MoviePy Software Architecture Review

MoviePy is a Python module for video editing, which can be used for basic operations (like cuts, concatenations, title insertions), video compositing (a.k.a. non-linear editing), video processing, or to create advanced effects. It can read and write the most common video formats, including GIF.

Use cases of MoviePy

- You have many videos to process or to compose in a complicated way.
- You want to automatize the creation of videos or GIFs on a web server (Django, Flask, etc.)
- You want to automatize tedious tasks, like title insertions tracking objects, cuting scenes, making end credits, subtitles, etc…
- You want to code your own video effects to do something no existing video editor can.
- You want to create animations from images generated by another python library (Matplotlib, Mayavi, Gizeh, scikit-images…)

Where is it NOT useful?

- This software package isn't useful when we want to perform frame-by-frame video analysis, for example, image processing needs. Libraries such OpenCV are more efficient and specialised for such kind of analysis.

- Simple video conversions or to turn a series of image files into a movie. It'd be easier to use ffmpeg directly and would be memory efficient as well.

Software Architecture

- MoviePy takes as it input video clips or audio clips or even a set of images.
- These inputs are read by FFMPEG and are then processed by python libraries such as Numpy, Pillow, Scikit and etc.
- Some of the various types of editing that can be performed using these libraries are concatenation, subclip (keep only the cut between t1 and t2), loop (plays the clip in a loop).
- Methods like clip.fx perform operations such as time mirror (plays the clip backwards) and black and white (converts the clip into black and white).
- There are many such operations performed in the processing stage which are elaborately and thoroughly documented here.
- Finally, all of the output is a video clip irrespective of the input format. All of output formats supported by FFMPEG are also supported by MoviePy.

The flow diagram below summarises the software architecture of MoviePy.

