

---

# **scikit-surgerytrackervisualisation Documentation**

**Stephen Thompson**

**Feb 24, 2022**



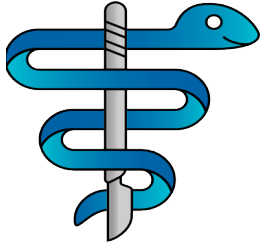
---

## Contents

---

<b>1</b>	<b>Developing</b>	<b>3</b>
<b>2</b>	<b>Installing</b>	<b>5</b>
<b>3</b>	<b>Licensing and copyright</b>	<b>7</b>
<b>4</b>	<b>Acknowledgements</b>	<b>9</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>





Author: Stephen Thompson

scikit-surgerytrackervisualisation is part of the [SciKit-Surgery](#) software project, developed at the [Wellcome EPSRC Centre for Interventional and Surgical Sciences](#), part of [University College London \(UCL\)](#).

scikit-surgerytrackervisualisation supports Python 2.7 and Python 3.6.

scikit-surgerytrackervisualisation implements a basic interface for showing tracking output from a SciKit-Surgery tracker.

```
python scikit-surgerytrackervisualisation.py
python scikit-surgerytrackervisualisation.py --config config.json
```

Please explore the project structure, and implement your own functionality.



### 1.1 Cloning

You can clone the repository using the following command:

```
git clone https://github.com/SciKit-Surgery/scikit-surgerytrackervisualisation
```

### 1.2 Running tests

Pytest is used for running unit tests:

```
pip install pytest  
python -m pytest
```

### 1.3 Linting

This code conforms to the PEP8 standard. Pylint can be used to analyse the code:

```
pip install pylint  
pylint --rcfile=tests/pylintrc scikit-surgerytrackervisualisation
```





You can pip install directly from the repository as follows:

```
pip install git+https://github.com/SciKit-Surgery/scikit-surgerytrackervisualisation
```

## 2.1 Contributing

Please see the [contributing guidelines](#).

## 2.2 Useful links

- [Source code repository](#)
- [Documentation](#)



## CHAPTER 3

---

### Licensing and copyright

---

Copyright 2019 University College London. scikit-surgerytrackervisualisation is released under the BSD-3 license. Please see the [license file](#) for details.



Supported by Wellcome and EPSRC.

## 4.1 Dependency Graph

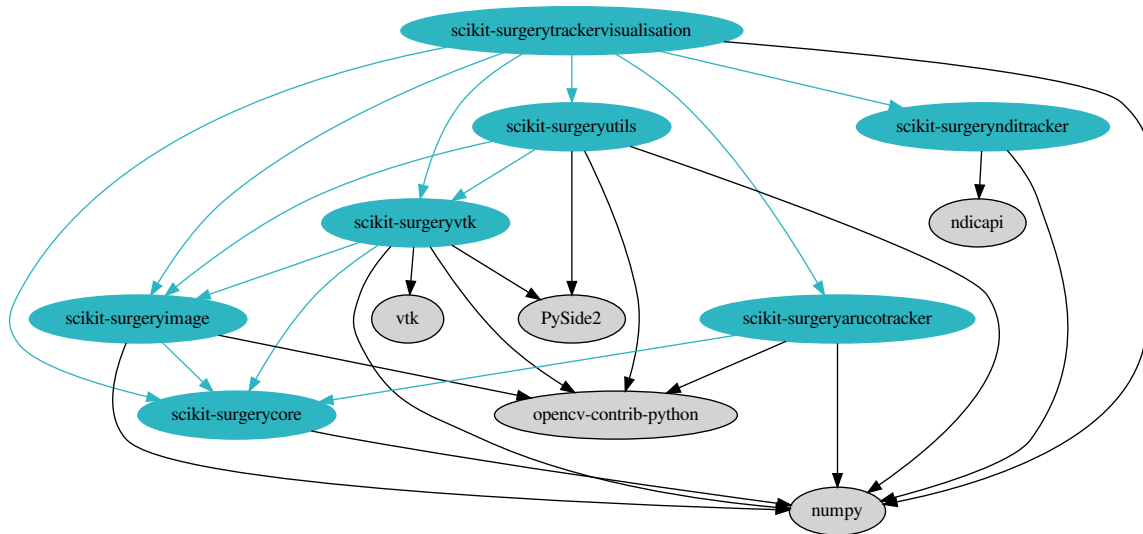


Fig. 1: The dependency graph

## 4.2 Requirements for scikit-surgerytrackervisualisation

This is the software requirements file for scikit-surgerytrackervisualisation, part of the SNAPPY project. The requirements listed below should define what scikit-surgerytrackervisualisation does. Each requirement can be matched to a unit test that checks whether the requirement is met.

### 4.2.1 Requirements

ID	Description	Test
0000	Module has a help page	pylint, see tests/pylint.rc and tox.ini
0001	Functions are documented	pylint, see tests/pylint.rc and tox.ini
0002	Package has a version number	No test yet, handled by git.

