

---

# **mplview Documentation**

***Release 0.0.3+0.g5180f94.dirty***

**John Kirkham**

**Nov 12, 2017**



---

## Contents

---

<b>1</b>	<b>mplview</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Usage</b>	<b>7</b>
<b>4</b>	<b>API</b>	<b>9</b>
<b>5</b>	<b>Contributing</b>	<b>13</b>
<b>6</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



Contents:



A simple, embeddable Matplotlib-based image viewer.

- Free software: BSD 3-Clause
- Documentation: <https://mplview.readthedocs.io>.

## 1.1 Features

- TODO

## 1.2 Credits

This package was created with [Cookiecutter](#) and the [nanshe-org/nanshe-cookiecutter](#) project template.



### 2.1 Stable release

To install `mplview`, run this command in your terminal:

```
$ pip install mplview
```

This is the preferred method to install `mplview`, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

### 2.2 From sources

The sources for `mplview` can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/jakirkham/mplview
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/jakirkham/mplview/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```



## CHAPTER 3

---

### Usage

---

To use mplview in a project:

```
import mplview
```



## 4.1 mplview package

### 4.1.1 Submodules

#### **mplview.core module**

**class** `mplview.core.MatplotlibViewer` (\*args, \*\*kwargs)

Bases: `matplotlib.figure.Figure`

Provides a way to interact with numpy arrays pulled from neuron images.

Wraps a Matplotlib figure instance.

**color\_range\_update** (*vmin*, *vmax*)

Handles an update to the vmin and vmax range based on the selection provided.

#### **Parameters**

- **the min value selected** (*vmin*) –
- **the max value selected** (*vmax*) –

**format\_coord** (*x*, *y*)

Include intensity when showing coordinates during mouseover.

#### **Parameters**

- **x** (*float*) – cursor's x position within the image.
- **y** (*float*) – cursor's y position within the image.

**Returns** coordinates and intensity if it can be gotten.

**Return type** `str`

**get\_image** (*i=None*)

Gets the current image or the image if it is a projection.

**Parameters** *i* (*int*) – image to retrieve (defaults to selection).

**Returns** the current image.

**Return type** `numpy.ndarray`

**set\_images** (*new\_neuron\_images*, *cmap*=<*matplotlib.colors.LinearSegmentedColormap* object>, *use\_matshow*=*False*, *vmin*=*None*, *vmax*=*None*)  
Sets the images to be viewed.

**Parameters** *new\_neuron\_images* (*numpy.ndarray*) – array of images (first index is which image)

**time\_update** ()

Method to be called by the SequenceNavigator when the time changes. Updates image displayed.

**class** `mplview.core.SequenceNavigator` (*fig*, *max\_time*, *min\_time*=0, *time\_step*=1, *ax\_color*='lightgoldenrodyellow', *hovercolor*='0.975')

Bases: `object`

**begin\_time** (*event*)

Sets time to *min\_time*.

**Parameters** `Matplotlib` event that caused the call to this callback.  
(*event*) –

**disconnect** (*cid*)

Disconnects the given *cid* from being notified of time updates.

**Parameters** ID of callback to pull (*cid*) –

**end\_time** (*event*)

Sets time to *max\_time*.

**Parameters** `Matplotlib` event that caused the call to this callback.  
(*event*) –

**next\_time** (*event*)

Sets time to one *time\_step* after.

**Parameters** `Matplotlib` event that caused the call to this callback.  
(*event*) –

**normalize\_val** (*val*)

Takes the time value and normalizes it to fit within the range. Then, makes sure it is a discrete number of steps from the *min\_time*.

**Parameters** float position from the slider bar to correct (*val*) –

**Returns** the normalized value.

**Return type** `int`

**on\_time\_update** (*func*)

Registers a callback function for notification when the time is updated.

**Parameters** *func* (*callable*) – function call when the time is updated

**Returns**

a callback ID or *cid* to allow pulling the callback when no longer necessary.

**Return type** `int`

**prev\_time** (*event*)

Sets time to one *time\_step* prior.

Parameters Matplotlib event that caused the call to this callback.  
(*event*) –

**time\_update** (*val*)

Takes the time value and normalizes it within the range if it does not fit.

Parameters float position from slider bar to move to (*val*) –



Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 5.1 Types of Contributions

### 5.1.1 Report Bugs

Report bugs at <https://github.com/jakirkham/mplview/issues>.

If you are 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.

### 5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

### 5.1.4 Write Documentation

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

### 5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/jakirkham/mplview/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 contributions are welcome :)

## 5.2 Get Started!

Ready to contribute? Here's how to set up *mplview* for local development.

1. Fork the *mplview* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/mplview.git
```

3. Install your local copy into an environment. Assuming you have conda installed, this is how you set up your fork for local development (on Windows drop *source*). Replace “<some version>” with the Python version used for testing.:

```
$ conda create -n mplviewenv python="<some version>"
$ source activate mplviewenv
$ python setup.py develop
```

4. Create a branch for local development:

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

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions:

```
$ flake8 mplview tests
$ python setup.py test or py.test
```

To get flake8, just conda install it into your environment.

6. 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
```

7. Submit a pull request through the GitHub website.

## 5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5, and 3.6. Check [https://travis-ci.org/jakirkham/mplview/pull\\_requests](https://travis-ci.org/jakirkham/mplview/pull_requests) and make sure that the tests pass for all supported Python versions.

## 5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_mplview
```



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### m

`mplview`, 9

`mplview.core`, 9



## B

`begin_time()` (`mplview.core.SequenceNavigator` method),  
10

## C

`color_range_update()` (`mplview.core.MatplotlibViewer`  
method), 9

## D

`disconnect()` (`mplview.core.SequenceNavigator` method),  
10

## E

`end_time()` (`mplview.core.SequenceNavigator` method),  
10

## F

`format_coord()` (`mplview.core.MatplotlibViewer`  
method), 9

## G

`get_image()` (`mplview.core.MatplotlibViewer` method), 9

## M

`MatplotlibViewer` (class in `mplview.core`), 9

`mplview` (module), 9

`mplview.core` (module), 9

## N

`next_time()` (`mplview.core.SequenceNavigator` method),  
10

`normalize_val()` (`mplview.core.SequenceNavigator`  
method), 10

## O

`on_time_update()` (`mplview.core.SequenceNavigator`  
method), 10

## P

`prev_time()` (`mplview.core.SequenceNavigator` method),  
10

## S

`SequenceNavigator` (class in `mplview.core`), 10

`set_images()` (`mplview.core.MatplotlibViewer` method),  
10

## T

`time_update()` (`mplview.core.MatplotlibViewer` method),  
10

`time_update()` (`mplview.core.SequenceNavigator`  
method), 11