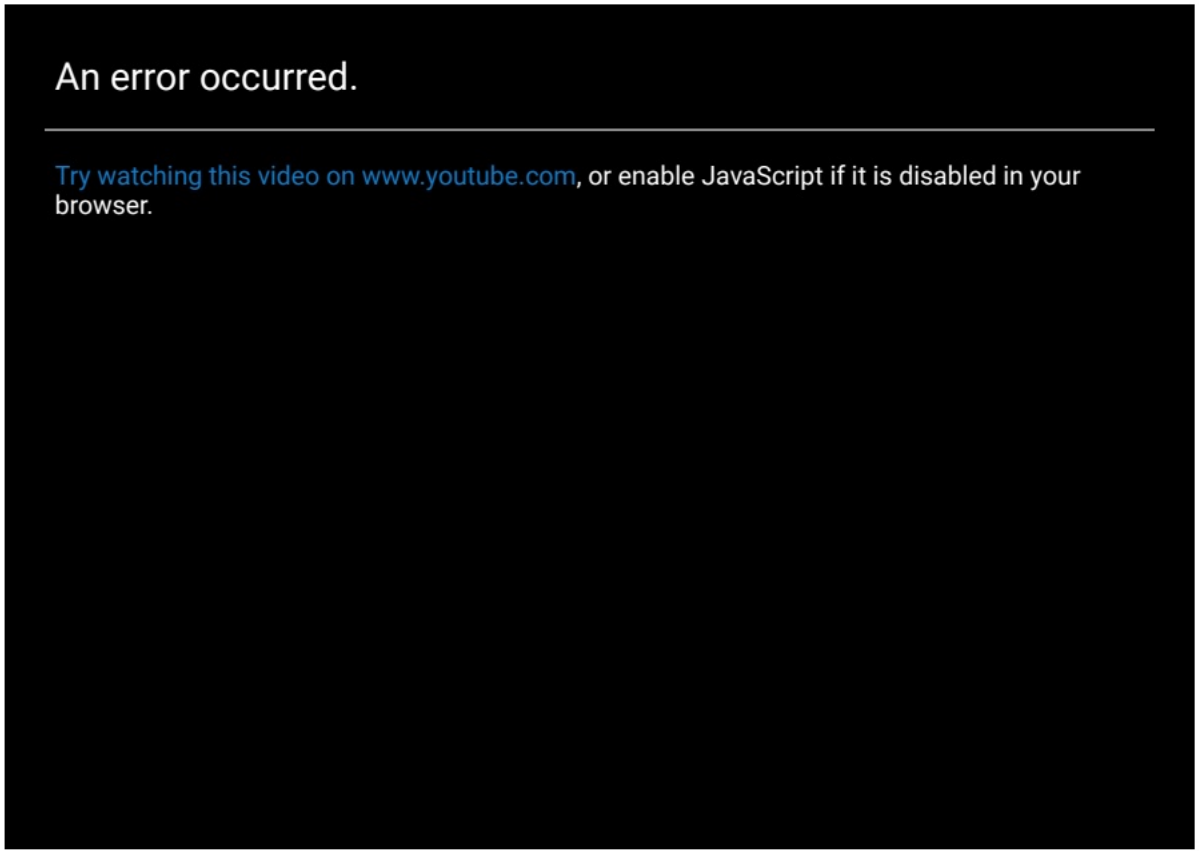
“The End” effect



So let’s explain this one: there is a clip with “The End” written in the middle, and *above* this clip there is the actual movie. The actual movie has a mask which represents a white (=opaque) circle on a black (=transparent) background. At the begining, that circle is so large that you see all the actual movie and you don’t see the “The End” clip. Then the circle becomes progressively smaller and as a consequence you see less of the actual movie and more of the “The End” clip.

```
from moviepy.editor import *
from moviepy.video.tools.drawing import circle

clip = VideoFileClip("../../videos/badl-0006.mov", audio=False).\
           subclip(26,31).\
           add_mask()

w,h = clip.size

# The mask is a circle with vanishing radius r(t) = 800-200*t
clip.mask.get_frame = lambda t: circle(screensize=(clip.w,clip.h),
                                       center=(clip.w/2,clip.h/4),
                                       radius=max(0,int(800-200*t)),
                                       col1=1, col2=0, blur=4)


the_end = TextClip("The End", font="Amiri-bold", color="white",
                   fontsize=70).set_duration(clip.duration)

final = CompositeVideoClip([the_end.set_pos('center'),clip],
                           size =clip.size)

final.write_videofile("../../theEnd.avi")
```

```
.. _opencv:

So you want to install OpenCV 2.4.6 ?
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

OpenCV is very optional, its installation is not always simple and I found it to be unstable, be warned !
The installation seems easy for Windows. On linux, here is what I found on the Internet:

- Remove any other version of OpenCV if you installed it through a package manager.
- Unzip the source code of `OpenCV 2.4.6` in some folder. open a terminal in this folder.
- Make a new directory and go into this directory: ::

      mkdir release
      cd release

- Run ``cmake``. Here is the line I used: ::

      cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON ..

- Run ``make``. This may take a few minutes (15 minutes on my computer). ::

      make

- Finally, install. ::

      sudo make install

And voilà !

You can check if it worked by opeing a Python console and typing ::

    import cv2
    print cv2.__version__

Advice: do not throw your ``release`` folder away. If later you have strange bugs with OpenCV involving ``.so`` files, just redo the
``sudo make install`` step.
```

Reference Manual

The documentation may be a little messy for the moment, it will get better with time. If you want to hack into the code or locate a particular function, read Organization of MoviePy's code .

```
Moviepy Docker
==============

Prequisites
-------------

1. Docker installed `Docker for Mac, Docker for windows, linux, etc <https://www.docker.com/get-docker/>`_
2. Build the Dockerfile ::

     docker build -t moviepy -f Dockerfile .


Steps to run the git repo unittests from docker
-----------------------------------------------

Get a bash prompt in the moviepy container ::

     cd tests
     docker run -it -v `pwd`:/tests moviepy bash

Run the tests ::

     cd tests
     python test_issues.py

Running your own moviepy script from docker
-------------------------------------------

Change directory to where your script is located

If moviepy docker container is already running, you can connect by: ::

     docker exec -it moviepy python myscript.py

If the container isn't running already ::

     docker run -it moviepy bash
     python myscript.py

You can also start a container and run a script in one command: ::

     docker run -it -v `pwd`:/code moviepy python myscript.py
```

Moviepy Docker

Prequisites

1. Docker installed Docker for Mac, Docker for windows, linux, etc
2. Build the Dockerfile

```
docker build -t moviepy -f Dockerfile .
```

Steps to run the git repo unittests from docker

Get a bash prompt in the moviepy container

```
cd tests
docker run -it -v `pwd`:/tests moviepy bash
```

Run the tests

```
cd tests
python test_issues.py
```

Running your own moviepy script from docker

Change directory to where your script is located

If moviepy docker container is already running, you can connect by:

```
docker exec -it moviepy python myscript.py
```

If the container isn't running already

```
docker run -it moviepy bash
python myscript.py
```

You can also start a container and run a script in one command:

```
docker run -it -v `pwd`:/code moviepy python myscript.py
```

Advanced tools

This section will briefly present the submodule `moviepy.video.tools` that can help you edit videos. It's not ready yet, see video.tools (the same in more complete and more technical) instead.

