Python Eulerian Video Magnification

Release 0.4.2

Mar 05, 2020

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Eulerian Video Magnification for Python

This is a python implementation of Eulerian Video Magnification ([Eulerian Video Magnification for Revealing Subtle Changes in the World](http://people.csail.mit.edu/mrub/evm/)). >Our goal is to reveal temporal variations in videos that are difficult or impossible to see with the naked eye and display them in an indicative manner. Our method, which we call Eulerian Video Magnification, takes a standard video sequence as input, and applies spatial decomposition, followed by temporal filtering to the frames. The resulting signal is then amplified to reveal hidden information.Using our method, we are able to visualize the flow of blood as it fills the face and also to amplify and reveal small motions. Our technique can run in real time to show phenomena occurring at temporal frequencies selected by the user.

This is a fork from [flyingzhao/PyEVM](https://github.com/flyingzhao/PyEVM) as a basis for own work. It now has an operational command line interface and is install able.

- Free software: BSD 2-Clause License

1.1 Installation

Up until now it is not available with PyPI, but if it will be you could use this code to install it.

```
pip install PyEVM
```

You can install the in-development version with:

```
pip install https://github.com/vgoehler/PyEVM/archive/master.zip
```

needed libraries (that get automatically installed) are:

- numpy (>=1.17.4)
- opencv-python (>=4.1.2.30)
- scipy (>=1.3.3)

1.2 Running

Navigate to sources directory and use

```
python3 -mpython_eulerian_video_magnification inputfile.video
```

if you just want to execute the code.

1.3 Usage

optional arguments:

`-h, --help`	show this help message and exit

system arguments:

`input`	the input video file to work on
`-o [O]`	output-folder
`--color_suffix [COLOR_SUFFIX]`	the suffix to use for color modified result files
`--motion_suffix [MOTION_SUFFIX]`	the suffix to use for motion modified result files
`--log {debug,info,warning,error, critical}`	log level

parameters:

`-m {color,motion}`	mode
`-c LOW, --low LOW`	low parameter (creek)
`-p HIGH, --high HIGH`	high parameter (peek)
`-l LEVELS, --levels LEVELS`	levels parameter
`-a AMPLIFICATION, --amplification AMPLIFICATION`	amplification parameter

1.4 Documentation

https://PyEVM.readthedocs.io/

1.5 Development

To run all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 2

Installation

At the command line:

```
pip install PyEVM
```

CHAPTER 3

Usage

To use Python Eulerian Video Magnification in a project:

```
import python_eulerian_video_magnification
```

CHAPTER 4

Reference

4.1 python_eulerian_video_magnification

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

Python Eulerian Video Magnification could always use more documentation, whether as part of the official Python Eulerian Video Magnification docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/vgoehler/PyEVM/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *PyEVM* for local development:

1. Fork PyEVM (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:vgoehler/PyEVM.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

Authors

- Volker G Göhler - https://github.com/vgoehler

6.1 EVM Project Forked from

- flyingzhao - https://github.com/flyingzhao/PyEVM

CHAPTER 7

Changelog

7.1 0.1.0 (2020-01-02)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search