

目录

1. 下载和安装

1.1 环境确认

1.2 安装 MoviePy

方法 1:pip 自动安装 (推荐)

方法 2:手动安装

1.3 环境介绍

2. 快速上手-MoviePy

2.1 快速预览

2.2 视频合成

Mixing clips (就是视频合成,

拼接混合)

视频叠加和视频拼接

CompositeVideoClips

开始和结束时间

Positioning clips

Compositing audio clips

2.3 Clips 变换与特效

moviepy 中的时间表示

改变 clip 属性的方法

clip.fx

创建用户自定义特效的方法

clip.fl

2.4 如何高效率使用 MoviePy

我们是不是应该使用

undefined undefined

很多预览 clip 的方式

clip.save_frame

clip.show and clip.preview

MoviePy – 中文文档 (一个专业的 python 音视频编辑库) 教程

作者: ucsheep

2021 年 3 月 24 日

本文字数: 25973 字

阅读完需
约 85
分钟

MoviePy 是一个用于视频编辑的 python 模块, 你可以用它实现一些基本的操作(比如视频剪辑, 视频拼接, 插入还可以实现视频合成, 还有视频处理, 抑或用它加入一些自定义的高级的特效。总之, 它的功能还是蛮丰富的。MoviePy 可以读写绝大多数常见的视频格式, 甚至包括 GIF 格式! 是不是很兴奋呢? 基于 Python2.7 以上的版本 MoviePy 可以轻松实现跨平台, Mac/Windows/Linux 统统没问题, 这也意味着, MoviePy 项目可以部署到服务端, 服务器进行视频处理。真是福音啊!

【PS:现在好多的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】

MoviePy 开发起来是很简便的, 来看一个操作例子 (在 IPython notebook 环境下开发)

```
In [12]: from moviepy.editor import *
# load the clip, rotate it 180°, and display
clip = VideoFileClip("lake.mp4").rotate(180)
clip.ipython_display(width=280)
```

Out[12]:



快速上手:

1. 下载和安装

其实还有更简单的办法, 让你快速使用 MoviePy, 告别繁琐的安装步骤, 如果你会使用 docker 的话, 直接跳到尾, 用 docker 使用 MoviePy

1.1 环境确认

首先, 确保当前系统中 Python 版本在 2.7 以上, pip 正常工作

```
wangyuxiangdeMacBook-Pro:~ ucsheep$ python -V
Python 2.7.10
wangyuxiangdeMacBook-Pro:~ ucsheep$ pip --version
pip 10.0.1 from /Library/Python/2.7/site-packages/pip-10.0.1-py2.7.egg/pip (pyth
on 2.7)
https://blog.csdn.net/ucsheep
```

对于 Mac 用户, 系统自带 python, 有可能大家会自己安装 python, 比如我是使用 python3.7, 系统中同时存在 p 7 和 3.7, 在下面的过程中, 如果使用 python3 来安装 moviepy 的话, 需要将命令 `python` 替换为 `python3`。

```
pip 10.0.1 from /usr/local/lib/python3.7/site-packages/pip-10.0.1-py3.7-pip_10.0.1.dist-info (python 3.7)
[wangyuxiangdeMacBook-Pro:~ ucsheep$ python3 -V
Python 3.7.0
[wangyuxiangdeMacBook-Pro:~ ucsheep$ pip3 --version
pip 10.0.1 from /usr/local/lib/python3.7/site-packages/pip (python 3.7)
```

1.2 安装 MoviePy

方法 1: pip 自动安装 (推荐)

```
1 pip install moviepy
```

复制

运行以上代码，理论上就安装好 MoviePy 了，如果你没有 `ez_setup` 的话，系统也会自动安装。

但是，如果同时没有 `ez_setup` 和 `setup_tools` 的话，会安装失败，这个时候莫慌！

依次执行下面的命令

```
1 python setup.py install
2 pip install moviepy
```

复制

此时的你应该已经完工嗑瓜子了

方法 2: 手动安装

你可以在后面任意一个地方下载源码。【[PyPI](#)】或【[Github](#)】

将所有源码解压到一个目录下，在该目录下命令行运用

```
1 python setup.py install
```

复制

1.3 环境介绍

MoviePy 依赖的 python 库有 `Numpy`, `imageio`, `Decorator`, 和 `tqdm` 这些都会在安装 MoviePy 的时候自 MoviePy 的视频读写依赖于我们大名鼎鼎的 `FFMPEG`, 你也无需担心，如果你之前没有安装过的话，`FFMPEG` `ImageIO` 安装的时候就被自动下载和安装了(当然，这有可能会话费一段时间)，有一点值得注意的是，`MoviePy` 又 `MPEG` 的版本要求是 `3.2.4`，一般自己安装会高于这个版本，所以会报错，我们可以参考源码文件 `moviepy/corrupts.py` 修改 `FFMPEG_BINARY` 属性，进而使用自定义的版本，如果害怕出错，让程序帮助你自动安装会是一个选择。

此外 MoviePy 是需要运行在 `Python2.7` 以上的；如果你在安装过程中遇到问题，可以联系我。

2. 快速上手-MoviePy

2.1 快速预览

快速了解

下面的内容介绍以下 MoviePy 适用于何种场景以及 MoviePy 如何工作。

我需要使用 MoviePy 吗？

出于以下的情景或原因，我们可能会有使用 Python 做视频编辑的需求。

我们有大量的视频需要处理，或者采用复杂的方式将他们拼接。

我们需要在服务端自动地创建大量视频或者 GIF 图。

我们需要在视频中创建视频编辑器中所没有的一些特殊的特效，我们只能敲代码来实现。

为其他 Python 库(例如：Matplotlib, Mayavi, Gizeh, scikit-images)生产的图片创建动画效果。

pleCV 这样专业的库去处理

我们仅仅是要将一段视频，或者一系列图片接进一个目标视频中时，我们使用 `ffmpeg` 就搞定了，也不用强行用 MoviePy

MoviePy 的优点与局限

MoviePy 在开发之初，就秉承着下面的理念

简单直观，基本操作一般一行代码搞定。对于初学者，代码很容易理解和学习。

灵活弹性，开发者拥有对视频或者音频中每一帧的全部控制权，这也使得我们在创建自定义效果时得心应手。

跨平台，使用的 `ffmpeg` 各个平台都有，可以移植到不同的平台运行。

MoviePy 的局限性如下：

不支持流媒体，它也确实不是为了处理这样的视频而设计的。

当同时使用太多(官网说>100)认为提阿多的视频，音频，或者图片的时候，我们有可能会遇到内存问题。内存亟待优化。

代码示例

在一个 MoviePy 脚本中，我们可以加载视频和音频，然后修改它们，将他们合并，然后把最终结果写入到一个新的文件中。下面的例子，加载视频，在视频中间添加一个标题显示 10 秒钟，然后把结果写入到一个新的文件内。

复制

```

1 # 导入需要的库
2 from moviepy.editor import *
3
4 # 从本地载入视频myHolidays.mp4并截取00:00:50 - 00:00:60部分
5 clip = VideoFileClip("myHolidays.mp4").subclip(50,60)
6
7 # 调低音频音量 (volume * 0.8)
8 clip = clip.volumex(0.8)
9
10 # 做一个txt clip. 自定义样式, 颜色。
11 txt_clip = TextClip("My Holidays 2013", fontsize=70, color='white')
12
13 # 文本clip在屏幕正中显示持续10秒
14 txt_clip = txt_clip.set_pos('center').set_duration(10)
15
16 # 把 text clip 的内容覆盖 video clip
17 video = CompositeVideoClip([clip, txt_clip])
18
19 # 把最后生成的视频导出到文件内
20 video.write_videofile("myHolidays_edited.webm")

```

MoviePy 如何工作

MoviePy 使用软件 `ffmpeg` 读取和导出视频和音频文件，使用 `ImageMagick` 生产文字和 GIF 图。这些处理过程于 Python 强大的数学处理库，高级特效和软件加强用到了许多的 Python 图像处理库。

基本概念

MoviePy 中最核心的对象就是 `clips`。`AudioClips` 或者 `VedioClips`，开发者可以对 clips 进行修改(剪切，调速度...)或者和其他 clip 混合拼接到一起。使用 PyGame 或者 IPython NoteBook 还可以预览。

`VedioClips` 可以由视频文件，图像，文本或者动画来创建实例。vedio clip 可以拥有一个音频轨道(audio clip)和加层的 vedio clip(这是一个特殊的 VedioClip，这意味着，当一个视频和其他 VedioClip 混合的时候，这个叠加层隐藏的)

一个 clip 可以被 MoviePy 中多种效果作用，比如`clip.resize(width="360")`, `clip.subclip(t1,t2)`, `ip.fx(vfx.black_white)`.