Tracking

Cuts

Subtitles

Credits

```
.. _install:

Download and Installation
=========================


Installation
--------------

**Method with pip:** if you have ``pip`` installed, just type this in a terminal (it will install ez_setup if you don't already have it)
::

    (sudo) pip install moviepy

If you have neither ``setuptools`` nor ``ez_setup`` installed the command above will fail, is this case type this before installing: ::

    (sudo) pip install ez_setup

**Method by hand:** download the sources, either on PyPI_ or (if you want the development version) on Github_, unzip everything in one
folder, open a terminal and type ::

    (sudo) python setup.py install

MoviePy depends on the Python modules Numpy_, imageio_, Decorator_, and tqdm_, which will be automatically installed during MoviePy's
installation. It should work  on Windows/Mac/Linux, with Python 2.7+ and 3 ; if you have trouble installing MoviePy or one of its
dependencies, please provide feedback !

MoviePy depends on the software FFMPEG for video reading and writing. You don't need to worry about that, as FFMPEG should be
automatically downloaded/installed by ImageIO during your first use of MoviePy (it takes a few seconds). If you want to use a specific
version of FFMPEG, you can set the FFMPEG_BINARY environment variable See ``moviepy/config_defaults.py`` for details.


Other optional but useful dependencies
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ImageMagick_ is not strictly required, only if you want to write texts. It can also be used as a backend for GIFs but you can do GIFs with
MoviePy without ImageMagick.

Once you have installed it, ImageMagick will be automatically detected by MoviePy, **except on Windows !**. Windows user, before
installing MoviePy by hand, go into the ``moviepy/config_defaults.py`` file and provide the path to the ImageMagick binary called
`magick`. It should look like this ::

    IMAGEMAGICK_BINARY = "C:\\Program Files\\ImageMagick_VERSION\\magick.exe"

You can also set the IMAGEMAGICK_BINARY environment variable See ``moviepy/config_defaults.py`` for details.

If you are using an older version of ImageMagick, keep in mind the name of the executable is not ``magick.exe`` but ``convert.exe``. In
that case, the IMAGEMAGICK_BINARY property should be ``C:\\Program Files\\ImageMagick_VERSION\\convert.exe``

PyGame_ is needed for video and sound previews (useless if you intend to work with MoviePy on a server but really essential for advanced
```

video editing *by hand*).

For advanced image processing you will need one or several of these packages. For instance using the method ``clip.resize`` requires that at least one of Scipy, PIL, Pillow or OpenCV are installed.

- The Python Imaging Library (PIL) or, better, its branch Pillow_ .
- Scipy_ (for tracking, segmenting, etc.), and can be used for resizing video clips if PIL and OpenCV aren't installed on your computer.
- `Scikit Image`_ may be needed for some advanced image manipulation.
- `OpenCV 2.4.6`_ or more recent (provides the package ``cv2``) or more recent may be needed for some advanced image manipulation.

If you are on linux, these packages will likely be in your repos.

.. _`Numpy`: https://www.scipy.org/install.html
.. _Decorator: https://pypi.python.org/pypi/decorator
.. _tqdm: https://pypi.python.org/pypi/tqdm

.. _ffmpeg: https://www.ffmpeg.org/download.html

.. _imageMagick: https://www.imagemagick.org/script/index.php
.. _Pygame: https://www.pygame.org/download.shtml
.. _imageio: https://imageio.github.io/

.. _Pillow: https://pillow.readthedocs.org/en/latest/
.. _Scipy: https://www.scipy.org/
.. _`Scikit Image`: http://scikit-image.org/download.html

.. _Github: https://github.com/Zulko/moviepy
.. _PyPI: https://pypi.python.org/pypi/moviepy
.. _`OpenCV 2.4.6`: https://sourceforge.net/projects/opencvlibrary/files/

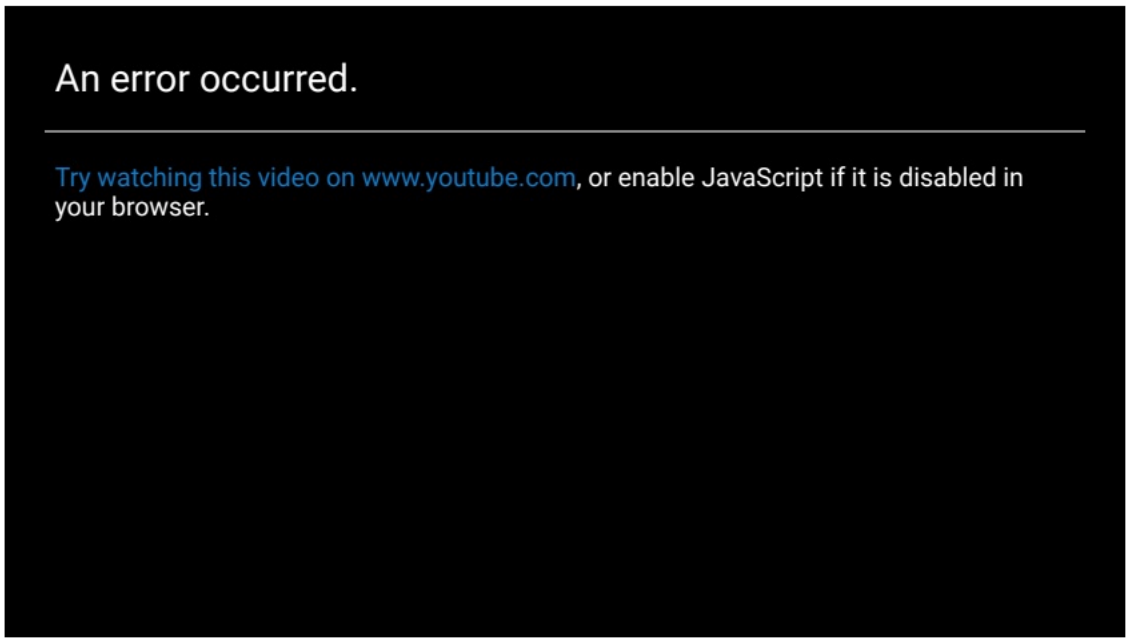
Gallery

Here are a few projects using MoviePy. The gallery will fill up as more people start using MoviePy (which is currently one year old). If you have a nice project using MoviePy let us know !

Videos edited with Moviepy

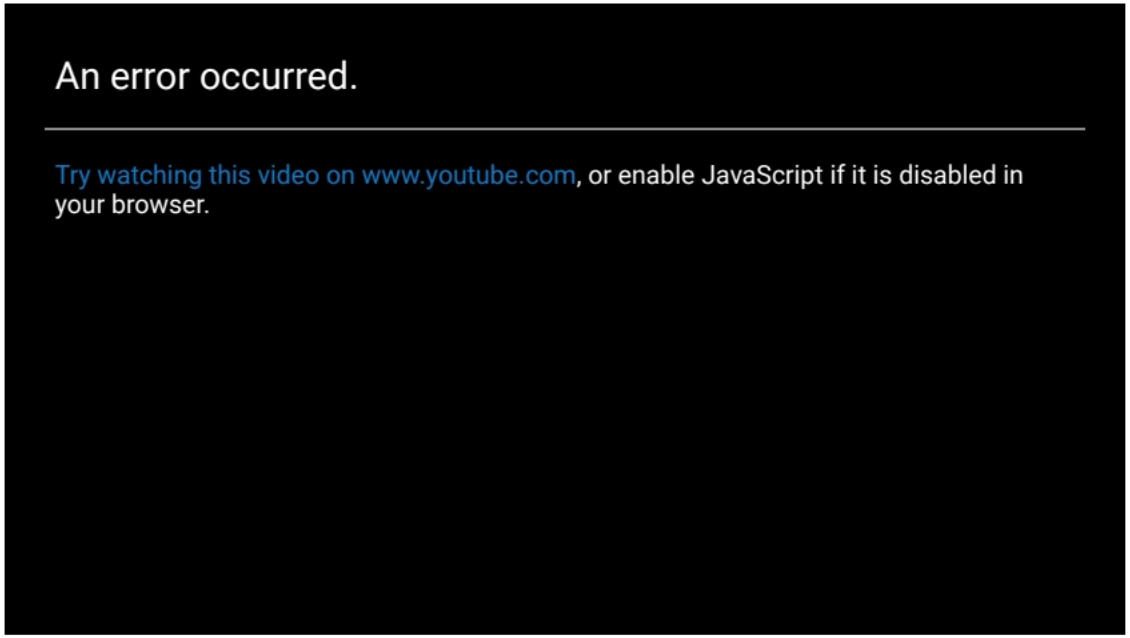
The Cup Song Covers Mix

This mix of 60 covers of the Cup Song demonstrates the non-linear video editing capabilities of MoviePy. Here is the (undocumented) MoviePy code that generated the video.



