Archtecture Assignment: MoviePy[1]

1 Overview

The MoviePy is a Python API library that helps to edit movies. The input could be images or videos and output could be videos of different formats. This API is built based on the following several dependencies:

- *numpy* - calculations and manipulations
- *imageio* - reading and writing images of different format
- *Decorator* - sync manipulation over videos on masks
- *tqdm* - showing video progress progress on command-line UI
- *youtube_dl* - interacting with Youtube
- *Sphinx* - creating *MoviePy* documentation
- *requests* - opening URL
- *ffmpeg* - processing video.

The Main components and their building blocks including:

- VideoClip - Image, Color, and Text
- AudioClip - Audio File and Composite Audio
- videofx - Video effects
- audiofx - Fade effect, looping, and volume
- videotools - Credits, drawing, segmenting, subtitles, and tracking
- audiotools - Not implemented
- ffmpeg - ffmpeg Tools for audio extraction, sub-clip extraction, merger video and audio, form video by frame, and resizing a video
- decorators - apply functions to mask and video

2 Architecture

The movie editing test came out with interaction with API components as shown in Figure 1.

[1]This review is done under docker MoviePy environment.

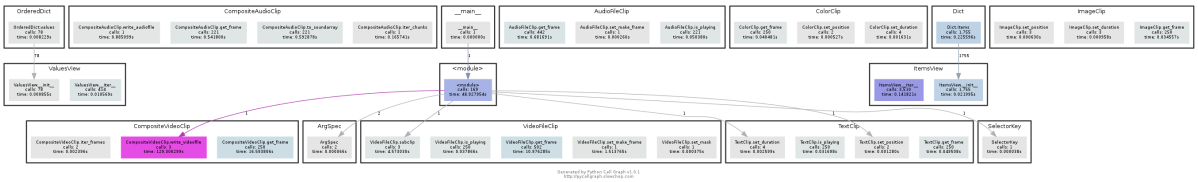

Figure 1: Architecture

This architecture flow graph outlines the function calls going through each involved component of the API. The test file resized a video input and output into a video in a different format. The main function source code is presented as the following.

```
#!/usr/bin/env python3
from moviepy.editor import *

video = VideoFileClip("myHolidays.mkv").subclip(50,60)

# Make the text. Many more options are available.
txt_clip = ( TextClip("My Holidays 2013",fontsize=70,color='white')
             .set_position('center')
             .set_duration(10) )

result = CompositeVideoClip([video, txt_clip]) # Overlay text on video
result.write_videofile("myHolidays_edited.webm",fps=25) # Many options...
```

The flow graph starts from the middle where the *__main__* module calls for components (Class): Composite Video Clip, Video Clip, and Text Clip. Within each class, several functions can be invoked. This specific main function invokes one to two functions in each class. However, inside the class, each function may be called by each other for several times and is shown on the flow graph. The ones not directly called but triggered by other functions in this specific main function are listed at the top.

3 Issue #640

The issue #640 is created regarding clip resizing function in file the "resize.py". This function plays an important role in video processing procedure. The issue for the function is considered bug and was proposed a fix for it. It is more of a pull request than an issue since a solution is proposed.

The problem code piece is attached as follow:

```
newsize2 = lambda t : trans_newsize(newsize(t))

        if clip.ismask:

            fun = lambda gf,t: (1.0*resizer((255 * gf(t)).astype('uint8'),
                                            newsize2(t))/255)

        else:

            fun = lambda gf,t: resizer(gf(t).astype('uint8'),
```

```
                                    newsize2(t))

        return clip.fl(fun, keep_duration=True,
                       apply_to= (["mask"] if apply_to_mask else []))
```

.

The issuer claims that the function fails for mask frames in the *else* argument. However, the *else* statement should not process any clip with a mask as the leading *if* statement is directing all the clip with mask into the first statement. The *else* statement should not handle any clips with a mask.

Nevertheless, the proposed solution is attached as below. The issuer copied handle from *ismask* statement to the else in an attempt to band-aid the error.

```
newsize2 = lambda t : trans_newsize(newsize(t))
        if clip.ismask:

            fun = lambda gf,t: (1.0*resizer((255 * gf(t)).astype('uint8'),
                                            newsize2(t))/255)

        else:

            fun = lambda gf,t: resizer(gf(t).astype('uint8'),
                                newsize2(t)) if len(gf(t).shape) == 3 else
                                    1.0*resizer((gf(t)*255).astype('uint8'),newsize2(t))/255

        return clip.fl(fun, keep_duration=True,
                       apply_to= (["mask"] if apply_to_mask else []))‘
```