当然，也可以被用户自定义的效果作用。MoviePy 实现了许多类似 `(clip.fl, clip.fx)` 这样的方法，可以简便地修图。

在 `moviepy.video.tools` 里面，还可以找到许多好东西，实现很多高级功能，比如跟踪视频中的一个对象，画一的图形，斜线，或者颜色，制作副标题等等效果

最后，即使 moviepy 没有图形化的用户交互界面，但是在我高质量地加工我们的视频的时候，我们仍然有很多许很好地控制和调节脚本的预览方式。

不得不说，moviepy 还是不少缺陷的，最开始的大部分时候是 ffmpeg 引起的 OOM，是我分配的内存不够，加大分配，每次运行完毕之后清除 cache，杀掉已经存在的 ffmpeg，稳定运行时间达到了 8 个小时多，然而在昨晚凌晨掉了。现在还在收拾，今天再来更新一下。

Mixing clips (就是视频合成，拼接混合)

视频的排版，众所周知，这是一个非线性编辑的技术活，有时候还要把几段 clip 在一个新的 clip 里面一起播放。还有一点很重要，一般情况下，每一个 video clip 都带有一个 audio 和一个 mask，分别是 audio clip 和 mask 俗语地讲就是每一个视频切片带有一个音频切片和一个蒙版切片。所以，当我们把几个视频排版混合到一起的时候轨道和 mask 也同时被混合在一起形成最终的 clip。所以但多数时候我们是不需要操心声音和 mask 的。

视频叠加和视频拼接

简单地把多个视频合成到一起的两种最简单的办法。1. 把视频一个接一个地拼接起来。2. 视频叠加在一块，比如的画面同时播几个视频。

视频拼接我们使用 `concatenate_videoclips` 函数来完成。

```
1 from moviepy.editor import VideoFileClip, concatenate_videoclips
2
3 clip1 = VideoFileClip("myvideo.mp4")
4 clip2 = VideoFileClip("myvideo2.mp4").subclip(50,60)
5 clip3 = VideoFileClip("myvideo3.mp4")
6
7 finalclip = concatenate_videoclips([clip1, clip2, clip3])
8 finalclip.write_videofile("my_concatenate.mp4")
```

复制

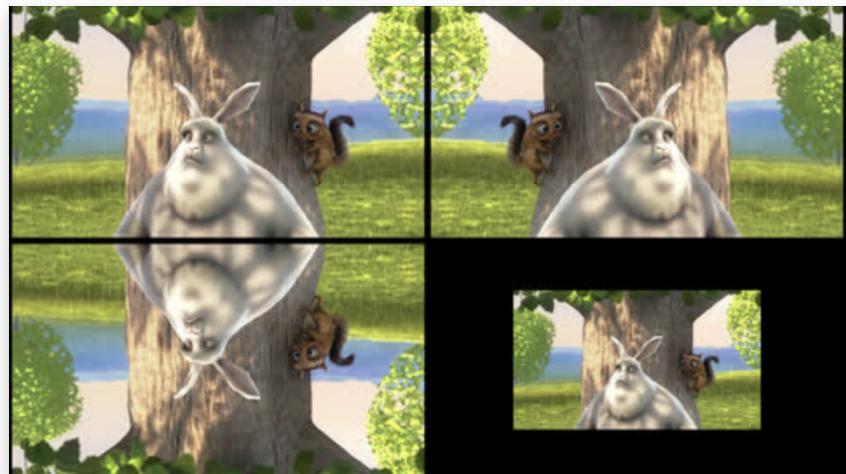
`finalclip` 会按照 `clip1`, `clip2`, `clip3` 的顺序将这三个 clip 播放。这些 clip 并不需要相同的时长或者大小，仅仅是连而已。我们还可以通过 `transition=my_clip` 这个参数来设置一下 clip 之间衔接的过渡动画。

视频叠加我们使用 `clip_array` 函数来完成

```
1 from moviepy.editor import VideoFileClip, clips_array, vfx
2
3 clip1 = VideoFileClip("myvideo.mp4").margin(10)
4 clip2 = clip1.fx(vfx.mirror_x)#x轴镜像
5 clip3 = clip1.fx(vfx.mirror_y)#y轴镜像
6 clip4 = clip1.resize(0.6)#尺寸等比缩放0.6
7
8 final_clip = clips_array([
9             [clip1, clip2],
10            [clip3, clip4]
11        ])
12 final_clip.resize(width=480).write_videofile("my_stack.mp4")
```

复制

举个栗子，大家得到画面差不多和这个相似



CompositeVideoClip 这个类提供来更加灵活的方式来排版视频，但是它可要比 concatenate_videoclips 和 clips 要复杂的多了。

复制

```
1 video = CompositeVideoClip([clip1, clip2, clip3])
```

现在，这 3 个 clip 是堆叠起来的，啥情况呢，就是，clip2 压在 clip1 上，clip3 压在最上面。大家可以想像成栈：从左往右，依次入栈这么堆叠在一块，对于一个普通的视频来说，这样堆叠实在是没办法看了。你想像一下，如的尺寸是最大的，比 clip1 和 clip2 都大，那我们只能看到 clip3，clip1 和 clip2 都被盖住了。最终要合成的 clip 是默认第一个 clip 的尺寸，一般第一个也就是做背景用了，但是我们有时候是需要我们的 clip 浮在一个比较大的上方的，那这个时候我们就需要声明合成 clip 的尺寸。看看下面的栗子。

复制

```
1 video = CompositeVideoClip([clip1, clip2, clip3], size=(720,480))
```

开始和结束时间

在堆叠视频中，每个 clip 会在通过 clip.start 函数声明的时间开始播放，我们可以像下面这样去设置。

复制

```
1 clip1 = clip1.set_start(5) #在5秒时开始
```

所以一般情况下，我们代码中的视频堆叠都长这个样子。

复制

```
1 video = CompositeVideoClip([
2                     clip1, #在第0秒开始
3                     clip2.set_start(5), #在第5秒开始
4                     clip3.set_start(9) #第9秒开始
5                 ])
```

在下面的栗子里，可能 clip2 开始的时候，正好是 clip1 将要结束的时机，这样的情况，我们可以让 clip2 使用在“渐入”的特效来显示。

复制

```
1 video = CompositeVideoClip([
2                     clip1,
3                     clip2.set_start(5).crossfadein(1),
4                     clip3.set_start(9).crossfadein(1.5)
5                 ])
```

Positioning clips

clip 们的位置设定。

clip2 和 clip3 都比 clip1 要小，那你可以通过设定位置决定 clip2 和 clip3 在画面中的位。下面的栗子就是通过指的形式（距离左上方的像素距离）把 clip2 和 clip3 在画面中指定位置。

复制

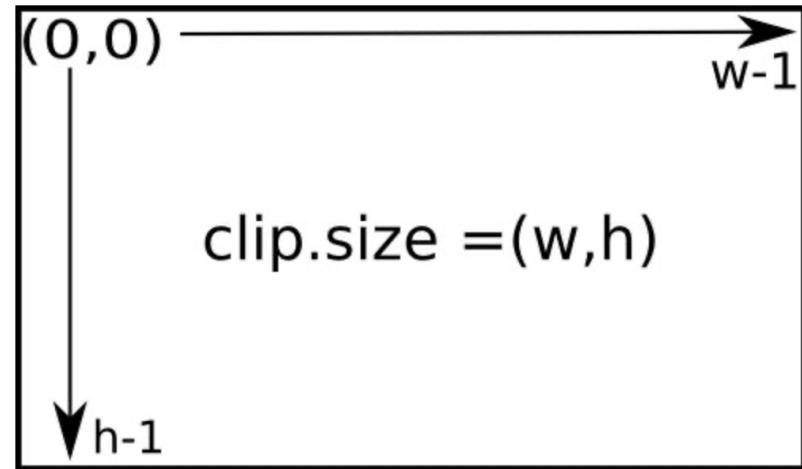
```
1 video = CompositeVideoClip([
2                     clip1,
3                     clip2.set_pos((45,150)),
4                     clip3.set_pos((90,100))
5                 ])
```

在 moviepy 中我们有很多的方法定位 clip 的位置

复制

```
1 clip2.set_pos((45,150)) #像素坐标
2 clip2.set_pos("center") #居中
3 clip2.set_pos(("center","top")) #水平方向居中，但是处置方向放置在顶部
```

clip 位置的坐标系，在指定位置时，出现在我们脑海里的坐标系应该是这样子的



Compositing audio clips

合成声音 clips

在混合 vedio clips 时，MoviePy 将会自动地把这些 vedio 自带的 audio clip 按照一定形式混合在一起形成最终的开发者不需要为了这个操心。

如果你有一些特殊的定制合成音频的需求，应该使用 CompositeAudioClip 和 concatenate_audioclips 这俩类

```
1 from moviepy.editor imports *
2
3 concat = concatenate_audioclips([clip1, clip2, clip3])
4 compo = CompositeAudioClip([
5     aclip1.volumex(1.2),
6     aclip2.set_start(5), # start at t=5s
7     aclip3.set_start(9)
8 ])
```

2.3 Clips 变换与特效

下面是 MoviePy 中的几种对 clip 的修改：

非常常见的修改 clip 属性的方法有：`clip.setduration`, `clip.setaudio`, `clip.setmask`, `clip.setstart` 等。