The (old) MoviePy reel video.

Made when MoviePy was a few weeks old and not as good as now. The code for most scenes can be found in the Example Scripts.



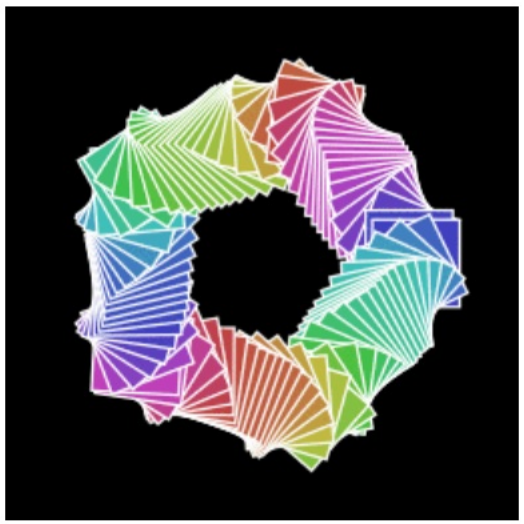
Animations edited with MoviePy

GIFs made from videos

This gifs tutorial gives you the basics to make gifs from video files (cutting, croping, adding text...). The last example shows how to remove a (still) background to keep only the animated part of a video.

Vector Animations

This vector animations tutorial shows how to combine MoviePy with Gizeh to create animations:



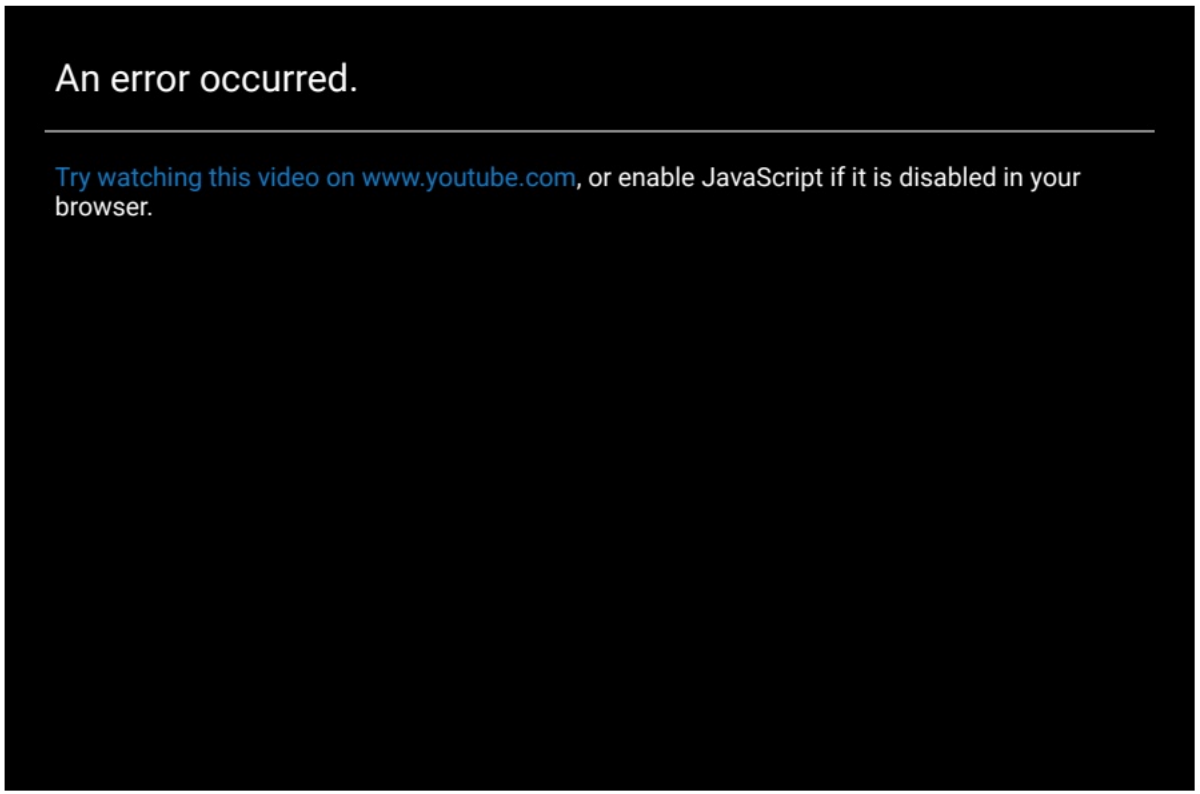
It is also possible to combine MoviePy with other graphic librairies like matplotlib, etc.

3D animations

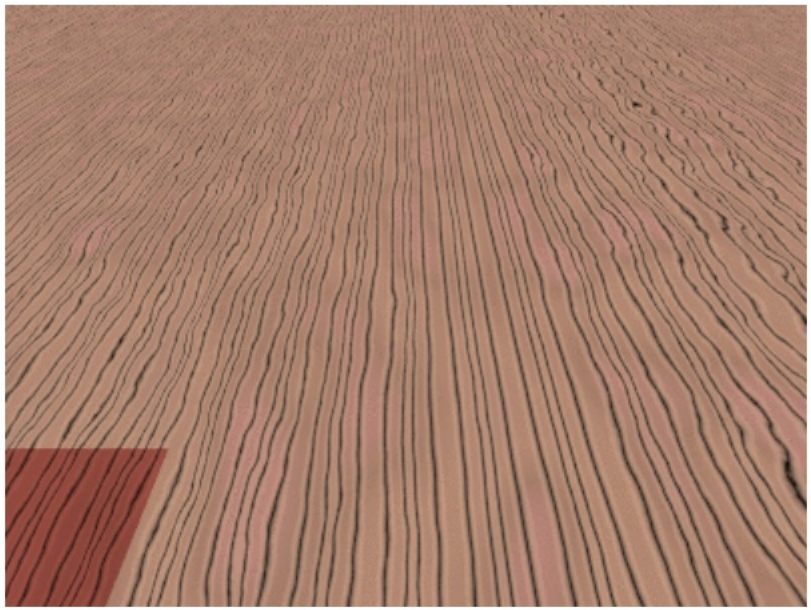
This 3d animation tutorial shows how to combine MoviePy with Vapory, a library to render 3D scenes using the free ray-tracer POV-Ray

VAPORY

With Vapory and MoviePy you can for instance embed a movie in a 3D scene:



Or render the result of this physics simulation made with PyODE (script):



Or use this script to make piano animations from MIDI files (which are some sort of electronic sheet music):

An error occurred.

Try watching this video on www.youtube.com, or enable JavaScript if it is disabled in your browser.

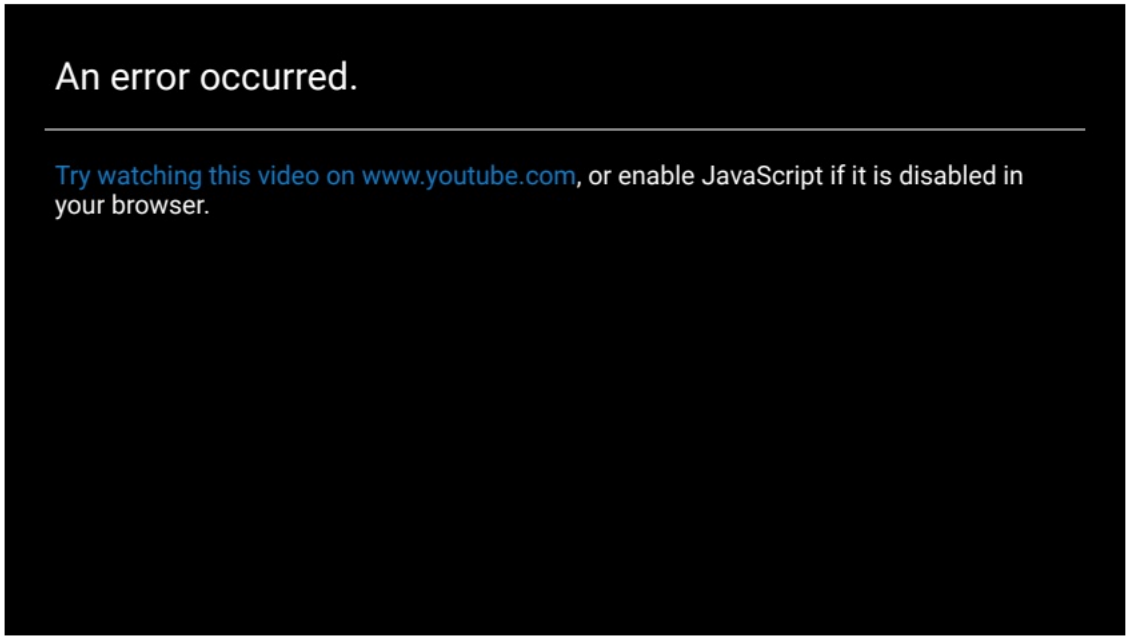
Data animations

This data animation tutorial shows how to use MoviePy to animate the different Python vizualization libraries: Mayavi, Vispy, Scikit-image, Matplotlib, etc.

