shwirl Documentation

Release 0.1.3

Dany Vohl

Feb 13, 2023

Contents:

1 About 3
- 1.1 GPUs . . . 3
- 1.2 Issues, requests and general inquiries . . . 3
- 1.3 Want to contribute? . . . 4

2 Installation 5
- 2.1 Prerequisites . . . 5
- 2.2 Example installation procedure . . . 5
  - 2.2.1 Install with pip . . . 5
- 2.3 Python 2.7 / PyQt4 users . . . 6
- 2.4 Installing PyQt5 from scratch (example) . . . 6

3 Usage and snapshots 7
- 3.1 Graphical User Interface (GUI) . . . 7
- 3.2 Visualisation and colour bar . . . 8
- 3.3 Widgets . . . 8
- 3.4 Cameras and transform . . . 8
- 3.5 Colour . . . 10
- 3.6 Filter . . . 12
- 3.7 Smooth . . . 14
- 3.8 Export . . . 14
- 3.9 Metadata . . . 15
- 3.10 Computing an emission line ratio in 3D space . . . 16
- 3.11 Axis labels . . . 16

4 Example data 17
- 4.1 Neutral Hydrogen (HI) cube of NGC 2903 . . . 17
- 4.2 Cube of the carbon monoxide (CO) in the Antennae galaxies . . . 17

Shwirl is a custom standalone Python program to visualise spectral data cubes with ray-tracing volume rendering. The program has been developed to investigate transfer functions and graphics shaders as enablers for scientific visualisation of astronomical data. Details about the transfer functions and shaders developed and implemented in **shwirl** can be found in a full length article by Vohl, Fluke, Barnes & Hassan (2017).

Disclaimer: While the software is available for download and ready to visualise data, this is not intended as a full software release just yet.

Shwirl is licensed under the terms of the (new) BSD license.

CHAPTER 1

About

Shwirl is a custom standalone Python program to visualise spectral data cubes with ray-tracing volume rendering. The program has been developed to investigate transfer functions and graphics shaders as enablers for scientific visualisation of astronomical data. Details about transfer functions and shaders developed and implemented in **shwirl** can be found in a full length article by Vohl, Fluke, Barnes & Hassan (2017).

A transfer function is an arbitrary function that combines volumetric elements (or voxels) to set the colour, intensity, or transparency level of each pixel in the final image. A graphics shader is an algorithmic kernel used to compute several properties of the final image such as colour, depth, and/or transparency. Shaders are particularly suited to computing transfer functions, and are an integral part of the graphics pipeline on Graphics Processing Units.

The code is available at https://github.com/macrocosme/shwirl. The program is built on top of Astropy to handle FITS files and World Coordinate System, Qt (and PyQt) for the user interface, and VisPy, an object-oriented Python visualisation library binding onto OpenGL. We implemented the algorithms in the fragment shader using the GLSL language.

The software has been used on Linux, Mac, and Windows machines, including remote desktop on cloud computing infrastructure.

1.1 GPUs

Up to now, we have tested the code on NVIDIA GeForce GTX TITAN X, NVIDIA GeForce GT 750M, and NVIDIA GRID K1. Rendering speed will vary depending on your hardware.

1.2 Issues, requests and general inquiries

Please send issues, feature requests and/or general inquiries to Dany Vohl via http://macrocosme.github.io/#contact. You can also simply open a new issue on github directly.

1.3 Want to contribute?

As mentioned above, shwirl is not intended to be a finished product yet. If you would like to contribute, GitHub pull requests are welcomed.

CHAPTER 2

Installation

2.1 Prerequisites

Shwirl utilises Astropy to handle FITS files and World Coordinate System, Qt (and PyQt) for the user interface, and VisPy, an object-oriented Python visualisation library binding onto OpenGL.

These are pre-requisites to be able to use the software.

In particular, current version relies on Python 3 and PyQt5. A bundled version of VisPy is included in the *extern* repository as custom modifications have been made that are not readily available via the official version.