已经实现的特效.`clip.subclip(t1,t2)`:截取 t1 到 t2 时间段内的片段；还有一些高级效果，loop：让 clip 循环播放。`memirror`:让 clip 倒播，这些方法位于特殊的模块 `moviepy.video.fx`, `moviepy.audio.fx`, 应用 `clip.fx` 方法，`ip.fx(timemirror)` 让视频倒播。

可你自己创造一些需要的特效。

以上的特效其实本质上并不是原地直接修改的（没有对原始视频修改），而是根据修改产生新的 clip。所以，我想让修改生效，需要将修改过的产生的 clip 赋值给某 clip，保存修改。举个栗子。

```
1 my_clip = VideoFileClip("some_file.mp4")
2 my_clip.set_start(t=5) #没有做任何改变，修改会丢失
3 my_new_clip = my_clip.set_start(t=5) #这样才对。moviepy中，修改过的clip要重新赋值给变量，修改才会被保存
```

所以，当我们写出 `clip.resize(width=640)`, moviepy 并不是立刻就逐帧修改 clip。一般只会先修改第一帧，其他的帧只有在需要的时候(最后写入文件或者预览)才会被 resize。

另一方面，可以这样讲，创建一个 clip，几乎是不会占用时间和内存的，几乎所有的计算其实发生在最后转换的

很多方法都接受时间参数，`clip.subclip(tstart,tend)`，截取两个时间点之间的 clip 片段，在这个方法中，时间即(`tstart=230.54`)，以秒的时间来表示，也可以用(`tstart=(3,50.54)`)，以 3 分 50.54 秒的形式来表示，还可以 (`tstart(0.54)`)或者 (`tstart=(00:03:50.54)`)，以，小时，分钟，秒的形式来表示。

大多数没有赋值的时间参数会有一个默认值，比如 `clip.subclip(tstart=50)`，tend 的默认值就是视频的长度，`clip(tend=50)`，那么 tstart 就默认为 0.当时间是负数的时候，代表倒数 n 秒。比如，`clip.subclip(-20, -10)`会截取秒到倒数 10 秒之间的片段。

改变 clip 属性的方法

clip.fx

假定我们很多修改 clip 的方法。这些方法呢，都是输入一个 clip 和一些参数，输出一个新的 clip。

```
1 effect_1(clip, args1) -> new clip
2 effect_2(clip, args2) -> new clip
3 effect_3(clip, args3) -> new clip
```

如果我们要按顺序，依次修改 clip，那么你可能会这样写：

```
1 newclip = effect_3( effect_2( effect_1(clip, args3), args2), args1)
```

但是上面的代码可读性不高，我们可以利用 clip.fx 来实现一种更简练的写法。

```
1 newclip = (clip.fx( effect_1, args1)
2                 .fx( effect_2, args2)
3                 .fx( effect_3, args3))
```

看上去是不是好多了。在模块 `moviepy.video.fx` 和 `moviepy.audio.fx` 中已经实现来一些修改 clip 的方法，这些方法会自动的作用到和 clip 相关的声音和 mask 上，所以我们在修改 clip 的时候并不需要去关心声音和 mask 除非我们确实要对声音或者 mask 做一些特殊的处理。

在实际应用中，当我们使用 `from moviepy import.editor *` 的时候，这两个模块会被加载为 `vfx` 和 `afx`，所以我们下面这样写

```
1 from moviepy.editor import *
2 clip = (VideoFileClip("myvideo.avi")
3         .fx( vfx.resize, width=460) # 尺寸变化，保持纵横比
4         .fx( vfx.speedx, 2) # 2倍速
5         .fx( vfx.colorx, 0.5)) # 画面调暗
```

为方便起见，当我们使用 `moviepy.editor` 的时候，比如我们使用 `resize` 的时候，我们经常会使用 `clip.resize(..)` 简便的写法来代替 `clip.fx(vfx.resize, ...)` 的写法。

创建用户自定义特效的方法

clip.fl

我们可以使用 `clip.fl_time`、`clip.fl_image` 或者更普遍的 `clip.fl` 来修改一个 clip。

我们可以使用 `clip.fl_time` 修改 clip 的时间线，像下面这样

```
1 modifiedClip1 = my_clip.fl_time(lambda t: 3*t)
2 modifiedClip2 = my_clip.fl_time(lambda t: 1+sin(t))
```

中，我们已经创建了一个无限持续的 clip（现在并不会出什么问题）。

我们也可以用 `clip.fl_image` 来修改一个 clip 的显示。接下来，将会把一个 clip 每一帧的绿色和蓝色通道转换。

```
1 def invert_green_blue(image):
2     return image[:, :, [0, 2, 1]]
3 modifiedClip = my_clip.fl_image(invert_green_blue)
```

接下来，我没还有可能需要同时对时间和每一帧的画面都做处理，我们用 `clip.fl(filter)` 方法也是可以实现的，这个必须是一个传入两个参数，返回一个画面的函数。第一个参数是 `get_frame` 方法（例如：`g(t)` 给定一个时刻，然后那一时刻这个 clip 中对应的一帧），第二个参数就是时间。

```
1 def scroll(get_frame, t):
2     ...
3     这个函数返回：当前帧的一个‘区域’
4     上述所说的‘区域’的位置依赖于第二个参数，时间
5     ...
6     frame = get_frame(t)
7     frame_region = frame[int(t):int(t)+360,:]
8     return frame_region
9 modifiedClip = my_clip.fl(scroll)
```

以上代码，将会向下滚动 360 个像素高度。

当我们在编写一些新的效果的时候，应该尽可能地使用 `fl_time` 和 `fl_image`，而不是 `fl`。原因在于，在这些效果于 `ImageClip` 的时候，MoviePy 会认为这些方法不需要应用于每一帧，渲染速度会更快。

2.4 如何高效率使用 MoviePy

接下来，这一节将会介绍许多全世界 MoviePy 开发者总结的开发经验。

开始学习使用 MoviePy 最好的方式是使用 IPython Notebook：它可以让预览 clip 变得简单，有自动补全的功能可以显示函数库中不同方法的文档。

我们是不是应该使用 `moviepy.editor`？

这个文档中的大部分例子都会用到子模块 `moviepy.editor`，但是这个子模块并不适用于所有需求。所以，我们应他吗？

简短回答：如果我们是使用 `moviepy` 手动地编辑视频，那就使用他。但是，如果我们使用的是 MoviePy 内置的库、程序、或者网络服务，最好还是避免它，只需要加载我们需要的方法就够了。

`moviepy.editor` 模块可以通过下列的 3 种导入方法的任意一种导入

```
1 from moviepy.editor import *      #全部导入，快速，但是不干净
2 import moviepy.editor as mpy    #干净，使用示例：mpy.VideoClip
3 from moviepy.editor import VideoFileClip    #只导入需要的
```

无论使用上面哪一种导入方式，`moviepy.editor` 模块事实上都会在幕后做大量的工作：加载很多 `moviepy` 常用的方法、还有子包、以及如果安装了 PyGame 的话还会加载 PyGame session，这样，才可以预览 video clips，实现简便的调用（比如：将一个 `clip` 用 `resize` 变化）。这样，我们可以使用 `clip.resize(width=240)` 而不是更长的 `fx(resize, width=240)`。简短说，`moviepy.editor` 提供给我们在各处播放和编辑所需要的所有的東西，但是这也有一部分时间用于加载（大于 1 秒钟）。所以，如果我们需要的仅仅是 library 中一两个特性的话，最好使用的导入是：只导入自己需要的，像下面这样

很多预览 clip 的方式

当我们使用 MoviePy 编辑视频或者实现一个效果的时候，我们会进行一些试验、和错误的过程。通常情况我每次试验都会话费相对较长的时间，这一节，会告诉大家很多的小窍门，让这些尝试的过程变得快一点。

clip.save_frame

大多情况下，我们只需要得到一帧画面就可以知道程序有没有正确执行，这样，我们只需要保存一帧画面到一个就可以去验证，像下面这样：

```
1 my_clip.save_frame("frame.jpeg")      #保存第一帧
2 my_clip.save_frame("frame.png", t=2)    #保存2s时刻的那一帧
```

复

clip.show and clip.preview

clip.show 和 **clip.preview** 这两个方法可以在 Pygame 的窗口中展示。这是最快的预览方式，clip 的生成和显示同时发生的，而且对于获取一个像素点的颜色和坐标也有用。完成后按 **Esc** 键退出。

一个 clip 可以按照下面的方式来预览

```
1 my_clip.preview() # 默认fps=15 预览
2 my_clip.preview(fps=25)
3 my_clip.preview(fps=15, audio=False) # 不生成/播放 音频
4 my_audio_clip.preview(fps=22000)
```

复

假如你在预览的画面中点击了某些地方，将会输出被点击像素的位置坐标和颜色。按 **Esc** 可以终端预览。

如果当 clip 十分复杂而且电脑配置很渣，运行很慢的时候，预览一般会出现比真实画面速度慢的情况，这个时候可以尝试降低帧率，或者使用 **clip.resize** 减小 clip 的尺寸。

ipython_display

如果不使用 **clip.show()** 或者 **clip.preview()**，使用 IPython Notebook 显示 clip 也是一个不错的选择。使用的画面一样的。