Scientific or technological projects

Piano rolls transcription to sheet music

This transcribing piano rolls blog post explains how to transform a video of a piano roll performance into playable sheet music. MoviePy is used for the frame-by-frame analysis of the piano roll video. The last video is also edited with MoviePy:



Misc. Programs and Scripts using MoviePy

Kapwing

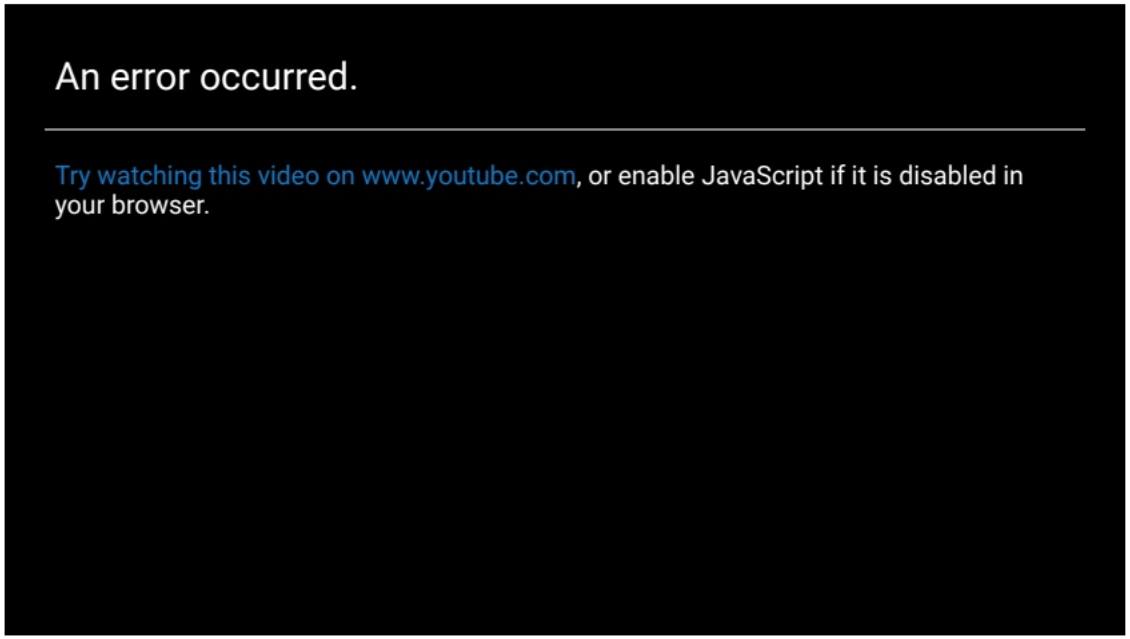
Kapwing is an online video meme generator. Content creators use Kapwing to add text around their videos, which results in higher engagement / views on social media sites like Facebook. Kapwing's creation process is powered by MoviePy! MoviePy is used to add the text, borders, and attribution directly to the uploaded videos.

Rinconcam

Rincomcam is a camera which films surfers on the Californian beach of Point Rincon. At the end of each day it cuts together a video, puts it online, and tweets it. Everything is entirely automatized with Python. MoviePy is used to add transitions, titles and music to the videos.

Videogrep

Videogrep is a python script written by Sam Lavigne, that goes through the subtitle tracks of movies and makes supercuts based on what it finds. For instance, here is an automatic supercut of every time the White House press secretary tells us what he can tell us:

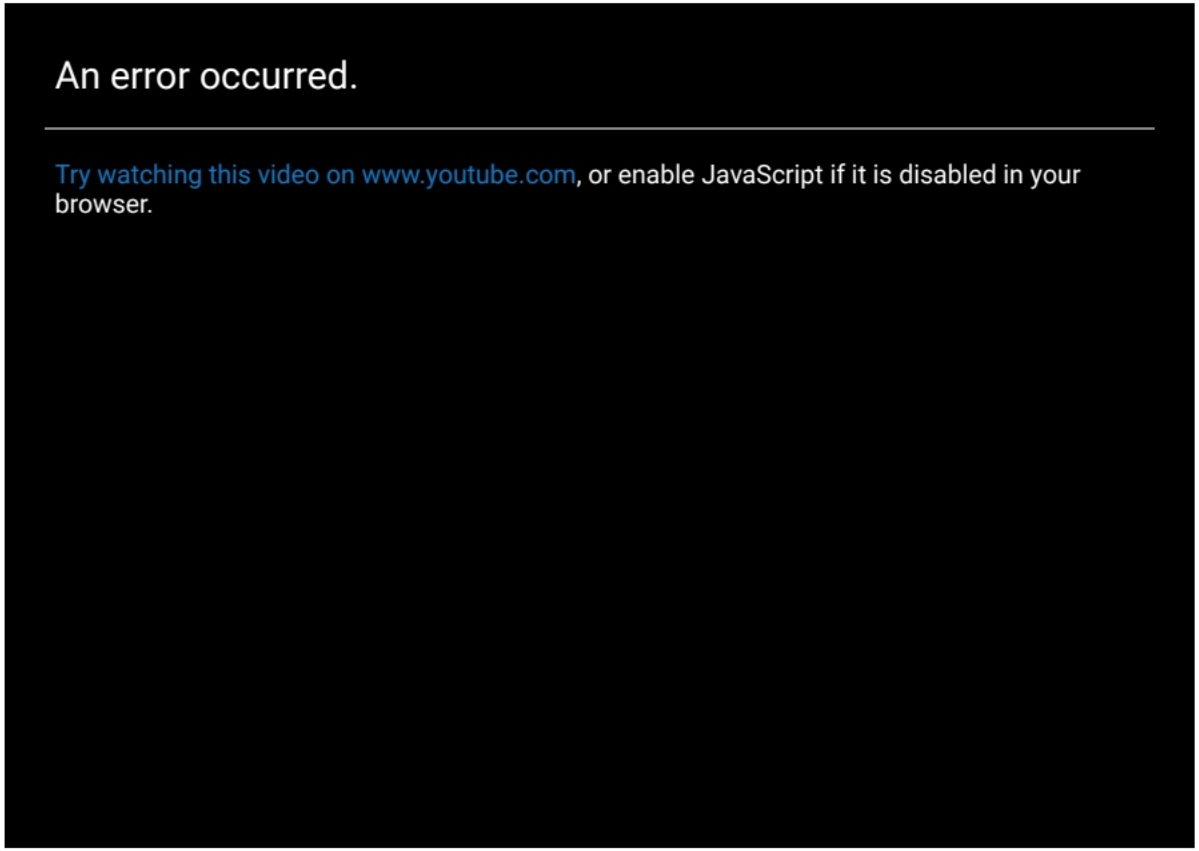


Here are Videogrep's introductory blog post and the Github Videogrep page.