## 4.3 stable

### 4.3.1 sksurgerytrackervisualisation package

#### Subpackages

#### sksurgerytrackervisualisation.algorithms package

#### Submodules

#### sksurgerytrackervisualisation.algorithms.algorithms module

Algorithms used by the trackervisualisation module

`sksurgerytrackervisualisation.algorithms.algorithms.configure_tracker` (*config*)

Configures the tracking system. :param: A dictionary containing configuration data :return: The tracker object  
:raises: KeyError if no tracker entry in config

`sksurgerytrackervisualisation.algorithms.algorithms.make_offset_matrix` (*model\_config*)  
generates an offset (or handeye) matrix

**Param** Model configuration

**Returns** If valid offset specified, returns a 4x4 offset matrix, if no offset, returns identity.

**Raises** ValueError

`sksurgerytrackervisualisation.algorithms.algorithms.np2vtk` (*mat*)

Converts a Numpy array to a vtk matrix :param: the number array, should be 4x4 :return: a vtk 4x4 matrix  
:raises: ValueError when matrix is not 4x4

`sksurgerytrackervisualisation.algorithms.algorithms.populate_models` (*model\_config*)

**Parses a model configuration dictionary, returning** a list of vtk actors and associated port handles

**param** model config a list of dictionaries, one for each model dictionary entries are: name : a descriptive name port handle : the port handle of the associated tracker load : True if model is to be loaded from file filename : if load is true the filename to load from source : supported values are cylinder, sphere, cone colour : the rgb colour to use for the actor height : the height of the cylinder or cone radius : the diameter of the cylinder, cone, or sphere

**Returns** a list of dictionaries, one for each model

**Returns** port\_handles

**Returns** actors

**Returns** transform\_managers

**Raises** KeyError if asked to load model without filename

## sksurgerytrackervisualisation.algorithms.background\_image module

A class to provide the background image

```
class sksurgerytrackervisualisation.algorithms.background_image.OverlayBackground (config)
    Bases: object

    Provides the background image for the overlay window.

    next_image ()
        Returns a background image. The behaviour is determined by the configuration dictionary used at init.
```

## sksurgerytrackervisualisation.algorithms.icp module

Algorithms for doing Iterative Closest Point

```
sksurgerytrackervisualisation.algorithms.icp.vtk_icp (source, target, locator=None,
                                                    max_iterations=100,
                                                    max_landmarks=1000,
                                                    check_mean_distance=False,
                                                    maxi-
                                                    mum_mean_distance=0.001)

    An iterative closest point algorithm, delegating to vtk. Target is a point set, source is a point cloud
```

## Module contents

### sksurgerytrackervisualisation.overlay\_app package

#### Submodules

### sksurgerytrackervisualisation.overlay\_app.overlay module

Main loop for tracking visualisation

```
class sksurgerytrackervisualisation.overlay_app.overlay.OverlayApp (config)
    Bases: sksurgeryutils.common_overlay_apps.OverlayBaseApp

    Inherits from OverlayBaseApp, adding code to move vtk models based on input from a scikitsurgery tracker.
    Adds a function to detect a key press event, ("g") and add points to a point cloud.

    key_press_event (_obj_not_used, _ev_not_used)
        Handles a key press event

    update ()
        Update the background renderer with a new frame, move the model and render
```

## Module contents

### skurgerytrackervisualisation.shapes package

#### Submodules

#### skurgerytrackervisualisation.shapes.cone module

VTK pipeline to represent a surface model via a vtkPolyData.

```
class skurgerytrackervisualisation.shapes.cone.VTKConeModel (height,      radius,
                                                         colour,      name,
                                                         visibility=True,
                                                         opacity=1.0)
```

Bases: skurgeryvtk.models.vtk\_surface\_model.VTKSurfaceModel

Class to create a VTK surface model of a cone.

#### skurgerytrackervisualisation.shapes.dynamic\_point\_cloud module

Class to represent a point cloud via a vtkPolyData, with the ability to dynamically add points

```
class skurgerytrackervisualisation.shapes.dynamic_point_cloud.VTKDynamicPointCloud (colour,
                                                                                   vis-
                                                                                   i-
                                                                                   bil-
                                                                                   ity=True,
                                                                                   opac-
                                                                                   ity=1.0)
```

Bases: skurgeryvtk.models.vtk\_base\_model.VTKBaseModel

Class to represent a point cloud via a vtkPolyData, with the ability to dynamically add points

**add\_point** (*point*)

Adds a point to the point cloud and updates the vtk actor to show the complete point cloud

**Param** A 3 tuple representing the point coordinate

**get\_polydata** ()

Returns a polydata consisting of the poind cloud

#### skurgerytrackervisualisation.shapes.sphere module

VTK pipeline to represent a surface model via a vtkPolyData.

```
class skurgerytrackervisualisation.shapes.sphere.VTKSphereModel (radius,
                                                                    colour,
                                                                    name,  visi-
                                                                    bility=True,
                                                                    opacity=1.0)
```

Bases: skurgeryvtk.models.vtk\_surface\_model.VTKSurfaceModel

Class to create a VTK surface model of a sphere.