通过 **ipython_display**，我们可以将图片、视频、音频等嵌入，既可以来自于一个文件，也可以来自于一个 clip

```
1 ipython_display(my_video_clip)
2 ipython_display(my_image_clip)
3 ipython_display(my_audio_clip)
```

复

只有当 rendering 在 notebook 的最后一行时才有效。我们也可以把 ipython_display 作为一个 clip 方法调用

```
1 my_video_clip.ipython_display()
```

复制

如果我们的 clip 需要一个帧率，我们可以指定 **ipython_display** 的 fps=25。

如果我们仅仅需要显示一下视频 clip 的快照，那么，我们直接指定 time 就行

```
1 my_video_clip.ipython_display(t=15) #将会显示15s处的快照
```

复制

我们也可以提供任何的 HTML5 选项作为关键词参数，例如：当一个 clip 特别大的时候，我们一般会这么写

```
1 ipython_display(my_clip, width=400) # HTML5 将会把他的尺寸变成400像素
```

复制

当我们编辑一个 GIF 格式的动画，并且确保他有完好的循环。我们可以让视频自动开始，并且循环播放。

```
1 ipython_display(my_clip, autoplay=1, loop=1)
```

复制

注意：事实上，**ipython_display** 是将 clip 完全地入我们的 notebook。这样做的优点是无论我们移动 notebook 到线上，视频都会正常的播放。缺点在于，nootbook 文件的大小将会变得很大。取决于你的浏览器，多次重新显示视频不光是花费很多的时间，也会占用特别多的内存（RAM），重新启动浏览器会解决这个问题。

2.5 使用 matplotlib (一个 2D 绘图库)

用户自定义动画

moviepy 允许开发者自定义动画：定义一个方法，以 numpy 数组的形式在动画中给定的时间返回一帧画面。一个例子如下：

```
1 from moviepy.editor import VideoClip
2
3 def make_frame(t):
4     ...
5     返回在t时刻的一帧画面
6     ...
7     # 通过其他的任意第三方库，创建一帧画面
8     return frame_from_time_t #(height * width * 3) Numpy array
9 animation = VideoClip(make_frame, duration=3)
```

复制

这样的 animation，通常可以按照 moviepy 中的方式导出，如下：

```
1 # 导出为一个视频
2 animation.write_videofile("my_animation.mp4", fps=24)
3 # 导出为一个GIF动图
4 animation.write_gif("my_animation.gif", fps=24) # 一般情况，这种方式会慢点
```

复制

简单的 matplotlib 示例

一个使用 matplotlib 操作动画的例子，如下：

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from moviepy.editor import VideoClip
```

复制

```

8 duration = 2
9
10 fig,ax = plt.subplots()
11 def make_frame(t):
12     ax.clear()
13     ax.plot(x, sinc(x**2) + np.sin(x + 2*np.pi/duration * t), lw=3)
14     ax.set_ylim(-1.5, 2.5)
15     return mplfig_to_npimage(fig)
16
17 animation = VideoClip(make_frame, duration=duration)

```

使用 Jupyter Notebook

如果我们在 Jupyter Notebook 写代码的话，我们就可以享受到使用 `ipython_display` 方法将 `VideoClips` 嵌入 notebook 的 `output` 部分。下面就是一个实现案例：

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from moviepy.editor import VideoClip
4 from moviepy.video.io.bindings import mplfig_to_npimage
5
6 x = np.linspace(-2, 2, 200)
7
8 duration = 2
9
10 fig, ax = plt.subplots()
11
12 def make_frame(t):
13     ax.clear()
14     ax.plot(x, np.sinc(x**2) + np.sin(x + 2*np.pi/duration * t), lw=3)
15     ax.set_ylim(-1.5, 2.5)
16     return mplfig_to_npimage(fig)
17
18 animation = VideoClip(make_frame, duration=duration)
19 animation.ipython_display(fps=20, loop=True, autoplay=True)

```

2.6 moviepy 中的音频

下面主要是演示在 moviepy 中如何创建和编辑 audio clips。

之前曾经说过，在 moviepy 中，当我们剪切，混合，拼接 video clip 的时候，audio clip 并不需要我们去操心，自动的随着 video 完成相应的处理。这篇博文提到的对于 audio clip 的操作主要是为了开发者的两种情况：首先，我们有兴趣只对音频做处理，其次就是我们希望为 video 配自定义的音频。

什么是 audio clips

AudioClips 和 moviepy 中 VideoClips 的概念很像：audio clip 有长度，像 video clip 一样可以被剪切，组合等等显著的区别在于 `audioclip.get_frame(t)`。

创建一个新的 audio clip

audio clip 创建可以有两种方式，一种是从一个文件直接创建 autio clip，第二种办法呢，就是从一个视频文件中提取出音频的部分。

```

1 from moviepy.editor import *
2 audioclip = AudioFileClip("some_audiofile.mp3")
3 audioclip = AudioFileClip("some_video.avi")

```

另外，也可以从已经创建的 `VideoClip` 对象中直接获取 audio clip

合成音频 (待续)

导出和预览 audio clips

我们可以将一个 audio clip 分配到一个 vedio clip 声音轨道上

```
1 videoclip2 = videoclip.set_audio(my_audioclip)
```

2.7 创建和导出 vedio clips

video 和 audio clips 是 moviepy 中的核心的对象。这篇博文，我们会介绍不同的短 clip，展示如何创建他们，以将它们导出到文件中。

接下来的代码是总结好的我们可以通过 moviepy 创建的一些基本的 clips。

```
1 # VIDEO CLIPS
2 clip = VideoClip(make_frame, duration=4) # 自定义动画
3 clip = VideoFileClip("my_vedio_file.mp4") # 文件格式还可以是avi、webm、gif等
4 clip = ImageSequenceClip(['imagefile.jpeg', ... ], fps=24)
5 clip = ImageClip('my_picture.png') # 文件格式还可以是 png、tiff等
6 clip = TextClip('Hello!', font="Amiri-Bold", fontsize=70, color='black')
7 clip = ColorClip(size=(460, 380), color=[R,G,B])
8
9 # AUDIO CLIPS
10 clip = AudioFileClip("my_audio_file.mp3") #文件格式还可以是ogg、wav或者也可以是一个vedio
11 clip = AudioArrayClip(numpy_array, fps=44100) # 一个numpy数组
12 clip = AudioClip(make_frame, duration=3) # 使用一个方法make_frame(t)
```

video clip 的分类

video clip 好比是较长的 video 的积木块。从技术上讲，这些 clip 都是使用 `clip.get_frame(t)` 方法，在某一时间的 HxWx3 numpy 格式的剪辑帧的数组。主要有两种分类，首先是动画形式的 clip（包括 VideoClip、VideoFile）其次是非动画形式的 clip，循环显示相同的画面（包括 ImageClip、TextClip、ColorClip 都在此列）。还有一种 clip，那就是 **masks**，它属于上面的类别，但是他输出的灰色帧表示在另一个剪辑中哪些部分是可见的还有哪些不可见的。

VideoClip

VideoClip 是 moviepy 中其他所有的 vedio clips 的父类。如果你的需求仅仅是编辑视频文件，你是不会用到他的。上，一般只有在我们通过一些第三方的库生成动画的时候才会用到这个类。我们所需要做的，就是定义一个 `make_frame(t)` 函数，这个函数在一个给定的时刻 t 返回 HxWx3 形式的 numpy 数组。下面是一个使用第三方图像库 Gizeh 子：

```
1 import gizeh
2 import moviepy.editor as mpy
3
4 def make_frame(t):
5     surface = gizeh.Surface(128, 128) # 宽、高
6     radius = W*(1+ (t*(2-t))**2 )/6 # 半径随时间变化
7     circle = gizeh.circle(radius, xy=(64,64), fill(1,0,0))
8     circle.draw(surface)
9     return surface.getnpimage() # 返回一个8bit RGB数组
10
11 clip = mpy.VideoClip(make_frame, duration=2)
12 clip.write_gif('circle.gif', fps=15)
```

VideoFileClip

VideoFileClip 就是从视频文件（大部分的视频格式都支持）或者 GIF 格式文件读取生成的 clip。我们可以像下面加载一个 video。

```
1 myclip = VideoFileClip("some_video.avi")
2 myclip = VideoFileClip("some_animation,gif")
```

每一个 clip 都有一个 fps 参数，当我们使用 `write_videofile`, `write_gif` 等去将一个 clip 切分成小的 clip 的时候，原有的 clip 的 fps 会默认地传递给新产生的 clip。

```
1
2 myclip = VideoFileClip("some_video.avi")
3 print(myclip.fps) # 输出为30
4 # 剪辑这个clip 的10s-25s, 这会保存fps
5 myclip2 = myclip.subclip(10, 25)
6 myclip2.write_gif("test.gif") # 这个gif文件的fps=30
```