If you liked it, also have a look at these Videogrep-inspired projects:

This Videogrep blog post attempts to cut a video precisely at the beginning and end of sentences or words:

```
words = ["Americans", "must", "develop", "open ", "source",
          " software", "for the", " rest ", "of the world",
          "instead of", " soldiers"]
numbers = [3,0,4,3,4,0,1,2,0,1,0] # take clip number 'n'

cuts = [find_word(word)[n] for (word,n) in zip(words, numbers)]
assemble_cuts(cuts, "fake_speech.mp4")
```



This other post uses MoviePy to automatically cut together all the highlights of a soccer game, based on the fact that the crowd cheers louder when something interesting happens. All in under 30 lines of Python:

```
FAQ and troubleshooting
=======================

This section will fill up as MoviePy advances through the next steps of
development (currently on the roadmap: MoviePy Studio, MoviePy WebApp, MoviePy OS, MoviePy
Trust Inc., and the MoviePy Charity Fundation).

Common errors that are not bugs
-------------------------------

These are very common errors which are not considered as bugs to be
solved (but you can still ask for this to change). If these answers
don't work for you, please open a bug report on Github_, or on the dedicated forum on Reddit_, or on the librelist_.

MoviePy generated a video that cannot be read by my favorite player.
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Known reason: one of the video's dimensions were not even,
for instance 720x405, and you used a MPEG4 codec like libx264 (default
in MoviePy). In this case the video generated uses a format that is
readable only on some readers like VLC.

I can't seem to read any video with MoviePy
"""""""""""""""""""""""""""""""""""""""""""""""""""

Known reason: you have a deprecated version of FFMPEG, install a recent version from the
website, not from your OS's repositories ! (see :ref:`install`).

Previewing videos make them slower than they are
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

It means that your computer is not good enough to render the clip in real time. Don't hesitate to play with the options of ``preview``:
for instance, lower the fps of the sound (11000 Hz is still fine) and the video. Also, downsizing your video with ``resize`` can help.

.. _Github: https://github.com/Zulko/moviepy
.. _Reddit: https://www.reddit.com/r/moviepy/
.. _librelist: mailto:moviepy@librelist.com
```

FAQ and troubleshooting

This section will fill up as MoviePy advances through the next steps of development (currently on the roadmap: MoviePy Studio, MoviePy WebApp, MoviePy OS, MoviePy Trust Inc., and the MoviePy Charity Fundation).

Common errors that are not bugs

These are very common errors which are not considered as bugs to be solved (but you can still ask for this to change). If these answers don't work for you, please open a bug report on Github, or on the dedicated forum on Reddit, or on the librelist.

MoviePy generated a video that cannot be read by my favorite player.

Known reason: one of the video's dimensions were not even, for instance 720x405, and you used a MPEG4 codec like libx264 (default in MoviePy). In this case the video generated uses a format that is readable only on some readers like VLC.

I can't seem to read any video with MoviePy

Known reason: you have a deprecated version of FFMPEG, install a recent version from the website, not from your OS's repositories ! (see Download and Installation).

Previewing videos make them slower than they are

It means that your computer is not good enough to render the clip in real time. Don't hesitate to play with the options of `preview`: for instance, lower the fps of the sound (11000 Hz is still fine) and the video. Also, downsizing your video with `resize` can help.

Example Scripts

Here are a few example scripts to get you started. Most are quite old now and will be soon replaced.

- Text with moving letters
- A reconstitution of 15th century dancing
- A simple music video
- An example with sound
- A Star-Wars like opening title
- Partially Hidden credits
- Freezing a movie frame with a painting effect
- Placing clips according to a picture
- MoviePy logo with a moving shadow
- Tracking and blurring someone's face
- Quick recipes
- Character duplication in a video
- "The End" effect

Example Scripts

Here are a few example scripts to get you started. Most are quite old now and will be soon replaced.

- Text with moving letters
- A reconstitution of 15th century dancing
- A simple music video
- An example with sound
- A Star-Wars like opening title
- Partially Hidden credits
- Freezing a movie frame with a painting effect
- Placing clips according to a picture
- MoviePy logo with a moving shadow
- Tracking and blurring someone's face
- Quick recipes
- Character duplication in a video
- "The End" effect

Getting started with MoviePy

These pages explain everything you need to start editing with MoviePy. To go further, have a look at the Gallery and the Example Scripts.

Download and Installation

Installation

Method with pip: if you have `pip` installed, just type this in a terminal (it will install ez_setup if you don't already have it)

```
(sudo) pip install moviepy
```

If you have neither `setuptools` nor `ez_setup` installed the command above will fail, is this case type this before installing:

```
(sudo) pip install ez_setup
```

Method by hand: download the sources, either on PyPI or (if you want the development version) on Github, unzip everything in one folder, open a terminal and type

```
(sudo) python setup.py install
```

MoviePy depends on the Python modules Numpy, imageio, Decorator, and tqdm, which will be automatically installed during MoviePy's installation. It should work on Windows/Mac/Linux, with Python 2.7+ and 3 ; if you have trouble installing MoviePy or one of its dependencies, please

provide feedback !

MoviePy depends on the software FFMPEG for video reading and writing. You don't need to worry about that, as FFMPEG should be automatically downloaded/installed by ImageIO during your first use of MoviePy (it takes a few seconds). If you want to use a specific version of FFMPEG, you can set the FFMPEG_BINARY environment variable See `moviepy/config_defaults.py` for details.

Other optional but useful dependencies

ImageMagick is not strictly required, only if you want to write texts. It can also be used as a backend for GIFs but you can do GIFs with MoviePy without ImageMagick.

Once you have installed it, ImageMagick will be automatically detected by MoviePy, **except on Windows !**. Windows user, before installing MoviePy by hand, go into the `moviepy/config_defaults.py` file and provide the path to the ImageMagick binary called . It should look like this

```
IMAGEMAGICK_BINARY = "C:\\Program Files\\ImageMagick_VERSION\\magick.exe"
```

You can also set the IMAGEMAGICK_BINARY environment variable See `moviepy/config_defaults.py` for details.

If you are using an older version of ImageMagick, keep in mind the name of the executable is not `magick.exe` but `convert.exe`. In that case, the IMAGEMAGICK_BINARY property should be `C:\\Program Files\\ImageMagick_VERSION\\convert.exe`

PyGame is needed for video and sound previews (useless if you intend to work with MoviePy on a server but really essential for advanced video editing).

For advanced image processing you will need one or several of these packages. For instance using the method `clip.resize` requires that at least one of Scipy, PIL, Pillow or OpenCV are installed.

- The Python Imaging Library (PIL) or, better, its branch Pillow .
- Scipy (for tracking, segmenting, etc.), and can be used for resizing video clips if PIL and OpenCV aren't installed on your computer.
- Scikit Image may be needed for some advanced image manipulation.
- OpenCV 2.4.6 or more recent (provides the package `cv2`) or more recent may be needed for some advanced image manipulation.

If you are on linux, these packages will likely be in your repos.

So you want to install OpenCV 2.4.6 ?

OpenCV is very optional, its installation is not always simple and I found it to be unstable, be warned ! The installation seems easy for Windows. On linux, here is what I found on the Internet:

- Remove any other version of OpenCV if you installed it through a package manager.
- Unzip the source code of in some folder. open a terminal in this folder.
- Make a new directory and go into this directory:

```
mkdir release
cd release
```

- Run `cmake`. Here is the line I used:

```
cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D
INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON ..
```

- Run `make`. This may take a few minutes (15 minutes on my computer).

```
make
```

- Finally, install.

```
sudo make install
```

And voilà !