2.2 Example installation procedure

Depending on your operating system and your current configuration, installation steps may vary. In general, you can install it using the following:

First, install Qt5. It can be downloaded via the Qt website. You also need PyQt5, which can be installed via a package manager like pip, brew, etc. E.g.

```
pip3 install PyQt5
```

2.2.1 Install with pip

You can install **shwirl** with pip:

```
pip3 install shwirl
```

Once installed, in your terminal, you can launch **shwirl** by typing:

```
shwirl
```

2.3 Python 2.7 / PyQt4 users

PyQt4 (and Python 2.7) are also supported.

2.4 Installing PyQt5 from scratch (example)

N.B. The following will vary depending on your OS and your different setings.

Download sip here.

```
cd sip-4.19.1/
python3 configure.py
sudo make
sudo make install
```

Download PyQt5 from the same location.

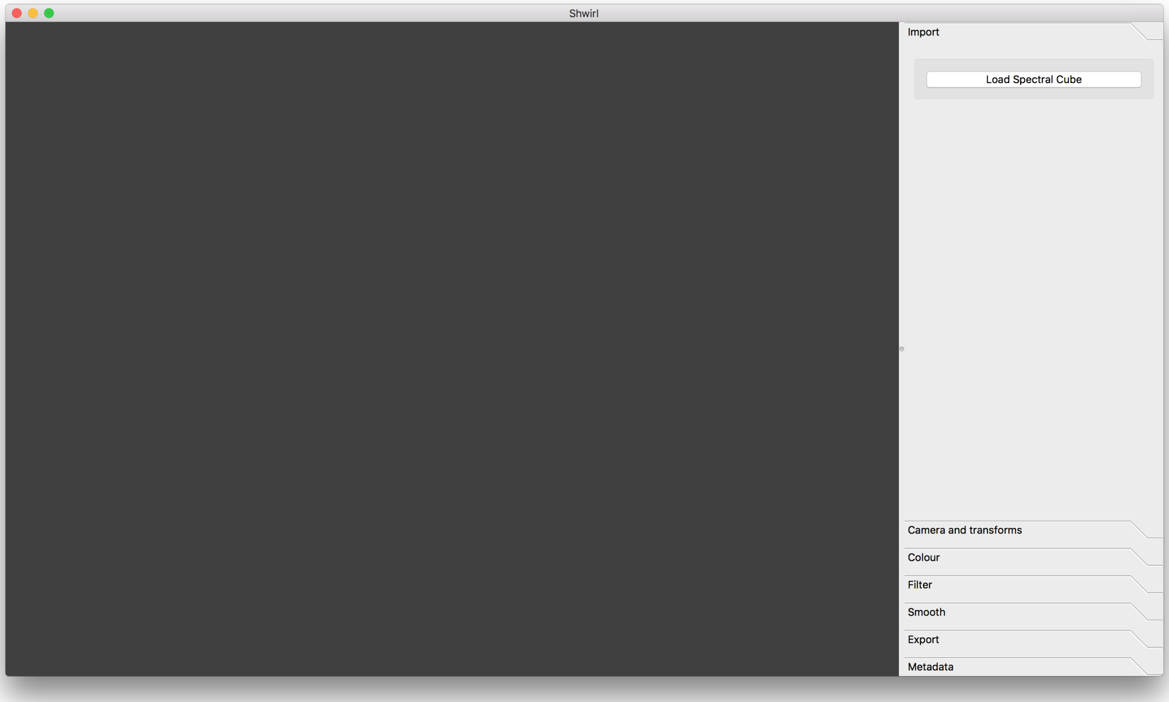
```
cd PyQt5_gpl-5.8.1/
sudo python3 configure.py --qmake /home/ubuntu/Qt5.7.0/5.7/gcc_64/bin/qmake --sip-
↪incdir /usr/include/python3.4m py_inc_dir=/usr/include/python3.4m
sudo make
sudo make install
```

CHAPTER 3

Usage and snapshots

3.1 Graphical User Interface (GUI)

Upon start of the software, you will need to open a new spectral cube (currently supports FITS 3D image files only) using the GUI by clicking the *Load Spectral Cube* button at the top right corner.