ImageSequenceClip

顾名思义，就是一系列图片创建的 clip，你可以像下面这样调用：

```
1 clip = ImageSequenceClip(images_list, fps=25)
```

`images_list` 可以有三种形式：

1. 图片 name list, 将会按照 list 顺序播放
2. 一个目录，这个目录下面的所有图片都会被播放，按照字母或者数字的顺序播放
3. 一系列 frames (NumPy arrays)，从其他的 clip 剪辑得到的，顺序是按照 list 顺序播放

当我们使用第一种或者第二种的时候，我们可以通过选择 `load_images=True` 来指定是否将所有的图片直接加载 M 中。当然，这样的操作仅仅是在我们的图片不是很多，且每张图片都被使用超过 1 次的时候。

ImageClip

ImageClip 就是一个一直播同一张图片的 vdeo clip，我们可以按照下面的方式创建：

```
1 myclip = ImageClip("some_pic.jpeg")
2 myclip = ImageClip(some_array) # 一个 (高 × 宽 × 3) RGB numpy 数组
3 myclip = some_video_clip.to_ImageClip(t = '01:00:00') # 在1h时刻的帧画面
```

TextClip

生成一个 TextClip 需要安装 `ImageMagick`，并且和 moviepy 关联。

下面是一个创建 TextClip 的案例

```
1 myclip = TextClip('Hello!', font='Amiri-Bold')
```

注意：这里的 font 可以是任何安装在你的电脑上的 font，如果照抄这里的代码，有可能会有问题，因为你的电脑有这个样式。那么如何查看当前设备可用的样式呢？

```
1 TextClip.list('fonts') # 可以获得当前设备所有可用的fonts
```

查找所有给定字体相关的所有字体，使用下面的方法：

```
1 TextClip.search('Amiri','fonts') # 查找所有包含Amiri的字体
```

注意：使用笔划或者（轮廓）的话，对于较小的字母来说不会有很好的结果。如果我们需要一个具有轮廓的 p，我们最好是先生成一个大点的文字，然后再将它缩小，这样的效果才会好。

```
1 myclip = TextClip('Hello!', fontsize=70, stroke_width=5).resize(height=15)
```

TextClip 是有很多很多的可选项的，对齐、字距（字母距离）、笔划大小、背景、字体环绕等等。如果需要详细信息，可以去官网参考手册去找或者等待后续中文文档更新。

Mask clips

mask clip 是一种特殊的 video clip，当一个带有 mask 的 video clip -- A 和其他的 video clips 合成的时候，可以表示哪些像素是可见的，哪些是不显示的。在导出 GIF 或者 PNG 文件的时候，mask 也可以用来定义透明度。

mask clip 和普通的 video clip 最基本的不同在于：

标准的 clip 的每一帧中的每一个像素是由 3 个色元即 R-G-B 组合成的，取值范围是 0-255。（0-255, 0-255, 0-255）

mask clip 每个像素仅一个色元组件，取值范围只有 0-1。要么着一个像素可见，要么这个像素透明。另一方面看，mask 只有灰色的通道。

当我们创建或者加载一个带有 mask 的 clip 的时候，我们会像下面这样声明：

```
1 maskclip = VideoClip(makeframe, duration=4, ismask=True)
2 maskclip = ImageClip('my_mask.jpeg', ismask=True)
3 maskclip = VideoFileClip("myvideo.mp4", ismask=True)
```

将一个 mask clip 覆盖在一个和他同样尺寸的 clip 时使用

```
1 myclip.set_mask(mask_clip)
```

一些通过一个 **alpha** 层来支持透明度的图片格式，比如 PNG，moviepy 中应该使用 mask 来对应：

```
1 myclip = ImageClip("image.png", transparent=True) # 默认为True
2 myclip.mask # 图片的alpha层
```

任何一个 video clip 都可以通过 **clip.to_mask()** 转换为一个 mask，任何一个 mask 也可以通过 **my_mask_clip.to()** 转换为标准的 RGB video clip.

导出 video clips

video 文件 (.mp4 , .webm, .ogv...)

将一个 clip 写入一个文件

```
1 myclip.write_videofile('movie.mp4') # 默认编码 'libx264', 24 fps
2 myclip.write_videofile('movie.mp4', fps=15)
3 myclip.write_videofile('movie.webm')
4 myclip.write_videofile('movie.webm', audio=False) # 不使用音频
```

特率，是否将音频也导出，文件大小优化，使用处理器个数，等等)

有些时候，我们使用 moviepy 是无法获得一个 clip 的 duration 这个值的。然后，duration 这个必须通过 clip.set_duration() 来指定：

```
1 # 创建一个clip, 显示一个花的图片5秒钟
2 my_clip = Image('flower.jpeg') # 现在是无限时间, 不停地循环
3 my_clip.write_videofile('flower.mp4') # 这一步将会出错, 因为没有指定duration
4 my_clip.set_duration(5).write_videofile('flower.mp4') # 正确执行
```

GIF 动画

将 video clip 导出为一个 gif 动画

```
1 my_clip.write_gif('test.gif', fps=12)
```

注意：需要安装 ImageMagick。此外，我们还可以通过添加选项 **program='ffmpeg'** 来使用 ffmpeg 生成 gif 动画。ffmpeg 生成 gif 动图，虽然要快一点，但是看起来效果可能会不太好，而且没有办法去优化。

导出图片

我们可以将某一帧导出为图片

```
1 myclip.save_frame('frame.png') # 默认保存第一帧画面
2 myclip.save_frame('frame.jpeg', t='01:00:00') # 保存1h时刻的帧画面
```

如果这些 clip 有 mask 的话，mask 将会作为图片的 alpha 层被保存。除非我们使用 **withmask=False** 指定不保存。

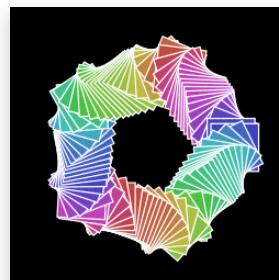
3. 效果展示

这里都是一些 moviepy 的实现，包括视频特效、Gif 动画、3D 特效果等等很炫的例子。

传送门：[MoviePy 画廊](#)

有一些是 youtube 的视频链接，需要上网，还有一些 youtube 链接失效了。

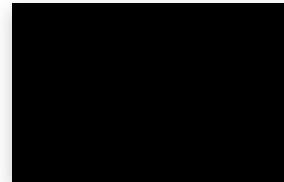
这边放几个炫酷的例子吊吊胃口！



4.MoviePy 实战案例

4.1 炫动的字母特效

这个例子，一定能很好的地说明为什么大家有兴趣基于 MoviePy 脚本来实现一些特效，我们也可以想像一下如果做的话，绝壁手抽筋了。



代码实现

```

1 import numpy as np
2 from moviepy.editor import *
3 from moviepy.video.tools.segmenting import findObjects
4
5 # 目标是创建炫动的文字，先创建TextClip，然后设置它居中
6
7 screensize = (720,460)
8 txtClip = TextClip('Cool effect',color='white', font="Amiri-Bold",
9                     kerning = 5, fontsize=100)
10 cvc = CompositeVideoClip([txtClip.set_pos('center')],
11                          size=screensize)
12
13 # 下面的四个函数，定义了四种移动字母的方式
14
15 # helper function
16 rotMatrix = lambda a: np.array([[np.cos(a),np.sin(a)],
17                                 [-np.sin(a),np.cos(a)]])
18
19 def vortex(screenpos,i,nletters):
20     d = lambda t : 1.0/(0.3+t**8) #damping
21     a = i*np.pi/ nletters # angle of the movement
22     v = rotMatrix(a).dot([-1,0])
23     if i%2 : v[1] = -v[1]
24     return lambda t: screenpos+400*d(t)*rotMatrix(0.5*d(t)*a).dot(v)
25
26 def cascade(screenpos,i,nletters):
27     v = np.array([0,-1])
28     d = lambda t : 1 if t<0 else abs(np.sinc(t)/(1+t**4))
29     return lambda t: screenpos+v*400*d(t-0.15*i)
30
31 def arrive(screenpos,i,nletters):
32     v = np.array([-1,0])
33     d = lambda t : max(0, 3-3*t)
34     return lambda t: screenpos-400*v*d(t-0.2*i)
35
36 def vortexout(screenpos,i,nletters):
37     d = lambda t : max(0,t) #damping
38     a = i*np.pi/ nletters # angle of the movement
39     v = rotMatrix(a).dot([-1,0])
40     if i%2 : v[1] = -v[1]
41     return lambda t: screenpos+400*d(t-0.1*i)*rotMatrix(-0.2*d(t)*a).dot(v)
42
43
44
45 # WE USE THE PLUGIN findObjects TO LOCATE AND SEPARATE EACH LETTER
46
47 letters = findObjects(cvc) # a list of ImageClips
48
49
50 # 让字母动起来
51 def moveLetters(letters, funcpos):
52     return [letter.set_pos(funcpos(letter.screenpos,i,len(letters)))
53             for i,letter in enumerate(letters)]