You can check if it worked by opeing a Python console and typing

```
import cv2
print cv2.__version__
```

Advice: do not throw your `release` folder away. If later you have strange bugs with OpenCV involving `.so` files, just redo the `sudo make install` step.

```
.. image:: logo.png
    :width: 50%
    :align: center

..  MoviePy
..  =======



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.

Here it is in action (run in an IPython Notebook):

.. image:: demo_preview.jpeg
    :width: 500px
    :align: center

User Guide
------------

.. toctree::
   :maxdepth: 1

   install
   getting_started/getting_started
   gallery
   examples/examples
   docker
   opencv_instructions
   FAQ
   advanced_tools/advanced_tools
   ref/ref

Contribute !
--------------

MoviePy is an open source software originally written by Zulko_ and released under the MIT licence. It works on Windows, Mac, and Linux,
with Python 2 or Python 3. The code is hosted on Github_, where you can push improvements, report bugs and ask for help. There is also a
MoviePy forum on Reddit_ and a mailing list on librelist_ .

.. raw:: html

    <a href="https://twitter.com/share" class="twitter-share-button"
    data-text="MoviePy - Video editing with Python" data-size="large" data-hashtags="MoviePy">Tweet
    </a>
    <script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0],p=/^http:/.test(d.location)?'http':'https';
    if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src=p+'://platform.twitter.com/widgets.js';
```

```
    fjs.parentNode.insertBefore(js,fjs);}}(document, 'script', 'twitter-wjs');
    </script>

    <iframe type="text/html" src="https://ghbtns.com/github-btn.html?user=Zulko&repo=moviepy&type=watch&count=true&size=large"
    allowtransparency="true" frameborder="0" scrolling="0" width="152px" height="30px"></iframe>
..    <a href="https://github.com/Zulko/moviepy">
..    <img style="position: absolute; top: 0; right: 0; border: 0;"
..    src="https://s3.amazonaws.com/github/ribbons/forkme_right_red_aa0000.png"
..    alt="Fork me on GitHub"></a>

.. _PyPI: https://pypi.python.org/pypi/moviepy
.. _Zulko: https://github.com/Zulko/
.. _Stackoverflow: https://stackoverflow.com/
.. _Github: https://github.com/Zulko/moviepy
.. _Reddit: https://www.reddit.com/r/moviepy/
.. _librelist: mailto:moviepy@librelist.com
```

```
.. _gallery:


Gallery
========

Here are a few projects using MoviePy. The gallery will fill up as more people start using MoviePy (which is currently one year old). If
you have a nice project using MoviePy let us know !

Videos edited with Moviepy
---------------------------


The Cup Song Covers Mix
~~~~~~~~~~~~~~~~~~~~~~~

This mix of 60 covers of the Cup Song demonstrates the non-linear video editing capabilities of MoviePy. Here is `the (undocumented)
MoviePy code <https://nbviewer.ipython.org/github/Zulko/--video-editing---Cup-Song-Covers-Mix/blob/master/CupSongsCovers.ipynb>`_ that
generated the video.

.. raw:: html

        <div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; margin-bottom:30px; height: 0; overflow: hidden;
         margin-left: 5%;">
        <iframe width="560" height="315" src="https://www.youtube.com/embed/rIehsqqYFEM" frameborder="0" allowfullscreen></iframe>
        </div>

The (old) MoviePy reel video.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Made when MoviePy was a few weeks old and not as good as now. The code for most scenes can be found
in the :ref:`examples`.

.. raw:: html

       <div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; margin-bottom:30px; height: 0; overflow: hidden; margin-
left: 5%;">
       <iframe width="560" height="315" src="https://www.youtube.com/embed/zGhoZ4UBxEQ" frameborder="0" allowfullscreen>
       </iframe>
       </div>


Animations edited with MoviePy
------------------------------


GIFs made from videos
~~~~~~~~~~~~~~~~~~~~~~~~~~

This  `gifs tutorial
```

<https://zulko.github.io/blog/2014/01/23/making-animated-gifs-from-video-files-with-python/>`_ gives you the basics to make gifs from video files (cutting, croping, adding text...). The last example shows how to remove a (still) background to keep only the animated part of a video.

.. raw:: html

<img src="https://i.imgur.com/Fo2BxBK.gif" title="Hosted by imgur.com"
style="max-width:50%; height:'auto'; display:block; margin-left: auto;margin-right: auto; margin-bottom:30px;" />

Vector Animations
~~~~~~~~~~~~~~~~~~

This `vector animations tutorial <https://zulko.github.io/blog/2014/09/20/vector-animations-with-python/>`_ shows how to combine MoviePy with Gizeh to create animations:

.. raw:: html

<img src="https://i.imgur.com/2YdW9yf.gif"
style="max-width:50%; height:'auto'; display:block; margin-left: auto;margin-right: auto; margin-bottom:30px;" />

It is also possible to combine MoviePy with other graphic librairies like matplotlib, etc.

3D animations
~~~~~~~~~~~~~~~~~~

This `3d animation tutorial <https://zulko.github.io/blog/2014/11/13/things-you-can-do-with-python-and-pov-ray/>`_ shows how to combine MoviePy with Vapory, a library to render 3D scenes using the free ray-tracer POV-Ray

.. raw:: html

<img src="https://i.imgur.com/XN7e2IP.gif"
style="max-width:70%; height:'auto'; display:block; margin-left: auto;margin-right: auto; margin-bottom:30px;" />

With Vapory and MoviePy you can for instance embed a movie in a 3D scene:

.. raw:: html

<div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; height: 0; margin-bottom:30px; overflow: hidden;
margin-left: 5%;">
<iframe type="text/html" src="https://youtube.com/embed/M9R21SquDSk?rel=0" frameborder="0"
style="position: absolute; top: 0; bottom: 10; left: 0; width: 90%; height: 100%;">
</iframe>

Or render the result of this physics simulation made with PyODE (`script <https://gist.github.com/Zulko/f828b38421dfbee59daf>`_):

```
.. raw:: html

         <a href="https://imgur.com/2YdW9yf"><img src="https://i.imgur.com/TdhxwGz.gif"
         style="max-width:70%; height:'auto'; display:block; margin-left: auto;margin-right: auto; margin-bottom:30px;" /></a>


Or use `this script <https://gist.github.com/Zulko/b910c8b22e8e1c01fae6>`_ to make piano animations from MIDI files (which are some sort
of electronic sheet music):


.. raw:: html

        <div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; height: 0; margin-bottom:30px; overflow: hidden;
margin-left: 5%;">
        <iframe type="text/html" src="https://youtube.com/embed/tCqQhmuwgMg?rel=0" frameborder="0"
        style="position: absolute; top: 0; bottom: 10; left: 0; width: 90%; height: 100%;">
        </iframe>
        </div>

Data animations
---------------

This `data animation tutorial <https://zulko.github.io/blog/2014/11/13/things-you-can-do-with-python-and-pov-ray/>`_ shows how to use
MoviePy to animate the different Python vizualization libraries: Mayavi, Vispy, Scikit-image, Matplotlib, etc.


Scientific or technological projects
------------------------------------


Piano rolls transcription to sheet music
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

This `transcribing piano rolls blog post <https://zulko.github.io/blog/2014/02/12/transcribing-piano-rolls/>`_ explains how to transform a
video of a piano roll performance into playable sheet music. MoviePy is used for the frame-by-frame analysis of the piano roll video. The
last video is also edited with MoviePy:

.. raw:: html

        <div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; height: 0; margin-bottom:30px; overflow: hidden;
margin-left: 5%;">
        <iframe width="560" height="315" src="https://www.youtube.com/embed/V2XCJNZjm4w" frameborder="0" allowfullscreen>
        </iframe>
        </div>



Misc. Programs and Scripts using MoviePy
----------------------------------------
Kapwing
```

```
----------

`Kapwing <https://www.kapwing.com/>`_ is an online video meme generator. Content creators use Kapwing to add text around their videos,
which results in higher engagement / views on social media sites like Facebook. Kapwing's creation process is powered by MoviePy! MoviePy
is used to add the text, borders, and attribution directly to the uploaded videos.


.. raw:: html

        <a href="https://www.kapwing.com/videos/58d5d8b96c239227a1622319"><img src="https://cdn-e1.streamable.com/image/z35jz.jpg?
token=1491619273_2fa6161bc77a258112bd1b353f1d0fd217d7a8eb"
        style="max-width:70%; height:'auto'; display:block; margin-left: auto;margin-right: auto; margin-bottom:30px;" /></a>


Rinconcam
----------

`Rincomcam <http://www.rinconcam.com/month/2014-03>`_ is a camera which films surfers on the Californian beach of Point Rincon. At the end
of each day it cuts together a video, puts it online, and tweets it. Everything is entirely automatized with Python.
MoviePy is used to add transitions, titles and music to the videos.


.. raw:: html

        <a href="https://imgur.com/2YdW9yf"><img src="https://pbs.twimg.com/media/B2_NlnwCMAAingW.jpg"
        style="max-width:70%; height:'auto'; display:block; margin-left: auto;margin-right: auto; margin-bottom:30px;" /></a>


Videogrep
~~~~~~~~~~

Videogrep is a python script written by Sam Lavigne, that goes through the subtitle tracks of movies and makes supercuts based on what it
finds. For instance, here is an automatic supercut of every time the White House press secretary tells us what he can tell us:

.. raw:: html

       <div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; margin-bottom:30px; height: 0; overflow: hidden;
margin-left: 5%;">
        <iframe width="560" height="315" src="https://www.youtube.com/embed/D7pymdCU5NQ" frameborder="0" allowfullscreen>
        </iframe>
        </div>

Here are `Videogrep's introductory blog post
<http://lav.io/2014/06/videogrep-automatic-supercuts-with-python/>`_ and the Github `Videogrep page
<https://github.com/antiboredom/videogrep/>`_.