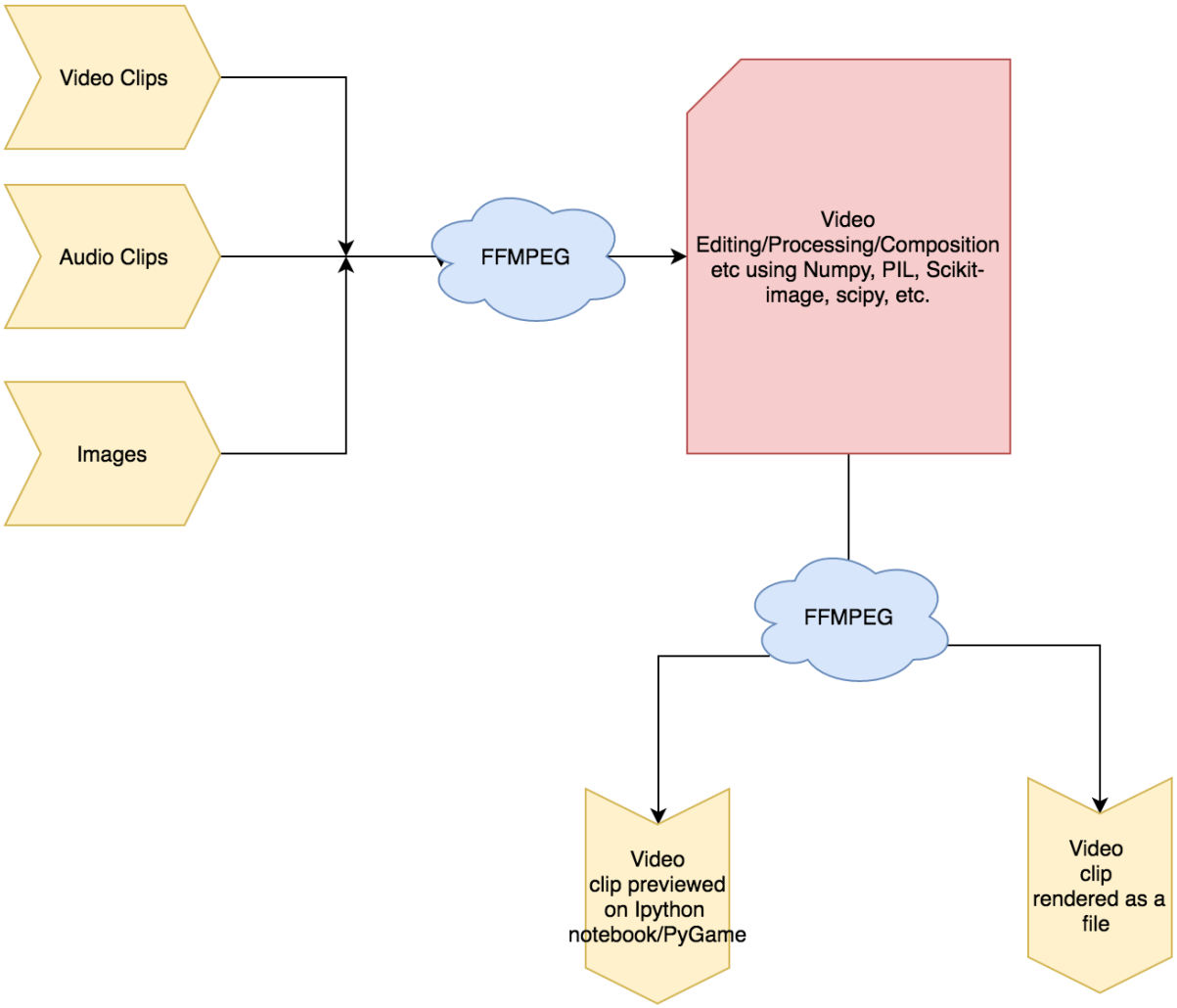

Fig 1. Flow diagram of the software architecture

Delving Deeper

Just to gain an understanding of how the library works, I implemented an example script provided in the documentation.

Fig 2. Example code

```
# Import everything needed to edit video clips
from moviepy.editor import *

# Load myHolidays.mp4 and select the subclip 00:00:50 - 00:00:60
clip = VideoFileClip("myHolidays.mp4").subclip(50,60)

# Reduce the audio volume (volume x 0.8)
clip = clip.volumex(0.8)

# Generate a text clip. You can customize the font, color, etc.
txt_clip = TextClip("My Holidays 2013",fontsize=70,color='white')

# Say that you want it to appear 10s at the center of the screen
txt_clip = txt_clip.set_pos('center').set_duration(10)

# Overlay the text clip on the first video clip
video = CompositeVideoClip([clip, txt_clip])

# Write the result to a file (many options available !)
video.write_videofile("myHolidays_edited.webm")
```

The central object of MoviePy is clips, which can be AudioClips or VideoClips and following code snippet shows how these clips are created. These clip objects are basically shallow copies of the original audio or video clip provided. This shallow copy is performed each time a filter or transformation is applied to a frame. Hence, making it very modular.

```
48
49      def __init__(self):
50
51          self.start = 0
52          self.end = None
53          self.duration = None
54
55          self.memoize = False
56          self.memoized_t = None
57          self.memoize_frame = None
58
59      def copy(self):
60          """ Shallow copy of the clip.
61
62          Returns a shallow copy of the clip whose mask and audio will
63          be shallow copies of the clip's mask and audio if they exist.
64
65          This method is intensively used to produce new clips every time
66          there is an outplace transformation of the clip (clip.resize,
67          clip.subclip, etc.)
68          """
69
70          newclip = copy(self)
71          if hasattr(self, 'audio'):
72              newclip.audio = copy(self.audio)
73          if hasattr(self, 'mask'):
74              newclip.mask = copy(self.mask)
75
76          return newclip
```

Fig 3. Code snippet of Clip class

Also, all of these transformations are applied to particular frames specified by the time at which the frame occurs. These frames are nothing but 2D Numpy arrays on which operations are performed.