```

```

58
59 # 连接，写入文件
60 final_clip = concatenate_videoclips(clips)
61 final_clip.write_videofile('coolTextEffects.mp4', fps=25, codec='mp4')

```

4.2 重新构建 15 世纪舞蹈视频



<https://blog.csdn.net/ucsheep>

回复

```

1 # -*- coding: utf-8 -*-
2
3 """
4 Result: https://www.youtube.com/watch?v=Qu7HJrsEYFg
5 This is how we can imagine knights dancing at the 15th century, based on a very
6 serious historical study here: https://www.youtube.com/watch?v=zvCvOC2VwDc
7 Here is what we do:
8 0- Get the video of a dancing knight, and a (Creative Commons) audio music file.
9 1- load the audio file and automatically find the tempo
10 2- load the video and automatically find a segment that loops well
11 3- extract this segment, slow it down so that it matches the audio tempo,
12     and make it loop forever.
13 4- Symmetrize this segment so that we will get two knights instead of one
14 5- Add a title screen and some credits, write to a file.
15 This example has been originally edited in an IPython Notebook, which makes it
16 easy to preview and fine-tune each part of the editing.
17 """
18
19 from moviepy.editor import *
20 from moviepy.video.tools.cuts import find_video_period
21 from moviepy.audio.tools.cuts import find_audio_period
22
23
24 # Next lines are for downloading the required videos from Youtube.
25 # To do this you must have youtube-dl installed, otherwise you will need to
26 # download the videos by hand and rename them, as follows:
27 #     https://www.youtube.com/watch?v=zvCvOC2VwDc => knights.mp4
28 #     https://www.youtube.com/watch?v=lkY3Ek9VPtg => frontier.mp4
29
30 import os
31 if not os.path.exists("knights.mp4"):
32     os.system("youtube-dl zvCvOC2VwDc -o knights.mp4")
33     os.system("youtube-dl lkY3Ek9VPtg -o frontier.mp4")
34 =====
35
36
37 # LOAD, EDIT, ANALYZE THE AUDIO
38
39
40 audio = (AudioFileClip("frontier.mp4")
41             .subclip((4,7), (4,18))
42             .audio_fadein(1)
43             .audio_fadeout(1))
44
45 audio_period = find_audio_period(audio)
46 print ('Analyzed the audio, found a period of %.02f seconds'%audio_period)
47
48
49 # LOAD, EDIT, ANALYZE THE VIDEO
50
51
52 clip = (VideoFileClip("./knights.mp4", audio=False)
53             .subclip((1,24.15),(1,26)))

```

```

58
59 edited_right = (clip.subclip(0,video_period)
60         .speedx(final_duration=2*audio_period)
61         .fx(vfx.loop, duration=audio.duration)
62         .subclip(.25))
63
64 edited_left = edited_right.fx(vfx.mirror_x)
65
66 dancing_knights = (clips_array([[edited_left, edited_right]])
67         .fadein(1).fadeout(1).set_audio(audio).subclip(.3))
68
69 # MAKE THE TITLE SCREEN
70
71
72 txt_title = (TextClip("15th century dancing\n(hypothetical)", fontsize=70,
73                     font="Century-Schoolbook-Roman", color="white")
74                     .margin(top=15, opacity=0)
75                     .set_position(("center","top")))
76
77 title = (CompositeVideoClip([dancing_knights.to_ImageClip(), txt_title])
78         .fadein(.5)
79         .set_duration(3.5))
80
81
82 # MAKE THE CREDITS SCREEN
83
84
85 txt_credits = """
86 CREDITS
87 Video excerpt: Le combat en armure au XVe siècle
88 By J. Donzé, D. Jaquet, T. Schmuziger,
89 Université de Genève, Musée National de Moyen Age
90 Music: "Frontier", by DOCTOR VOX
91 Under licence Creative Commons
92 https://www.youtube.com/user/DOCTORVOXofficial
93 Video editing © Zulko 2014
94 Licence Creative Commons (CC BY 4.0)
95 Edited with MoviePy: http://zulko.github.io/moviepy/
96 """
97
98 credits = (TextClip(txt_credits, color='white',
99                 font="Century-Schoolbook-Roman", fontsize=35, kerning=-2,
100                interline=-1, bg_color='black', size=title.size)
101                .set_duration(2.5)
102                .fadein(.5)
103                .fadeout(.5))
104
105
106 # ASSEMBLE EVERYTHING, WRITE TO FILE
107
108 final = concatenate_videoclips([title, dancing_knights, credits])
109
110 final.write_videofile("dancing_knights.mp4", fps=clip.fps,
111                         audio_bitrate="1000k". bitrate="4000k")

```

4.3 一个简单的音乐视频

4.4 一个操作音频的例子

4.5 做一个《星战》风格的片头



这种透视效果需要比较复杂的转换，但是这样的练习也挺不错的

复制

```

1  """
2 Description of the video:
3 Mimic of Star Wars' opening title. A text with a (false)
4 perspective effect goes towards the end of space, on a
5 background made of stars. Slight fading effect on the text.
6 """
7
8 import numpy as np
9 from skimage import transform as tf
10
11 from moviepy.editor import *
12 from moviepy.video.tools.drawing import color_gradient
13
14
15 # RESOLUTION
16
17 w = 720
18 h = w*9/16 # 16/9 screen
19 moviesize = w,h
20
21
22
23 # THE RAW TEXT
24 txt = "\n".join([
25 "A long time ago, in a faraway galaxy,",
26 "there lived a prince and a princess",
27 "who had never seen the stars, for they",
28 "lived deep underground.",
29 "",
30 "Many years before, the prince's",
31 "grandfather had ventured out to the",
32 "surface and had been burnt to ashes by",
33 "solar winds.",
34 "",
35 "One day, as the princess was coding",
36 "and the prince was shopping online, a",
37 "meteor landed just a few megameters",
38 "from the couple's flat."
39 ])
40
41 # Add blanks
42 txt = 10*"\\n" +txt + 10*"\\n"
43
44
45
46 # CREATE THE TEXT IMAGE
47
48
49 clip_txt = TextClip(txt,color='white', align='West',fontsize=25,
50 font='Xolonium-Bold', method='label')
51
52
53 # SCROLL THE TEXT IMAGE BY CROPPING A MOVING AREA
54
55 txt_speed = 27
56 fl = lambda gf,t : gf(t)[int(txt_speed*t):int(txt_speed*t)+h,:]
57 moving_txt= clip_txt.fl(fl, apply_to=['mask'])
58
59
60 # ADD A VANISHING EFFECT ON THE TEXT WITH A GRADIENT MASK

```

```

65 fl = lambda pic : np.minimum(pic,gradmask.img)
66 moving_txt.mask = moving_txt.mask.fl_image(fl)
67
68
69 # WARP THE TEXT INTO A TRAPEZOID (PERSPECTIVE EFFECT)
70
71 def trapzWarp(pic,cx,cy,ismask=False):
72     """ Complicated function (will be latex packaged as a fx) """
73     Y,X = pic.shape[:2]
74     src = np.array([[0,0],[X,0],[X,Y],[0,Y]])
75     dst = np.array([[cx*X,cy*Y],[((1-cx)*X,cy*Y),(X,Y),(0,Y)]])
76     tform = tf.ProjectiveTransform()
77     tform.estimate(src,dst)
78     im = tf.warp(pic, tform.inverse, output_shape=(Y,X))
79     return im if ismask else (im*255).astype('uint8')
80
81 fl_im = lambda pic : trapzWarp(pic,0.2,0.3)
82 fl_mask = lambda pic : trapzWarp(pic,0.2,0.3, ismask=True)
83 warped_txt= moving_txt.fl_image(fl_im)
84 warped_txt.mask = warped_txt.mask.fl_image(fl_mask)
85
86
87 # BACKGROUND IMAGE, DARKENED AT 60%
88
89 stars = ImageClip('.../videos/stars.jpg')
90 stars_darkened = stars.fl_image(lambda pic: (0.6*pic).astype('int16'))
91
92
93 # COMPOSE THE MOVIE
94
95 final = CompositeVideoClip([
96     stars_darkened,
97     warped_txt.set_pos(('center','bottom')),
98     size = moviesize)
99
100
101 # WRITE TO A FILE
102
103 final.set_duration(8).write_videofile("starworms.avi", fps=5)
104
105 # This script is heavy (30s of computations to render 8s of video)
106
107
108
109 =====
110     CODE FOR THE VIDEO TUTORIAL
111     We will now code the video tutorial for this video.
112     When you think about it, it is a code for a video explaining how to
113     make another video using some code (this is so meta !).
114     This code uses the variables of the previous code (it should be placed
115     after that previous code to work).
116 =====
117
118
119
120 def annotate(clip,txt,txt_color='white',bg_color=(0,0,255)):
121     """ Writes a text at the bottom of the clip. """
122
123     txtclip = TextClip(txt, fontsize=20, font='Ubuntu-bold',
124                         color=txt_color)
125
126     txtclip = txtclip.on_color((clip.w,txtclip.h+6), color=(0,0,255),
127                               pos=(6,'center'))
128
129     cvc = CompositeVideoClip([clip , txtclip.set_pos((0,'bottom'))])
130
131     return cvc.set_duration(clip.duration)
132
133
134 def resizeCenter(clip):
135     return clip.resize( height=h).set_pos('center')
136
137
138 def composeCenter(clip):

