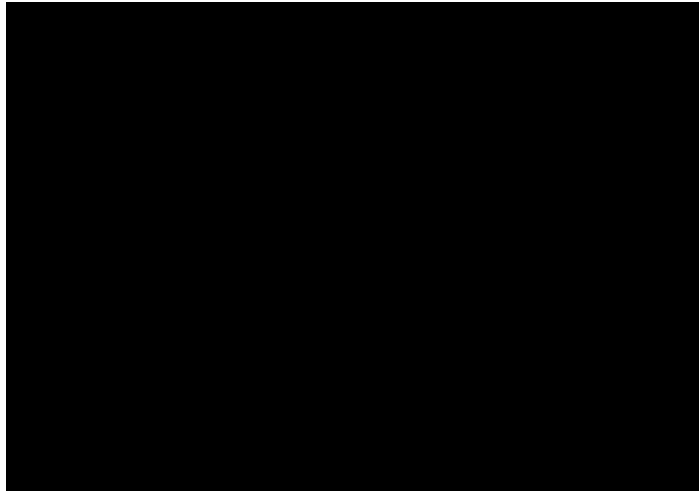
```

输出结果如下图所示：



案例三：MoviePy作者官方实现电影字幕

- https://zulko.github.io/moviepy/examples/star_worms.html



```
import numpy as np

from moviepy.editor import *
from moviepy.video.tools.segmenting import findObjects

# WE CREATE THE TEXT THAT IS GOING TO MOVE, WE CENTER IT.
screensize = (720,460)
txtClip = TextClip('Cool effect',color='white', font="Amiri-Bold",
                    kerning = 5, fontsize=100)
cvc = CompositeVideoClip( [txtClip.set_pos('center')],
                           size=screensize)

# THE NEXT FOUR FUNCTIONS DEFINE FOUR WAYS OF MOVING THE LETTERS

# helper function
rotMatrix = lambda a: np.array( [[np.cos(a),np.sin(a)],
                                  [-np.sin(a),np.cos(a)]] )

def vortex(screenpos,i,nletters):
    d = lambda t : 1.0/(0.3+t**8) #damping
    a = i*np.pi/ nletters # angle of the movement
    v = rotMatrix(a).dot([-1,0])
    if i%2 : v[1] = -v[1]
    return lambda t: screenpos+400*d(t)*rotMatrix(0.5*d(t)*a).dot(v)

def cascade(screenpos,i,nletters):
    v = np.array([0,-1])
    d = lambda t : 1 if t<0 else abs(np.sinc(t)/(1+t**4))
    return lambda t: screenpos+v*400*d(t-0.15*i)

def arrive(screenpos,i,nletters):
    v = np.array([-1,0])
    d = lambda t : max(0, 3-3*t)
    return lambda t: screenpos-400*v*d(t-0.2*i)

def vortexout(screenpos,i,nletters):
    d = lambda t : max(0,t) #damping
    a = i*np.pi/ nletters # angle of the movement
    v = rotMatrix(a).dot([-1,0])
    if i%2 : v[1] = -v[1]
    return lambda t: screenpos+400*d(t-0.1*i)*rotMatrix(-0.2*d(t)*a).dot(v)

# WE USE THE PLUGIN findObjects TO LOCATE AND SEPARATE EACH LETTER
letters = findObjects(cvc) # a list of ImageClips
```

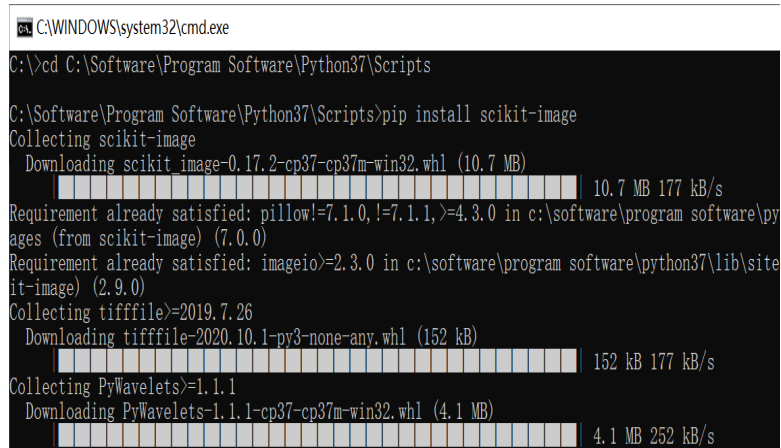