If you liked it, also have a look at these Videogrep-inspired projects:

This `Videogrep blog post <https://zulko.github.io/blog/2014/06/21/some-more-videogreping-with-python/>`_ attempts to cut a video
precisely at the beginning and end of sentences or words: ::
```

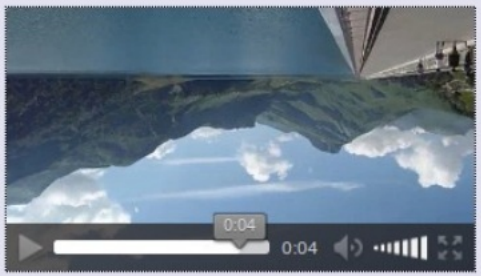
```
    words = ["Americans", "must", "develop", "open ", "source",
              " software", "for the", " rest ", "of the world",
              "instead of", " soldiers"]
    numbers = [3,0,4,3,4,0,1,2,0,1,0] # take clip number 'n'

    cuts = [find_word(word)[n] for (word,n) in zip(words, numbers)]
    assemble_cuts(cuts, "fake_speech.mp4")

.. raw:: html

        <div style="position: relative; padding-bottom: 56.25%; padding-top: 30px; margin-bottom:30px; height: 0; overflow: hidden;
margin-left: 5%;">
        <iframe type="text/html" src="https://youtube.com/embed/iWRYGULFd_c?rel=0" frameborder="0"
        style="position: absolute; top: 0; bottom: 10; left: 0; width: 90%; height: 100%;">
        </iframe>
        </div>


This `other post <https://zulko.github.io/blog/2014/07/04/automatic-soccer-highlights-compilations-with-python/>`_ uses MoviePy to
automatically cut together `all the highlights of a soccer game <http://youtu.be/zJtWPFX2bA0>`_, based on the fact that the crowd cheers
louder when something interesting happens. All in under 30 lines of Python:
```

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.

Here it is in action (run in an IPython Notebook):

```
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]:



User Guide

- Download and Installation
- Getting started with MoviePy
- Gallery
- Example Scripts

- Moviepy Docker
- So you want to install OpenCV 2.4.6 ?
- FAQ and troubleshooting
- Advanced tools
- Reference Manual

Contribute !

MoviePy is an open source software originally written by Zulko and released under the MIT licence. It works on Windows, Mac, and Linux, with Python 2 or Python 3. The code is hosted on Github, where you can push improvements, report bugs and ask for help. There is also a MoviePy forum on Reddit and a mailing list on librelist .

Tweet

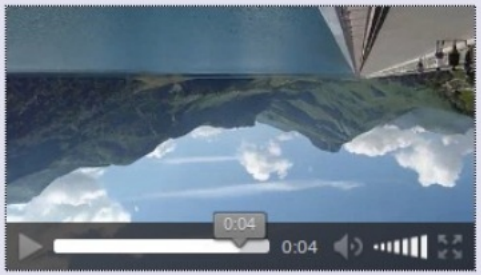
https://zulko.github.io/moviepy/_images/logo.png M

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.

Here it is in action (run in an IPython Notebook):

```
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]:



User Guide

- Download and Installation
- Getting started with MoviePy
- Gallery
- Example Scripts

- Moviepy Docker
- So you want to install OpenCV 2.4.6 ?
- FAQ and troubleshooting
- Advanced tools
- Reference Manual

Contribute !

MoviePy is an open source software originally written by Zulko and released under the MIT licence. It works on Windows, Mac, and Linux, with Python 2 or Python 3. The code is hosted on Github, where you can push improvements, report bugs and ask for help. There is also a MoviePy forum on Reddit and a mailing list on librelist .

Tweet 

Creating and exporting video clips

Video and audio clips are the central objects of MoviePy. In this section we present the different sorts of clips, how to create them, and how to write them to a file. For informations on modifying a clip (cuts, effects, etc.), see Clips transformations and effects. For how to put clips together see Mixing clips and to see how to preview clips before writing a file, refer to How to be efficient with MoviePy.

The following code summarizes the base clips that you can create with moviepy:

```
# VIDEO CLIPS
clip = VideoClip(make_frame, duration=4) # for custom animations (see below)
clip = VideoFileClip("my_video_file.mp4") # or .avi, .webm, .gif ...
clip = ImageSequenceClip(['image_file1.jpeg', ...], fps=24)
clip = ImageClip("my_picture.png") # or .jpeg, .tiff, ...
clip = TextClip("Hello !", font="Amiri-Bold", fontsize=70, color="black")
clip = ColorClip(size=(460,380), color=[R,G,B])

# AUDIO CLIPS
clip = AudioFileClip("my_audiofile.mp3") # or .ogg, .wav... or a video !
clip = AudioArrayClip(numpy_array, fps=44100) # from a numerical array
clip = AudioClip(make_frame, duration=3) # uses a function make_frame(t)
```

The best to understand these clips is to read the full documentation for each in the Reference Manual. The next sections In this section we see how to create clips, (for instance from video or audio files), how to mix them together, and how to write them to a file.

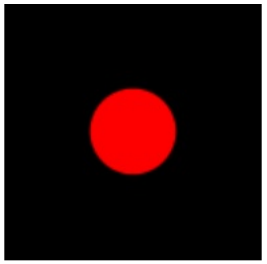
Categories of video clips

Video clips are the building blocks of longer videos. Technically, they are clips with a `clip.get_frame(t)` method which outputs a HxWx3 numpy array representing the frame of the clip at time . There are two main categories: animated clips (made with `VideoFileClip` and `VideoClip`) and unanimated clips which show the same picture for an a-priori infinite duration (`ImageClip`, `TextClip`,``ColorClip``). There are also special video clips call masks, which belong to the categories above but output greyscale frames indicating which parts of another clip are visible or not. A video clip can carry around an audio clip (`clip.audio`) which is its , and a mask clip.

VideoClip

`VideoClip` is the base class for all the other video clips in MoviePy. If all you want is to edit video files, you will never need it. This class is practical when you want to make animations from frames that are generated by another library. All you need is to define a function `make_frame(t)` which returns a HxWx3 numpy array (of 8-bits integers) representing the frame at time t. Here is an example with the graphics library Gizeh:

```
import gizeh
import moviepy.editor as mpy

def make_frame(t):
    surface = gizeh.Surface(128,128) # width, height
    radius = W*(1+ (t*(2-t))**2 )/6 # the radius varies over time
    circle = gizeh.circle(radius, xy = (64,64), fill=(1,0,0))
    circle.draw(surface)
    return surface.get_npimage() # returns a 8-bit RGB array

clip = mpy.VideoClip(make_frame, duration=2) # 2 seconds
clip.write_gif("circle.gif",fps=15)
```



Note that clips make with a do not have an explicit frame rate, so you must provide a frame rate (`fps` , frames er second) for `write_gif` and `write_videofile` , and more generally for any methods that requires iterating through the frames.

VideoFileClip

A VideoFileClip is a clip read from a video file (most formats are supported) or a GIF file. You load the video as follows:

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

Note that these clips will have an `fps` (frame per second) attribute, which will be transmitted if you do small modifications of the clip, and will be used by default in `write_videofile` , `write_gif` , etc. For instance:

```
myclip = VideoFileClip("some_video.avi")
print (myclip.fps) # prints for instance '30'
# Now cut the clip between t=10 and 25 secs. This conserves the fps.
myclip2 = myclip.subclip(10, 25)
myclip2.write_gif("test.gif") # the gif will have 30 fps
```

For more, see `VideoFileClip` .

ImageSequenceClip

This is a clip made from a series of images, you call it with

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

where `images_list` can be either a list of image names (that will be) in that order, a folder name (at which case all the image files in the folder will be played in alphanumerical order), or a list of frames (Numpy arrays), obtained for instance from other clips.

When you provide a folder name or list of file names, you can choose `load_images=True` to specify that all images should be loaded into the RAM. This is only interesting if you have a small number of images that will be each used more than once (e.g. if the images form a looping animation).

ImageClip

An ImageClip is a video clip that always displays the same image. You can create one as follows:

```
myclip = ImageClip("some_picture.jpeg")
myclip = ImageClip(somme_array) # a (height x width x 3) RGB numpy array
myclip = some_video_clip.to_ImageClip(t='01:00:00') # frame at t=1 hour.
```

For more, see `ImageClip`.

Two examples of ImageClip shown below are the TextClip and ColorClip

TextClip

Generating a TextClip requires to have ImageMagick installed and (for windows users) linked to MoviePy, see the installation instructions.

Here is how you make a textclip (you won't need all these options all the time):

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

The font can be any font installed on your computer, but ImageMagick will have specific names for it. For instance the Amiri font will be called `Amiri-Regular` while the Impact font will be called `Impact-Normal`. To get a list of the possible fonts, type `TextClip.list('font')`. To find all the font names related to a given font, use for instance

```
TextClip.search('Amiri', 'font') # Returns all font names containing Amiri
```

Note also that the use of a stroke (or contour) will not work well on small letters, so if you need a small text with a contour, it is better to generate a big text, then downsize it:

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

TextClips have many, many options: alignment, kerning (distance between the letters), stroke size, background, word wrapping, etc. see `TextClip` for more.

Mask clips

A mask is a special video clip which indicates which pixels will be visible when a video clip carrying this mask will be composed with other video clips (see Mixing clips). Masks are also used to define transparency when you export the clip as GIF file or as a PNG.

The fundamental difference between masks and standard clips is that standard clips output frames with 3 components (R-G-B) per pixel, comprised between 0 and 255, while a mask has just one composant per pixel, between 0 and 1 (1 indicating a fully visible pixel and 0 a transparent pixel). Seen otherwise, a mask is always in greyscale.

When you create or load a clip that you will use as a mask you need to declare it:

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

In the case of video and image files, if these are not already black and white they will be converted automatically.

Then you attach this mask to a clip (which must have the same dimensions) with `myclip.set_mask(maskclip)`.

Some image formats like PNG support transparency with an , which MoviePy will use as a mask:

```
myclip = ImageClip("image.png", transparent=True) # True is the default
myclip.mask # <- the alpha layer of the picture.
```

Any video clip can be turned into a mask with `clip.to_mask()`, and a mask can be turned to a standard RGB video clip with `my_mask_clip.to_RGB()`.

Masks are treated differently by many methods (because their frames are different) but you can do with a mask pretty much everything you can do with a standard clip: you can cut it, edit it, preview it, write it to a video file, make snapshots, etc.

Exporting video clips

Video files (.mp4, .webm, .ogv...)

To write a clip as a video file, use

```
my_clip.write_videofile("movie.mp4") # default codec: 'libx264', 24 fps
my_clip.write_videofile("movie.mp4",fps=15)
my_clip.write_videofile("movie.webm") # webm format
my_clip.write_videofile("movie.webm",audio=False) # don't render audio.
```

MoviePy has default codec names for the most common file extensions. If you want to use exotic formats or if you are not happy with the defaults you can provide the codec with `codec='mpeg4'` for instance. There are many many options when you are writing a video (bitrate, parameters of the audio writing, file size optimization, number of processors to use, etc.). Please refer to `write_videofile()` for more.

Sometimes it is impossible for MoviePy to guess the `duration` attribute of the clip (keep in mind that some clips, like ImageClips displaying a picture, have an infinite duration). Then, the `duration` must be set manually with `clip.set_duration`:

```
# Make a video showing a flower for 5 seconds
my_clip = Image("flower.jpeg") # has infinite duration
my_clip.write_videofile("flower.mp4") # Will fail ! NO DURATION !
my_clip.set_duration(5).write_videofile("flower.mp4") # works !
```

Animated GIFs

To write your video as an animated GIF, use

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

Note that this requires ImageMagick installed. Otherwise you can also create the GIF with ffmpeg by adding the option `program='ffmpeg'`, it will be much faster but won't look as nice and won't be optimized.

There are many options to optimize the quality and size of a gif. Please refer to `write_gif()`.

Note that for editing gifs the best way is to preview them in the notebook as explained here: ipython_display

For examples of use, see this blog post for informations on making GIFs from video files, and this other post for GIF animations with vector graphics.

Export images

You can write a frame to an image file with

```
myclip.save_frame("frame.png") # by default the first frame is extracted
myclip.save_frame("frame.jpeg", t='01:00:00') # frame at time t=1h
```

If the clip has a mask it will be exported as the alpha layer of the image unless you specify `withmask=False`.

Download and Installation

Installation

Method with pip: if you have `pip` installed, just type this in a terminal (it will install ez_setup if you don't already have it)

```
(sudo) pip install moviepy
```

If you have neither `setuptools` nor `ez_setup` installed the command above will fail, is this case type this before installing:

```
(sudo) pip install ez_setup
```

Method by hand: download the sources, either on PyPI or (if you want the development version) on Github, unzip everything in one folder, open a terminal and type

```
(sudo) python setup.py install
```

MoviePy depends on the Python modules Numpy, imageio, Decorator, and tqdm, which will be automatically installed during MoviePy's installation. It should work on Windows/Mac/Linux, with Python 2.7+ and 3 ; if you have trouble installing MoviePy or one of its dependencies, please

provide feedback !

MoviePy depends on the software FFMPEG for video reading and writing. You don't need to worry about that, as FFMPEG should be automatically downloaded/installed by ImageIO during your first use of MoviePy (it takes a few seconds). If you want to use a specific version of FFMPEG, you can set the FFMPEG_BINARY environment variable See `moviepy/config_defaults.py` for details.

Other optional but useful dependencies

ImageMagick is not strictly required, only if you want to write texts. It can also be used as a backend for GIFs but you can do GIFs with MoviePy without ImageMagick.

Once you have installed it, ImageMagick will be automatically detected by MoviePy, **except on Windows !**. Windows user, before installing MoviePy by hand, go into the `moviepy/config_defaults.py` file and provide the path to the ImageMagick binary called . It should look like this

```
IMAGEMAGICK_BINARY = "C:\\Program Files\\ImageMagick_VERSION\\magick.exe"
```

You can also set the IMAGEMAGICK_BINARY environment variable See `moviepy/config_defaults.py` for details.

If you are using an older version of ImageMagick, keep in mind the name of the executable is not `magick.exe` but `convert.exe`. In that case, the IMAGEMAGICK_BINARY property should be `C:\\Program Files\\ImageMagick_VERSION\\convert.exe`

PyGame is needed for video and sound previews (useless if you intend to work with MoviePy on a server but really essential for advanced video editing).

For advanced image processing you will need one or several of these packages. For instance using the method `clip.resize` requires that at least one of Scipy, PIL, Pillow or OpenCV are installed.

- The Python Imaging Library (PIL) or, better, its branch Pillow .
- Scipy (for tracking, segmenting, etc.), and can be used for resizing video clips if PIL and OpenCV aren't installed on your computer.
- Scikit Image may be needed for some advanced image manipulation.
- OpenCV 2.4.6 or more recent (provides the package `cv2`) or more recent may be needed for some advanced image manipulation.

If you are on linux, these packages will likely be in your repos.