```

```

143 annotated_clips = [ annotate(clip,text) for clip,text in [
144
145 (composeCenter(resizeCenter(stars)).subclip(0,3),
146     "This is a public domain picture of stars"),
147
148 (CompositeVideoClip([stars],moviesize).subclip(0,3),
149     "We only keep one part."),
150
151 (CompositeVideoClip([stars_darkened],moviesize).subclip(0,3),
152     "We darken it a little."),
153
154 (composeCenter(resizeCenter(clip_txt)).subclip(0,3),
155     "We generate a text image."),
156
157 (composeCenter(moving_txt.set_mask(None)).subclip(6,9),
158     "We scroll the text by cropping a moving region of it."),
159
160 (composeCenter(gradmask.to_RGB()).subclip(0,2),
161     "We add this mask to the clip."),
162
163 (composeCenter(moving_txt).subclip(6,9),
164     "Here is the result"),
165
166 (composeCenter(warped_txt).subclip(6,9),
167     "We now warp this clip in a trapezoid."),
168
169 (final.subclip(6,9),
170     "We finally superimpose with the stars.")
171 ]]
172
173 # Concatenate and write to a file
174
175

```

4.6 字幕特效-部分隐藏



```

1 from moviepy.editor import *
2 from moviepy.video.tools.credits import credits1
3
4 # 加载山背景的clip, 截取, 变慢, 画面变暗
5 clip = (VideoFileClip('..../videos/badl-0001.mov', audio=False)
6         .subclip(37,46)
7         .speedx( 0.4)
8         .fx( vfx.colorx, 0.7))
9
10 # 保存第一帧画面, 一会使用GIMP处理, 增加一个mask

```

```

15 mountainmask = ImageClip('....../credits/mountainMask2.png', ismask=True)
16
17 # 用一个文本文件内容生成字幕
18 credits = credits1('....../credits/credits.txt', 3*clip.w/4)
19 scrolling_credits = credits.set_pos(lambda t:('center',-10*t))
20
21
22 # 让字幕以10像素每秒的速度滚动起来
23 final = CompositeVideoClip([clip,
24                             scrolling_credits,
25                             clip.set_mask(mountainmask)])

```

4.7 使用画笔特效-定格视频中的某一帧

这样的处理手法，会让一帧画面看起来像画一样：

1. 用 Sobel 算法算出图像中的边缘，我们就获得了像手绘的黑白画面
2. 图片矩阵相乘，获得比较亮的画面，再叠加第一步获得的轮廓

最终的 clip 包含三级：处理之前的部分，处理的部分，处理过后的部分。处理的部分，是按照以下三步走来获得

1. 定格一帧画面，制作成手绘风格，作为一个 clip
2. 添加一个写着[Audrey]的 text clip 到第一步的 clip
3. 把的以上得到的 clip 覆盖在原始的 clip 上，让后让它用渐入和渐出的效果显示和移除显示

下面是代码

```

1 """ 需要安装1 scikit-image (for vfx.painting) """
2
3 from moviepy.editor import *
4
5 # WE TAKE THE SUBCLIPS WHICH ARE 2 SECONDS BEFORE & AFTER THE FREEZE
6
7 charade = VideoFileClip("....../videos/charade.mp4")
8 tfreeze = cvsecs(19.21) # Time of the freeze, 19'21
9
10 # when using several subclips of a same clip, it can be faster
11 # to create 'coreaders' of the clip (=other entrance points).
12 clip_before = charade.coreader().subclip(tfreeze -2,tfreeze)
13 clip_after = charade.coreader().subclip(tfreeze ,tfreeze +2)
14
15
16 # 被定格的一帧画面
17 im_freeze = charade.to_ImageClip(tfreeze)
18 painting = (charade.fx( vfx.painting, saturation = 1.6,black = 0.006)
19             .to_ImageClip(tfreeze))
20 #签名的TextClip
21 txt = TextClip('Audrey',font='Amiri-regular',fontsize=35)
22
23 painting_txt = (CompositeVideoClip([painting,txt.set_pos((10,180))])
24                  .add_mask()
25                  .set_duration(3)
26                  .crossfadein( 0.5)
27                  .crossfadeout( 0.5))
28
29 # FADEIN/FADEOUT EFFECT ON THE PAINTED IMAGE
30
31 painting_fading = CompositeVideoClip([im_freeze,painting_txt])
32
33
34 final_clip = concatenate_videoclips([ clip_before,
35                                         painting_fading.set_duration(3),

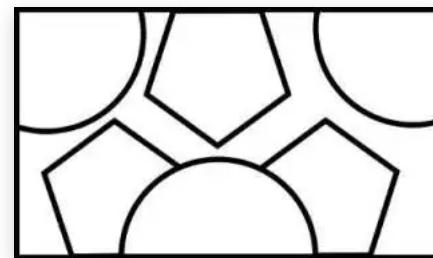
```

4.8 把多个 clip 放置在一个画面中（超美）



这么复杂的合成，是怎麽做出来的啊？！

在这个脚本中，我们会使用下面这个图：



我们将找到这些区域，然后把不同的 clip 填充到这些区域

```

1 from moviepy.editor import *
2 from moviepy.video.tools.segmenting import findObjects
3
4 # 加载用来指定区域的图像
5 im = ImageClip("../ultracompositing/motif.png")
6
7 # 加载这些区域返回一个ImageClip列表
8 regions = findObjects(im)
9
10
11 # 载入美国国家公园的7个clip
12 clips = [VideoFileClip(n, audio=False).subclip(18,22) for n in
13     [ "../videos/romo_0004.mov",
14     "../videos/apis-0001.mov",
15     "../videos/romo_0001.mov",
16     "../videos/elma_s0003.mov",
17     "../videos/elma_s0002.mov",
18     "../videos/calo-0007.mov",
19     "../videos/grsm_0005.mov"]]
20
21 # 把每一个clip都放置在对应的图片中的区域
22 comp_clips = [c.resize(r.size)
23                 .set_mask(r.mask)
24                 .set_pos(r.screenpos)
25                 for c,r in zip(clips,regions)]
26
27 cc = CompositeVideoClip(comp_clips,im.size)
28 cc.resize(0.6).write_videofile("../composition.mp4")
29
30 # 这个特殊的合成任务要花费很长的时间

```



4.10 追踪人脸，打马赛克



首先，我们需要先追踪到人脸，即，获得两个函数 `fx` 和 就像`(fx(t), (t)`)获得某时刻 `t` 脸的中心的坐标，通过 `tracking`，这个功能很快就实现了。然后我们就需要把脸的中心为半径的一部分区域做模糊化处理。

```
1 import pickle
2
3 from moviepy.editor import *
4 from moviepy.video.tools.tracking import manual_tracking, to_fxfy
5
6
7 # 加载clip，截取一个卓别林电影的6'51-7'01之间的片段
8 clip = VideoFileClip("../videos/chaplin.mp4").subclip((6,51.7),(7,01.3))
9
10 # 手动跟踪标记头部
11
12 # 下面的三行代码，手动跟踪，然后把结果保存进文件，应该在一次运行之后就完成量跟踪标记
13 # 注意：我们保存的格式是一个(ti,xi,yi)list，不是函数fx和fy
14
15 #txy, (fx,fy) = manual_tracking(clip, fps=6)
16 #with open("../chaplin_txy.dat",'w+') as f:
17 #     pickle.dump(txy)
18
19
20
21 # 已经完成手动跟踪人脸并标记的情况下
22 # fx(t),fy(t)的形式加载跟踪标记的数据
23
24 with open("../chaplin_txy.dat",'r') as f:
25     fx,fy = to_fxfy( pickle.load(f) )
26
27
28 # 在clip中，模糊卓别林的头部
29
30 clip_blurred = clip.fx( vfx.headblur, fx, fy, 25)
31
32
33 # 生成文本，灰色背景
34
```

```

38
39
40 # 把卓别林的vedio clip和TextClip连接起来, 添加audio clip
41
42 final = concatenate_videoclips([clip_blurred,txt.set_duration(3)].\
43         set_audio(clip.audio)
44
45 # 将比特率修改为3000k是为了画面不至于太丑
46
47 final.write_videofile(' / /blurredChonlin. avi' hitrate="3000k")

```

4.11 漂亮的结尾



<https://blog.csdn.net/ucsheep/>

回 复

```

1 from moviepy.editor import *
2 from moviepy.video.tools.drawing import circle
3
4 # 原有视频
5 clip = VideoFileClip("../videos/badl-0006.mov", audio=False).
6         subclip(26,31).
7         add_mask()
8
9 w,h = clip.size
10
11 # 这里的mask是一个半径按照 r(t) = 800-200*t 根据时间变化消失的圆
12 clip.mask.get_frame = lambda t: circle(screensize=(clip.w,clip.h),
13                                         center=(clip.w/2,clip.h/4),
14                                         radius=max(0,int(800-200*t)),
15                                         col1=1, col2=0, blur=4)
16
17
18 # 搞一个TextClip来放The End
19 the_end = TextClip("The End", font="Amiri-bold", color="white",
20                     fontsize=70).set_duration(clip.duration)
21
22 final = CompositeVideoClip([the_end.set_pos('center'),clip],
23                            size =clip.size)
24
25 final.write_videofile("../theEnd. avi")