```
# WE ANIMATE THE LETTERS
def moveLetters(letters, funcpos):
    return [ letter.set_pos(funcpos(letter.screenpos,i,Len(letters)))
            for i,letter in enumerate(letters)]

clips = [ CompositeVideoClip( moveLetters(letters,funcpos),
                             size = screensize).subclip(0,5)
          for funcpos in [vortex, cascade, arrive, vortexout] ]

# WE CONCATENATE EVERYTHING AND WRITE TO A FILE
final_clip = concatenate_videoclips(clips)
final_clip.write_videofile('coolTextEffects.avi',fps=25,codec='mpeg4')
```

pip install scikit-image安装相关扩展包。



```
C:\WINDOWS\system32\cmd.exe
C:\>cd C:\Software\Program Software\Python37\Scripts
C:\Software\Program Software\Python37\Scripts>pip install scikit-image
Collecting scikit-image
  Downloading scikit_image-0.17.2-cp37-cp37m-win32.whl (10.7 MB)
    |#####| 10.7 MB 177 kB/s
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in c:\software\program software\python37\lib\site-packages (from scikit-image) (7.0.0)
Requirement already satisfied: imageio>=2.3.0 in c:\software\program software\python37\lib\site-packages (from scikit-image) (2.9.0)
Collecting tifffile>=2019.7.26
  Downloading tifffile-2020.10.1-py3-none-any.whl (152 kB)
    |#####| 152 kB 177 kB/s
Collecting PyWavelets>=1.1.1
  Downloading PyWavelets-1.1.1-cp37-cp37m-win32.whl (4.1 MB)
    |#####| 4.1 MB 252 kB/s
```






三.MoviePy抖音短视频剪切

我们通常在看抖音短视频过程中，通常会下载视频，但视频最后有个抖音的logo，我们能否用Python剪切视频去除logo，再合成集合呢？比如将某个UP主的猫视频自动处理成集合。



假设文件夹中有5个抖音短视频，如下图所示：

名称

 0a6478f1ed20b4998000b84121e77101.mp4
 0add5fd45762e7625d27c37ac46fc818.mp4
 0c88b9b471b792e4e629331807ad2594.mp4
 1c388dd38e731ad3ea5c5b82b1e7c2bc.mp4
 5d1ac362bdbb67bc8b751f29970f0d76.mp4

完整代码如下：

```

# -*- coding: utf-8 -*-
import os
from moviepy.editor import *

#递归获取文件名称
def file_name(file_dir):
    L=[]
    for root, dirs, files in os.walk(file_dir):
        for file in files:
            if os.path.splitext(file)[1] == '.mp4':
                L.append(os.path.join(root, file))
    return L

#主函数
if __name__ == '__main__':
    filePath = 'vedio'
    file_list = file_name(filePath)

    k = 1
    for name in file_list:
        print(name)
        #获取视频总时间
        video = VideoFileClip(name)
        times = video.duration
        print(times)

        #剪切视频广告 省略最后4秒
        video = VideoFileClip(name).subclip(0,times-4)
        result = "save" + str(k) + ".mp4"
        video.write_videofile(result)
        k = k + 1

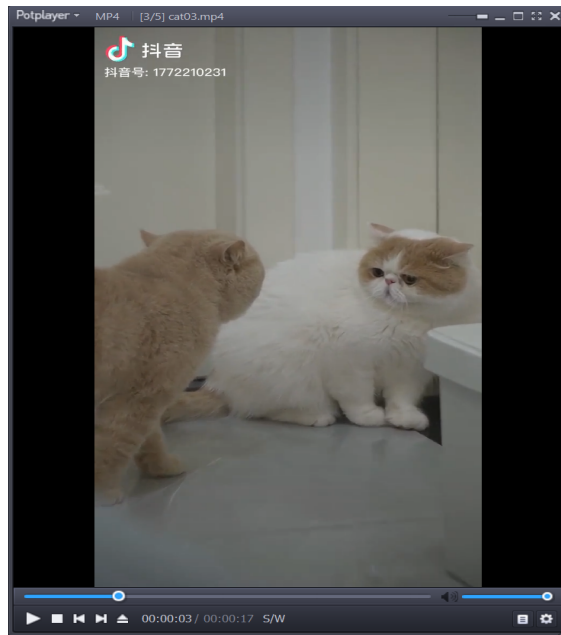
```

如果运行速度较慢，可以使用线程进行优化或ffmpeg开启GPU加速。输出结果如下图所示，成功将视频LOGO删除。