## Module contents

### sk surgerytrackervisualisation.ui package

#### Submodules

#### sk surgerytrackervisualisation.ui.sk surgerytrackervisualisation\_cl module

Command line processing

`sk surgerytrackervisualisation.ui.sk surgerytrackervisualisation_cl.main(args=None)`  
Entry point for scikit-surgerytrackervisualisation application

#### sk surgerytrackervisualisation.ui.sk surgerytrackervisualisation\_demo module

Tracker visualisation demo module

`sk surgerytrackervisualisation.ui.sk surgerytrackervisualisation_demo.run(configfile)`  
Run the application

## Module contents

scikit-surgerytrackervisualisation

## Module contents

scikit-surgerytrackervisualisation

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



### S

- sksurgerytrackervisualisation, [13](#)
- sksurgerytrackervisualisation.algorithms,
  - [11](#)
- sksurgerytrackervisualisation.algorithms.algorithms,
  - [10](#)
- sksurgerytrackervisualisation.algorithms.background\_image,
  - [11](#)
- sksurgerytrackervisualisation.algorithms.icp,
  - [11](#)
- sksurgerytrackervisualisation.overlay\_app,
  - [12](#)
- sksurgerytrackervisualisation.overlay\_app.overlay,
  - [11](#)
- sksurgerytrackervisualisation.shapes,
  - [13](#)
- sksurgerytrackervisualisation.shapes.cone,
  - [12](#)
- sksurgerytrackervisualisation.shapes.dynamic\_point\_cloud,
  - [12](#)
- sksurgerytrackervisualisation.shapes.sphere,
  - [12](#)
- sksurgerytrackervisualisation.ui, [13](#)
- sksurgerytrackervisualisation.ui.sksurgerytrackervisualisation\_cl,
  - [13](#)
- sksurgerytrackervisualisation.ui.sksurgerytrackervisualisation\_demo,
  - [13](#)



## A

`add_point()` (*skurgerytrackervisualisation.shapes.dynamic\_point\_cloud.VTKDynamicPointCloud* method), 12

## C

`configure_tracker()` (*in module skurgerytrackervisualisation.algorithms.algorithms*), 10

## G

`get_polydata()` (*skurgerytrackervisualisation.shapes.dynamic\_point\_cloud.VTKDynamicPointCloud* method), 12

## K

`key_press_event()` (*skurgerytrackervisualisation.overlay\_app.overlay.OverlayApp* method), 11

## M

`main()` (*in module skurgerytrackervisualisation.ui.skurgerytrackervisualisation\_cl*), 13

`make_offset_matrix()` (*in module skurgerytrackervisualisation.algorithms.algorithms*), 10

## N

`next_image()` (*skurgerytrackervisualisation.algorithms.background\_image.OverlayBackgroundImage* method), 11

`np2vtk()` (*in module skurgerytrackervisualisation.algorithms.algorithms*), 10

## O

`OverlayApp` (*class in skurgerytrackervisualisation.overlay\_app.overlay*), 11

`OverlayBackground` (*class in skurgerytrackervisualisation.algorithms.background\_image*), 11

## P

`populate_models()` (*in module skurgerytrackervisualisation.algorithms.algorithms*), 10

## R

`run()` (*in module skurgerytrackervisualisation.ui.skurgerytrackervisualisation\_demo*), 13

## S

`skurgerytrackervisualisation` (*module*), 13

`skurgerytrackervisualisation.algorithms` (*module*), 11

`skurgerytrackervisualisation.algorithms.algorithms` (*module*), 10

`skurgerytrackervisualisation.algorithms.background_image` (*module*), 11

`skurgerytrackervisualisation.algorithms.icp` (*module*), 11

`skurgerytrackervisualisation.overlay_app` (*module*), 12

`skurgerytrackervisualisation.overlay_app.overlay` (*module*), 11

`skurgerytrackervisualisation.shapes` (*module*), 13

`skurgerytrackervisualisation.shapes.cone` (*module*), 12

`skurgerytrackervisualisation.shapes.dynamic_point_cloud` (*module*), 12

`skurgerytrackervisualisation.shapes.sphere` (*module*), 12

`skurgerytrackervisualisation.ui` (*module*), 13

`skurgerytrackervisualisation.ui.skurgerytrackervisualisation_demo` (*module*), 13

`skurgerytrackervisualisation.ui.skurgerytrackervisualisation_demo` (*module*), 13

## U

`update()` (*skurgerytrackervisualisation*)

*tion.overlay\_app.overlay.OverlayApp* method),  
[11](#)

## V

`vtk_icp()` (in module *sksurgerytrackervisualisation.algorithms.icp*), [11](#)

`VTKConeModel` (class in *sksurgerytrackervisualisation.shapes.cone*), [12](#)

`VTKDynamicPointCloud` (class in *sksurgerytrackervisualisation.shapes.dynamic\_point\_cloud*),  
[12](#)

`VTKSphereModel` (class in *sksurgerytrackervisualisation.shapes.sphere*), [12](#)