3.2 Visualisation and colour bar

Once you have successfully loaded a spectral cube, the 3D volume can be rotated, panned and zoomed using a mouse or a trackpad. To rotate the cube, simply click and move. Panning can be achieved by click-n-move while pressing the *shift* key. Zooming in and out is achieved using the wheel, or using trackpad interactions like using two fingers (depending on device).

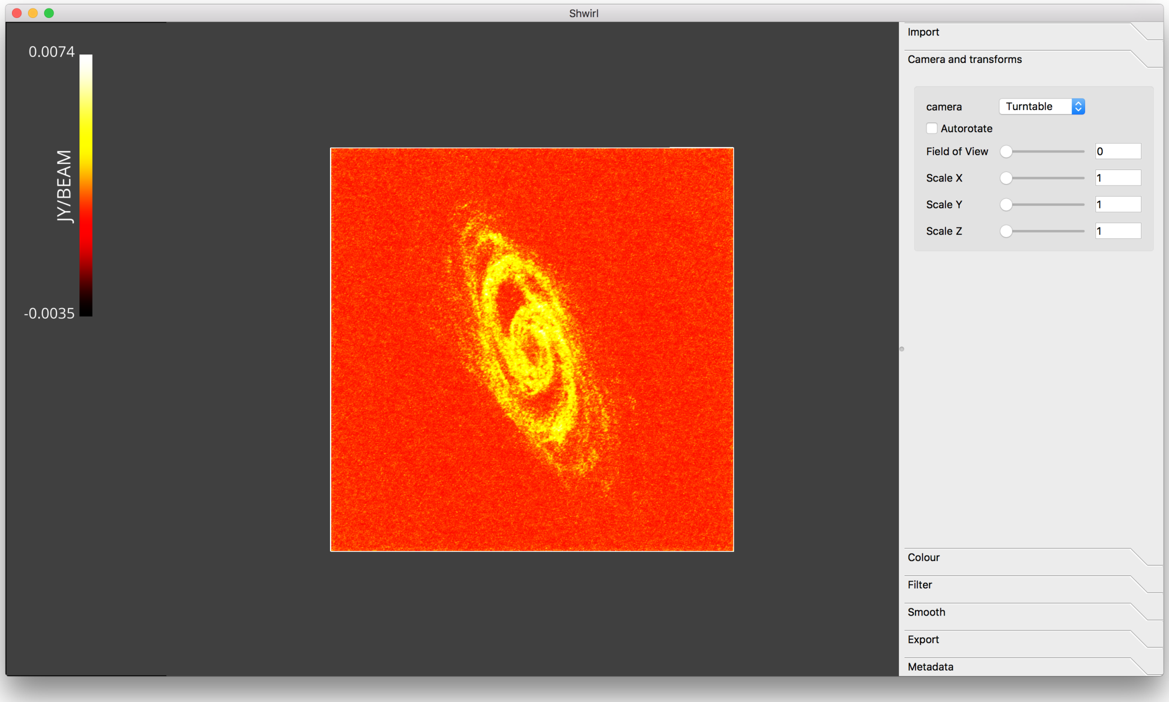
The colour bar information is based on the FITS header and will vary depending on the colouring method. If you find any oddities, please let me know.

3.3 Widgets

The widgets can be used to interact and modify the visualisation outcome. Usage is straight forward: *click, slide, select*. See what happens!

3.4 Cameras and transform

You can set the field of view of the camera:



camera: Turntable

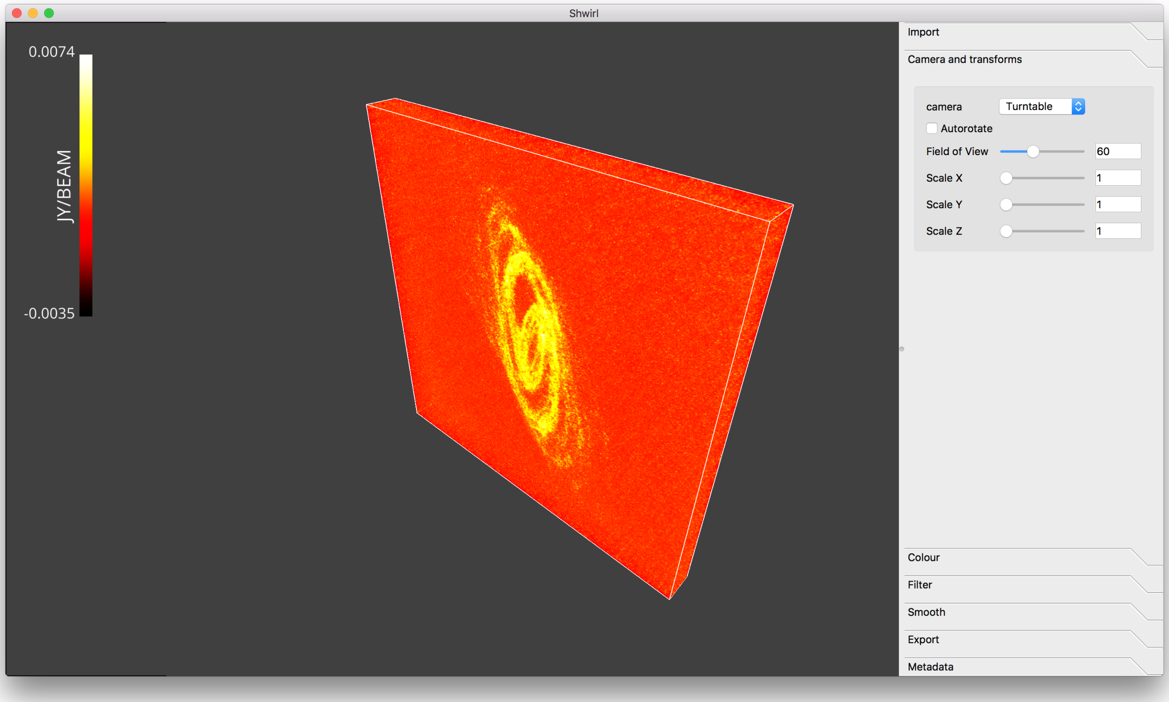
□ Autorotate

Field of View: 0

Scale X: 1

Scale Y: 1

Scale Z: 1



camera Turntable

□ Autorotate

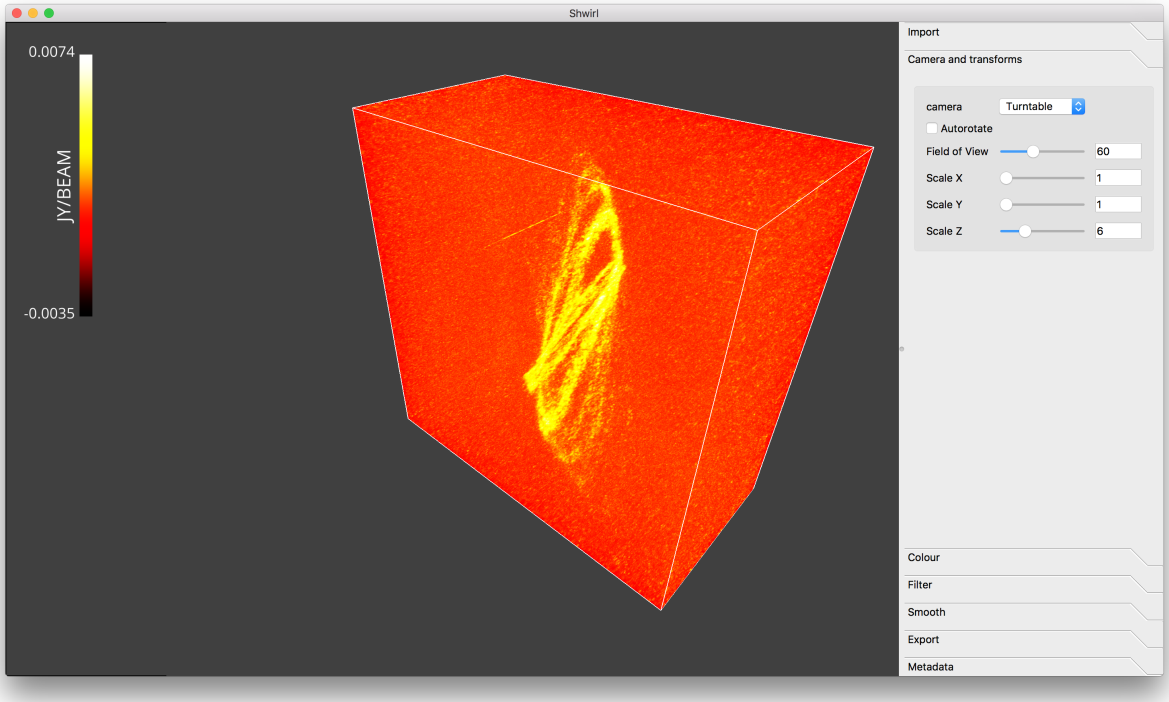
Field of View 60

Scale X 1

Scale Y 1

Scale Z 1

You can scale axes (e.g. if the spectral axis does not have as many elements as the spatial axes):