```

vedio\0a6478f1ed20b4998000b84121e77101. mp4
14.1
vedio\0add5fd45762e7625d27c37ac46fc818. mp4
18.57
vedio\0c88b9b471b792e4e629331807ad2594. mp4
18.13
vedio\1c388dd38e731ad3ea5c5b82b1e7c2bc. mp4
7.17
vedio\5d1ac362bdbb67bc8b751f29970f0d76. mp4
9.17

```



四.MoviePy抖音短视频合并

案例一：视频常规合并

接着将视频合并生成合集，先给出一个最简单的代码，如下所示。

```
# -*- coding: utf-8 -*-
import os
from moviepy.editor import *

#递归获取文件名称
def file_name(file_dir):
    L=[]
    for root, dirs, files in os.walk(file_dir):
        for file in files:
            if os.path.splitext(file)[1] == '.mp4':
                L.append(os.path.join(root, file))
    return L

#主函数
if __name__ == '__main__':
    filePath = 'vedio'
    file_list = file_name(filePath)

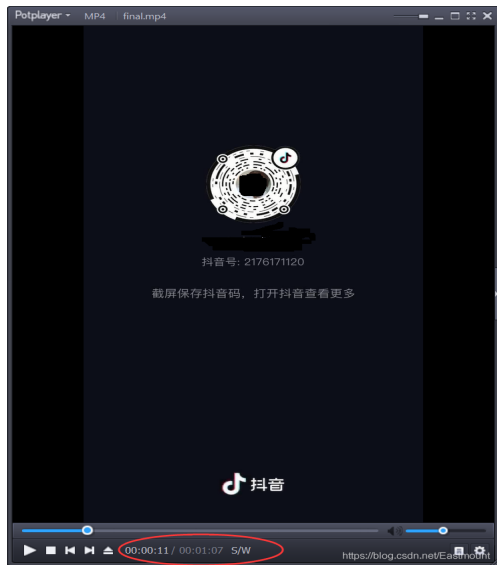
    k = 1
    L = []
    for name in file_list:
        print(name)
        #获取视频总时间
        video = VideoFileClip(name)
        times = video.duration
        L.append(video)
        print(times)

    #视频合成 帧数大致就是24
    final_clip = concatenate_videoclips(L)
    final_clip.to_videofile('./final.mp4', fps= 24, remove_temp=True)
```

运行结果如下图所示：