The following code snippet is the the function definition of get_frame method.

```
77
78      @convert_to_seconds(['t'])
79      def get_frame(self, t):
80          """
81          Gets a numpy array representing the RGB picture of the clip at time t
82          or (mono or stereo) value for a sound clip
83          """
84          # Coming soon: smart error handling for debugging at this point
85          if self.memoize:
86              if t == self.memoized_t:
87                  return self.memoized_frame
88              else:
89                  frame = self.make_frame(t)
90                  self.memoized_t = t
91                  self.memoized_frame = frame
92                  return frame
93          else:
94              return self.make_frame(t)
95
```

Issue #646

Title: Duration floating point round-off causes "IndexError: list index out of range" in write_videofile

Description:
The sample code creates a color clip with a duration of 2.24, a size of [1080, 1080] and fps=50.0 The clip writes properly after CompositeVideoClip but if it is concatenated then the duration changes slightly and writing fails.

Code used:

```
>>> vclip = ColorClip([1080, 1080], color=[255, 128, 64], duration=2.24)
>>> vclip.duration
2.24
>>> vclip.size
(1080, 1080)
>>> comp = CompositeVideoClip([vclip])
>>> comp = comp.set_fps(50.0)
>>> comp.fps
50.0
>>> comp.write_videofile('comp.mp4', preset='ultrafast')
```

The file is written successfully.

Error Generated while concatenation operation is performed:

```
>>> vclip = ColorClip([1080, 1080], color=[255, 128, 64], duration=2.24)
>>> vclip.duration
2.24
>>> vclip.size
(1080, 1080)
>>> comp = CompositeVideoClip([vclip])
>>> comp = comp.set_fps(50.0)
>>> comp.fps
50.0
>>> comp.write_videofile('comp.mp4', preset='ultrafast')
```

The write_videofile method fails with IndexError

```
  Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<decorator-gen-51>", line 2, in write_videofile
  File "/usr/local/lib/python3.6/site-packages/moviepy/decorators.py", line 54, in requires
    return f(clip, *a, **k)
  File "<decorator-gen-50>", line 2, in write_videofile
  File "/usr/local/lib/python3.6/site-packages/moviepy/decorators.py", line 137, in use_cli
    return f(clip, *new_a, **new_kw)
  File "<decorator-gen-49>", line 2, in write_videofile
  File "/usr/local/lib/python3.6/site-packages/moviepy/decorators.py", line 22, in convert_
    return f(clip, *a, **k)
  File "/usr/local/lib/python3.6/site-packages/moviepy/video/VideoClip.py", line 349, in wr
    progress_bar=progress_bar)
  File "/usr/local/lib/python3.6/site-packages/moviepy/video/io/ffmpeg_writer.py", line 209
    fps=fps, dtype="uint8"):
  File "/usr/local/lib/python3.6/site-packages/tqdm/_tqdm.py", line 833, in __iter__
    for obj in iterable:
  File "/usr/local/lib/python3.6/site-packages/moviepy/Clip.py", line 475, in generator
    frame = self.get_frame(t)
  File "<decorator-gen-14>", line 2, in get_frame
  File "/usr/local/lib/python3.6/site-packages/moviepy/decorators.py", line 89, in wrapper
    return f(*new_a, **new_kw)
  File "/usr/local/lib/python3.6/site-packages/moviepy/Clip.py", line 95, in get_frame
    return self.make_frame(t)
  File "/usr/local/lib/python3.6/site-packages/moviepy/video/compositing/concatenate.py", l
    return clips[i].get_frame(t - tt[i])
IndexError: list index out of range
```

Interpretation and proposed Solution:

The issue here is that since the duration provided by user varies by 0.0000000000000002 s while performing the “concat” or concatenation operation. This causes the program to throw an IndexError.

In my opinion, 0.0000000000000002 s is an insignificant time in terms of visual representation since it’s impossible to notice such changes. In my opinion, by reducing the number of decimals to which the duration is rounded to probably 6 decimal places, the issue can be resolved.

Credentials:
Submitted by Arshitha Basavaraj
BUID: U27704208