camera Turntable

□ Autorotate

Field of View 60

Scale X 1

Scale Y 1

Scale Z 6

Other options include:

- Choice of different cameras (VisPy's cameras)

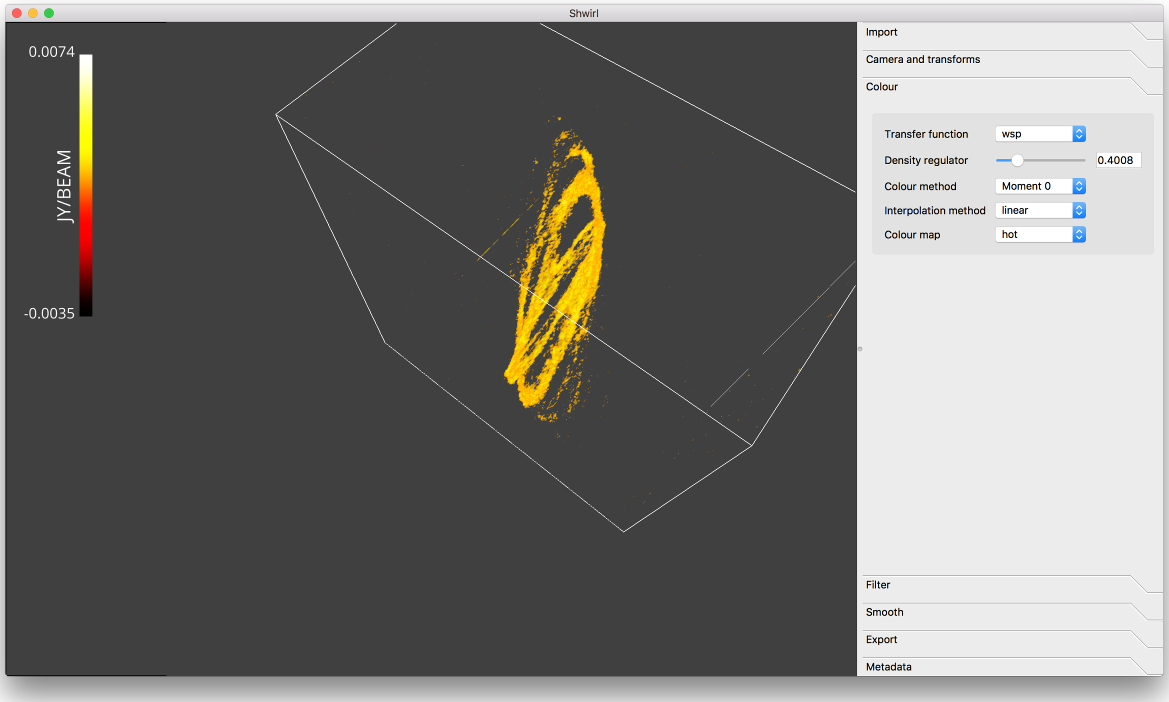
- Turntable (Default)
- Argball
- Fly (using keyboard's arrows)
- Autorotate

3.5 Colour

Shwirl includes different transfer functions and colouring methods.