```
>>>
===== RESTART: C:\Users\xiuzhang\Desktop\blog29\test06.py =====
vedio\0a6478f1ed20b4998000b84121e77101.mp4
14.1
vedio\0add5fd45762e7625d27c37ac46fc818.mp4
18.57
vedio\0c88b9b471b792e4e629331807ad2594.mp4
18.13
vedio\1c388dd38e731ad3ea5c5b82b1e7c2bc.mp4
7.17
vedio\5d1ac362bdbb67bc8b751f29970f0d76.mp4
9.17
Moviepy - Building video ./final.mp4
MoviePy - Writing audio in finalTEMP MPY_wvf_snd.mp3
chunk: 0% | 0/1481 [00:00:00.00, 0.00it/s, now=None] chunk: 0% | 28/1481 [00:00:00:05:06, 4.82it/s, now=None] chunk: 2% | 68/1481 [00:00:02:26, 9.67it/s, now=None] chunk: 6% | 84/1481 [00:00:01:44, 13.35it/s, now=None] chunk: 8% | 125/1481 [00:01:00:46, 29.40it/s, now=None] chunk: 9% | 136/1481 [00:01:00:36, 36.48it/s, now=None] chunk: 10% | 153/1481 [00:01:00:28, 45.92it/s, now=None] chunk: 12% | 180/1481 [00:01:00:21, 61.13it/s, now=None] chunk: 14% | 200/1481 [00:01:00:16, 77.13it/s, now=None] chunk: 15% | 217/1481 [00:01:00:14, 85.51it/s, now=None] chunk: 17% | 258/1481 [00:02:00:10, 112.05it/s, now=None] chunk: 19% | 281/1481 [00:02:00:09, 130.98it/s, now=None] chunk: 21% | 313/1481 [00:02:00:07, 159.09it/s, now=None] chunk: 23% | 339/1481 [00:02:00:06, 179.93it/s, now=None] chunk: 25% | 365/1481 [00:02:00:05, 193.02it/s, now=None] chunk: 26% | 390/1481 [00:02:00:06, 176.43it/s, now=None] chunk: 28% | 412/1481 [00:02:00:08, 133.05it/s, now=None] chunk: 29% | 430/1481 [00:03:00:09, 113.14it/s, now=None] chunk: 30% | 445/1481 [00:03:00:11, 88.03it/s, now=None] chunk: 31% | 458/1481 [00:03:00:10, 94.59it/s, now=None] chunk: 32% | 470/1481 [00:03:00:10, 93.75it/s, now=None] chunk: 33% | 482/1481 [00:03:00:10, 91.35it/s, now=None] chunk: 33% | 493/1481 [00:03:00:11, 89.75it/s, now=None] chunk: 34% | 503/1481 [00:04:00:12, 77.26it/s, now=None] chunk: 35% | 512/1481 [00:04:00:25, 37.84it/s, now=None] chunk: 35% | 519/1481 [00:05:00:54, 17.65it/s, now=None] chunk: 35% | 524/1481 [00:05:00:47, 20.29it/s, now=None] chunk: 36% | 529/1481 [00:05:00:49, 19.19it/s, now=None] chunk: 36% | 533/1481 [00:06:00:44, 21.29it/s, now=None] chunk: 36%
```

其中 final.mp4 是最终生成的文件，fps是每秒钟传输的帧数，比如人眼一般一秒钟能看到的帧数大致就是 24，而程序最终会生成一个临时文件，remove_temp=True 指的就是将这个临时文件删除。



注意：在我的电脑上，如果一次性合成的文件数过多，程序会崩溃，并且速度是真的慢！！！花了24小时合并了5个小视频（1分7秒），泪奔~

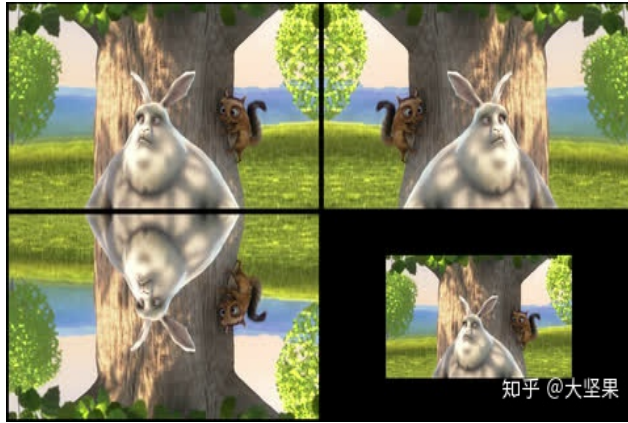
案例二：视频预处理合并

同时，补充知乎“大坚果”老师读取视频文件合并的核心代码。

```
clip1 = VideoFileClip("1.flv").subclip(10,20) #读取视频，并截取10-20秒的内容
clip2 = VideoFileClip("2.webm").resize(0.60) # 将视频画面尺寸缩小到60%
final_clip = concatenate_videoclips([clip1,clip2]) #视频合并
final_clip.write_videofile("hebing.mp4")
```

将多段视频以列表展现的形式显示

```
final_clip = clips_array([[clip1, clip2],[clip3, clip4]])
```



推荐CompositeVideoClips方法，可能比concatenate_videoclips、clips_array好用，它可选择起止位置，也可设置渐进切换。