```

回 复

5.Docker

之前有不少同学说搭建 moviepy 环境失败了，大部分是卡在 ffmpeg 自动下载的步骤了，原因在于从国外下载 ff。当然有办法解决：可以手动下载，然后放在`~/.imageio/ffmpeg` 目录下。

```

1 ucsheep@ucsheep-B250M-D2V:~/.imageio/ffmpeg$ ll
2 总用量 44864
3 drwxrwxr-x 2 ucsheep ucsheep 4096 8月 21 10:44 .
4 drwxrwxr-x 3 ucsheep ucsheep 4096 8月 14 08:54 ..
5 -rwxrw-r-- 1 ucsheep ucsheep 45929032 8月 21 10:44 ffmpeg-linux64-v3.3.1*

```

有两种办法，一种是通过 Dockerfile 构建，一种是直接在 Docker Hub 寻找有么有现成的镜像。

注意：进行以下操作的首要前提是，您的电脑已经安装 Docker

5.1 使用 Docker 镜像快速搭建 moviepy 环境

首先，查找镜像

```
1 ucsheep@ucsheep-B250M-D2V:~$ docker search moviepy
2 NAME                                     DESCRIPTION                                              STARS
3 dkarchmervue/moviepy                   Image for moviepy, with latest Python 3.4, l... 1
4 jdauphant/moviepy
5 mentlsve/anaconda3-opencv-moviepy     Anaconda3 (Python 3.5) with opencv and movie... 0
6 srid/moviepy
7 mbeacom/moviepy
8 jacquesvdm/moviepy                     Image that contains PHP and MoviePy          0
9 hongyusir/python-moviepy                docker image with python, pil and moviepy in... 0
10 ib5k/moviepy
11 screencom/moviepy
12 verynb/moviepy
13 hongyusir/mnpp                         mongo, nodejs and python together, and with ... 0
14 lucasrodesg/deepr                      Modified from ufoym/deepr. Incorporates pyth... 0
```

然后，选择第一个，拉取镜像，创建一个容器

```
1 ucsheep@ucsheep-B250M-D2V:~$ docker run -dit --name moviepy-test dkarchmervue/moviepy
2 4e4afde5a3702171ac345f924f4a40596fb081f20cf6984544c0b7855d5585b7
```

查看容器运行情况

```
1 ucsheep@ucsheep-B250M-D2V:~$ docker ps -a
2 CONTAINER ID        IMAGE               COMMAND      CREATED             STATUS              PORTS
3 4e4afde5a370        dkarchmervue/moviepy   "/bin/bash"   About a minute ago   Up 1 minute
```

进入容器，看看 moviepy 环境是否可用

```
1 ucsheep@ucsheep-B250M-D2V:~$ docker attach moviepy-test
2 root@4e4afde5a370:/work#
3 root@4e4afde5a370:/work# python3
4 Python 3.4.4 (default, May 25 2016, 06:34:52)
5 [GCC 4.8.4] on linux
6 Type "help", "copyright", "credits" or "license" for more information.
7 >>> import moviepy
8 >>>
```

没有报错，大功告成。可以将开发目录挂载到容器中的 work 目录，方便运行。

5.2 Dockerfile 构建

首先，clone MoviePy 源码

```
1 ucsheep@ucsheep-B250M-D2V:~/git-projects$ git clone https://github.com/Zulko/moviepy.git
```

查看，看到了 Dockerfile

```
1 ucsheep@ucsheep-B250M-D2V:~/git-projects$ cd moviepy/
2 ucsheep@ucsheep-B250M-D2V:~/git-projects/moviepy$ ls
3 appveyor      CONTRIBUTING.md  examples      MANIFEST.in  setup.py
4 appveyor.yml  Dockerfile      ez_setup.py  moviepy      tests
5 CHANGELOG.md  docs           LICENCE.txt  README.rst
```

```

1  FROM python:3
2
3  # Install numpy using system package manager
4  RUN apt-get -y update && apt-get -y install libav-tools imagemagick libopencv-dev python-opencv
5
6  # Install some special fonts we use in testing, etc..
7  RUN apt-get -y install fonts-liberation
8
9  RUN apt-get install -y locales && \
10   locale-gen C.UTF-8 && \
11   /usr/sbin/update-locale LANG=C.UTF-8
12
13 ENV LC_ALL C.UTF-8
14
15 # do we need all of these, maybe remove some of them?
16 RUN pip install imageio numpy scipy matplotlib pandas sympy nose decorator tqdm pillow pytest r
17
18 # install scikit-image after the other deps, it doesn't cause errors this way.
19 RUN pip install scikit-image sklearn
20
21 ADD . /var/src/moviepy/
22 #RUN git clone https://github.com/Zulko/moviepy.git /var/src/moviepy
23 RUN cd /var/src/moviepy/ && python setup.py install
24
25 # install ffmpeg from imageio.
26 RUN python -c "import imageio; imageio.plugins.ffmpeg.download()"
27
28 #add soft link so that ffmpeg can executed (like usual) from command line
29 RUN ln -s /root/.imageio/ffmpeg/ffmpeg.linux64 /usr/bin/ffmpeg
30
31 # modify ImageMagick policy file so that Textclips work correctly.
32 RUN cat /etc/ImageMagick-6/policy.xml | sed 's/none/read,write/g'> /etc/ImageMagick-6/policy.xml

```

就是在 Python3 的基础上，搭建量 MoviePy 的运行环境

那么，接下来，开始构建

```

1 ucsheep@ucsheep-B250M-D2V:~/git-projects/moviepy$ docker build -t moviepy -f Dockerfile .
2 Sending build context to Docker daemon 7.115MB
3 Step 1/12 : FROM python:3
4  --> 825141134528
5 Step 2/12 : RUN apt-get -y update && apt-get -y install libav-tools imagemagick libopencv-dev python-opencv
6  --> Running in b09eb178cc0a
7 Ign:1 http://deb.debian.org/debian stretch InRelease
8 Get:2 http://security.debian.org/debian-security stretch/updates InRelease [94.3 kB]
9 Get:3 http://deb.debian.org/debian stretch-updates InRelease [91.0 kB]
10 Get:4 http://deb.debian.org/debian stretch Release [118 kB]
11 Get:5 http://security.debian.org/debian-security stretch/updates/main amd64 Packages [437 kB]
12 Get:6 http://deb.debian.org/debian stretch-updates/main amd64 Packages [5148 B]
13 Get:7 http://deb.debian.org/debian stretch Release.gpg [2434 B]
14 Get:8 http://deb.debian.org/debian stretch/main amd64 Packages [7099 kB]

```

最终执行到 Step 12/12 后会构建完毕

接下来，可以运行容器，测试 moviepy 给出的示例代码：

```

1 ucsheep@ucsheep-B250M-D2V:~/git-projects/moviepy$ cd tests/
2 ucsheep@ucsheep-B250M-D2V:~/git-projects/moviepy/tests$ ls
3 download_media.py      test_fx.py          test_resourcererelease.py
4 __init__.py            test_helper.py       test_TextClip.py
5 README.rst             test_ImageSequenceClip.py  test_tools.py
6 test_AudioClips.py    test_issues.py      test_VideoClip.py
7 test_compositing.py   test_misc.py        test_VideoFileClip.py
8 test_examples.py       test_PR.py         test_Videos.py
9 test_ffmpeg_reader.py test_resourcerereleasedemo.py test_videotools.py
10 ucsheep@ucsheep-B250M-D2V:~/git-projects/moviepy/tests$ docker run -it -v `pwd`:/tests moviepy

```

版权声明: 本文为 InfoQ 作者【ucsheep】的原创文章。

原文链接: <https://xie.infoq.cn/article/23e694841b8526b2ba9d5fb7c>。

本文遵守【CC BY-NC】协议, 转载请保留原文出处及本版权声明。

[Python](#) [音视频](#) [视频剪辑](#) [Moviepy](#) [视频合成](#)



ucsheep

譬如朝露 去日苦多 2021.03.22 加入
You see! Sheep!



点赞 | 收藏 | 微信 | 微博 | 部落 |

评论 (2 条评论)

写下你的想法, 一起交流



1_bit

冲冲冲

2021 年 04 月 12 日 20:29

0



Geek_61fcca

您好! 安装了imagemagick之后, 在python根目录, MoviePy的配置环境变量的文件里config_defaults.py的后缀也打不开, 打不开是因为没有py的后缀吗?是什么原因造成的呢? 安装ImageMagick后产生的没有magick.exe这个文件, 有什么解决办法吗?

2021 年 04 月 08 日 07:38

0

没有更多了