```
video = CompositeVideoClip([clip1,                                #starts at t=0
                             clip2.set_start(5),                 #start at t=5s
                             clip3.set_start(9)])                #start at t=9s, fade-in 的形式转换

video = CompositeVideoClip([clip1,
                             clip2.set_pos((45,150)),
                             clip3.set_pos((90,100))])
```

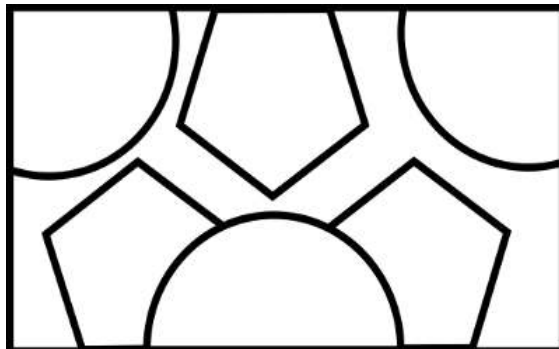
案例三：多视频按规定合并

该案例也是MoviePy官方提供，这里参考ucsheep大佬的代码供大家学习，推荐大家去学习。

- https://zulko.github.io/moviepy/examples/compo_from_image.html
- <https://blog.csdn.net/ucsheep/article/details/82787821>



它需要按照我们指定的图片进行投影，注意是PNG透明图片。



这个特殊的合成任务要花费很长的时间，这里仅给出完整代码。

```
from moviepy.editor import *
from moviepy.video.tools.segmenting import findObjects

# 加载用来指定区域的图像
im = ImageClip("../ultracompositing/motif.png")

# 加载这些区域返回一个ImageClip列表
regions = findObjects(im)

# 载入美国国家公园的7个clip
clips = [VideoFileClip(n, audio=False).subclip(18,22) for n in
    [ "../videos/romo_0004.mov",
      "../videos/apis-0001.mov",
      "../videos/romo_0001.mov",
      "../videos/elma_s0003.mov",
      "../videos/elma_s0002.mov",
      "../videos/calio-0007.mov",
      "../videos/grsm_0005.mov" ] ]

# 把每一个clip都放置在对应的图片中的区域
comp_clips = [c.resize(r.size)
               .set_mask(r.mask)
               .set_pos(r.screenpos)
               for c,r in zip(clips,regions)]

cc = CompositeVideoClip(comp_clips,im.size)
cc.resize(0.6).write_videofile("../composition.mp4")
```

五.总结

本篇文章主要讲解MoviePy视频编辑库基础知识，主要实现视频处理、视频剪切和视频拼接。但该扩展包也存在缺陷，速度非常慢。后续随着作者深入，希望能够分享更好的代码。希望这篇基础性文章对读者有一定帮助，也希望这些知识点为读者从事Python图像处理相关项目实践或科学研究提供一定基础。

2020年8月18新开的“娜璋AI安全之家”，主要围绕Python大数据分析、网络空间安全、人工智能、Web渗透及攻防技术进行讲解，同时分享CCF、SCI、南核北核论文的算法实现。娜璋之家会更加系统，并重构作者的所有文章，从零讲解Python和安全，写了近十年文章，真心想把自己所学所感所做分享出来，还请各位多多指教，真诚邀请您的关注！谢谢。



(By:Eastmount 2020-10-06 深夜10点夜于武汉 <http://blog.csdn.net/eastmount/>)

参考文献：

- [1] <https://pypi.org/project/moviepy/>
- [2] <https://github.com/Zulko/moviepy>
- [3] MoviePy - 中文文档(一个专业的python视频编辑库)教程
- [4] https://blog.csdn.net/weixin_42081389/article/details/104322629
- [5] <https://blog.csdn.net/SnailPace/article/details/107016442>
- [6] <https://zhuanlan.zhihu.com/p/46341173>
- [7] <https://www.jianshu.com/p/98a0c091c4bf>
- [8] <https://blog.csdn.net/mp624183768/article/details/81434408>
- [9] https://zulko.github.io/moviepy/examples/moving_letters.html
- [10] https://blog.csdn.net/weixin_43354181/article/details/104272789