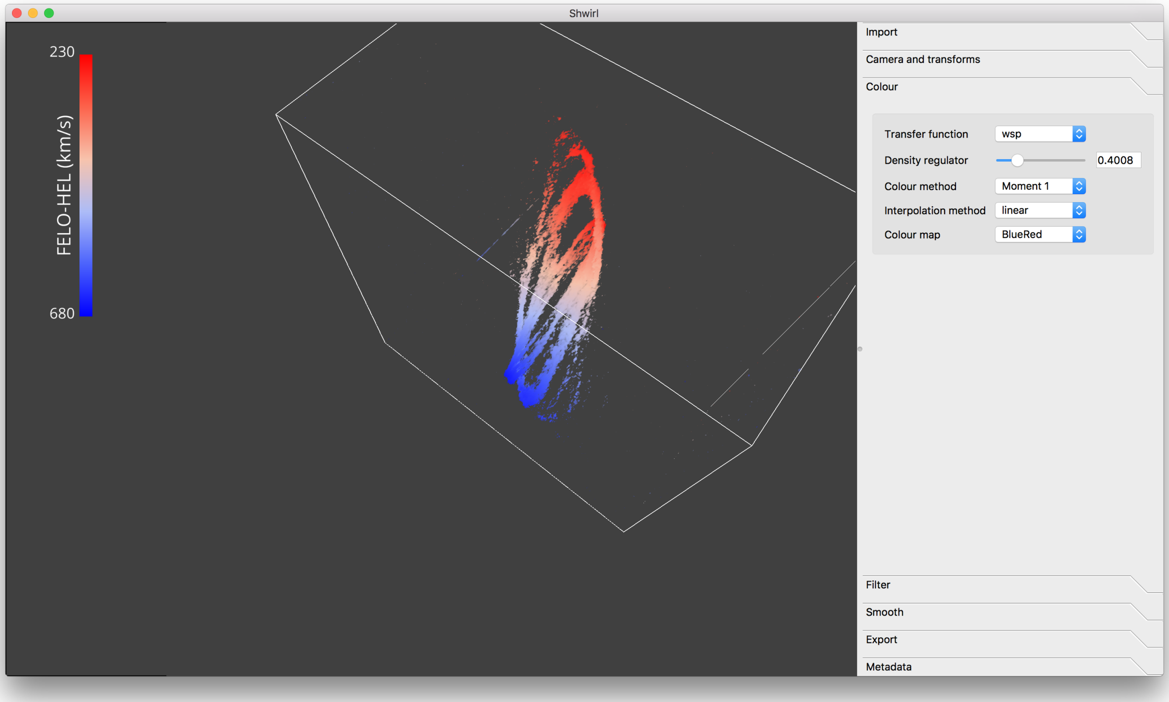
- Transfer functions
  - Maximum Intensity Projection (MIP)
  - Local Maximum Intensity Projection (LMIP)
  - Accumulated Voxel Intensity Projection (AVIP)
  - Iso surface
- Colouring methods
- Moment 0 (maps voxel intensity to colour)
- Moment 1 (maps velocity to colour and voxel intensity to transparency)
- RGB cube (maps voxel xyz position to rgb, and voxel intensity to transparency)

Example of AVIP + Moment 0:



Transfer function: wsp
Density regulator: 0.4008
Colour method: Moment 0
Interpolation method: linear
Colour map: hot

Example of AVIP + Moment 1:



Transfer function wsp

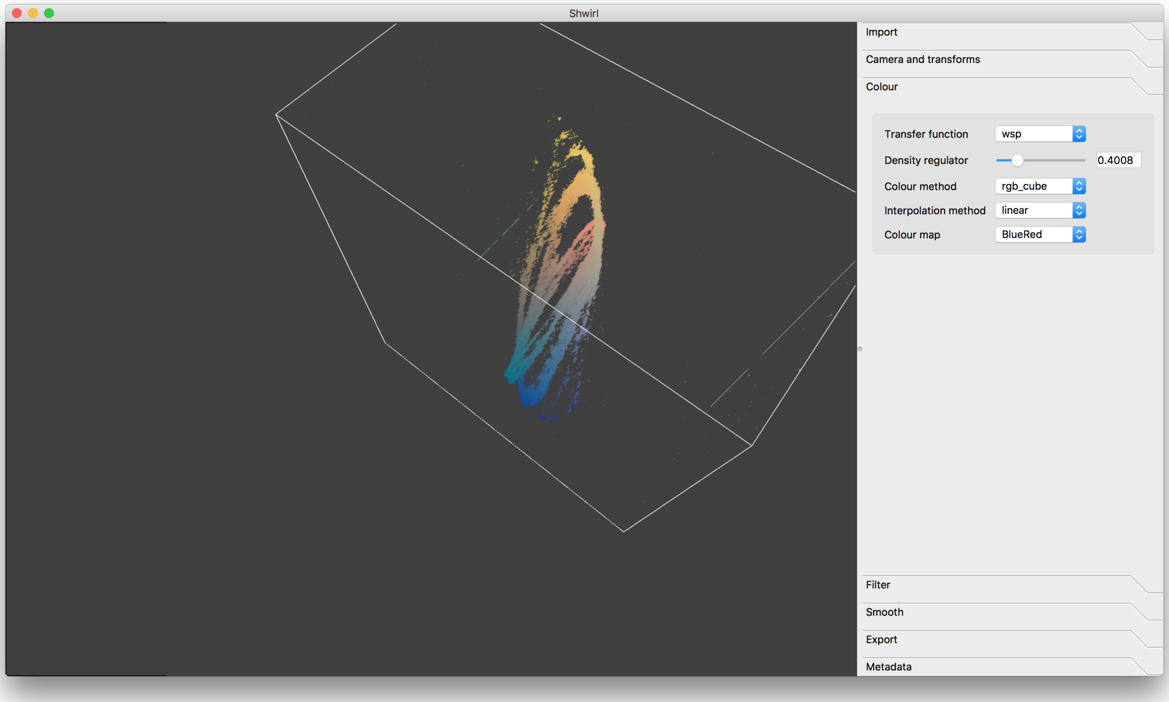
Density regulator 0.4008

Colour method Moment 1

Interpolation method linear

Colour map BlueRed

Example of AVIP + RGB cube:



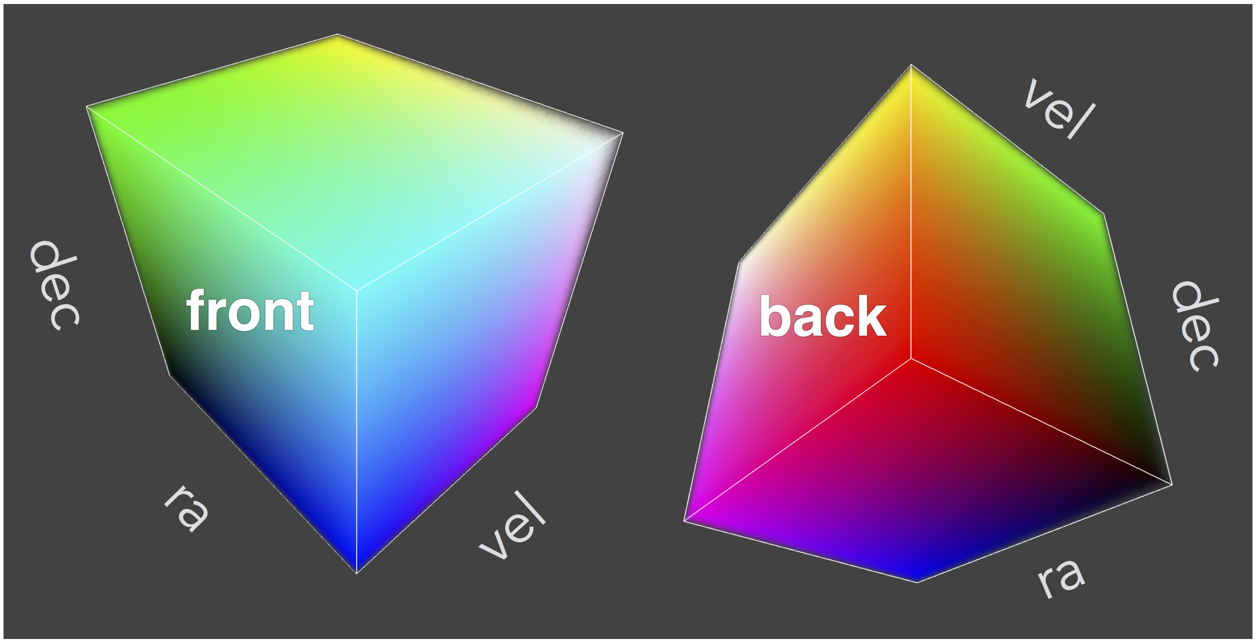
Transfer function wsp

Density regulator 0.4008

Colour method rgb_cube

Interpolation method linear

Colour map BlueRed

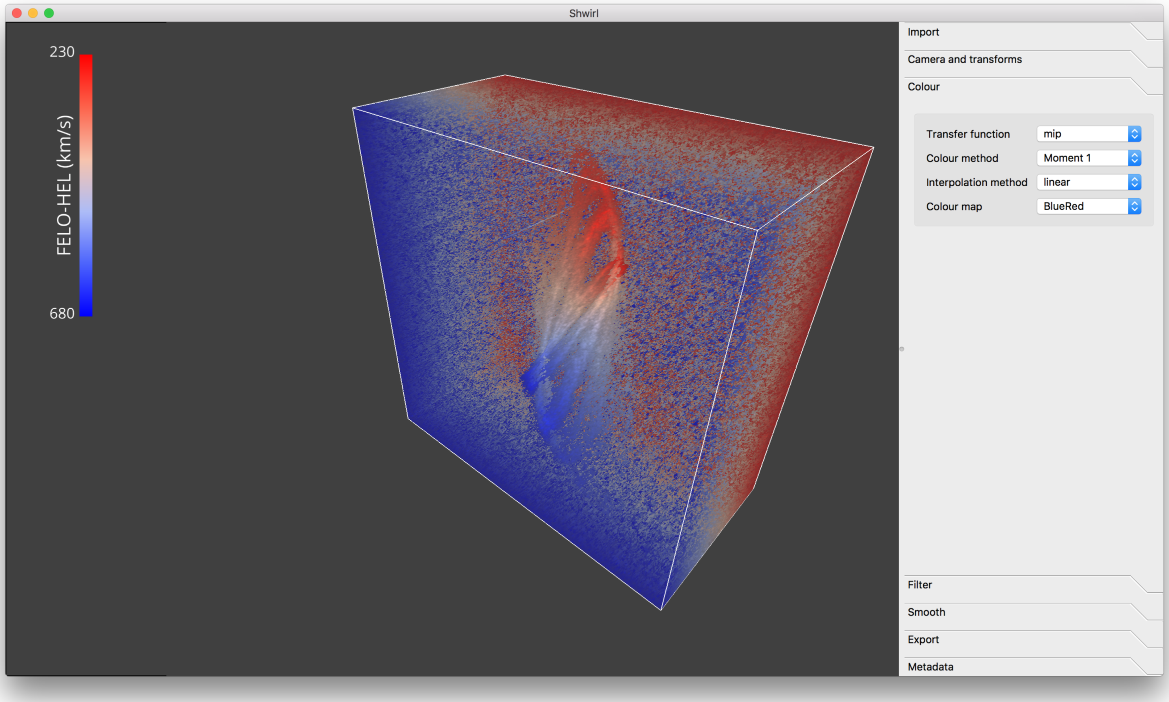


MIP is used in the other examples.

3.6 Filter

- Intensity clipping (Filter out)
- Intensity domain scaling (Rescale)

No Filter:



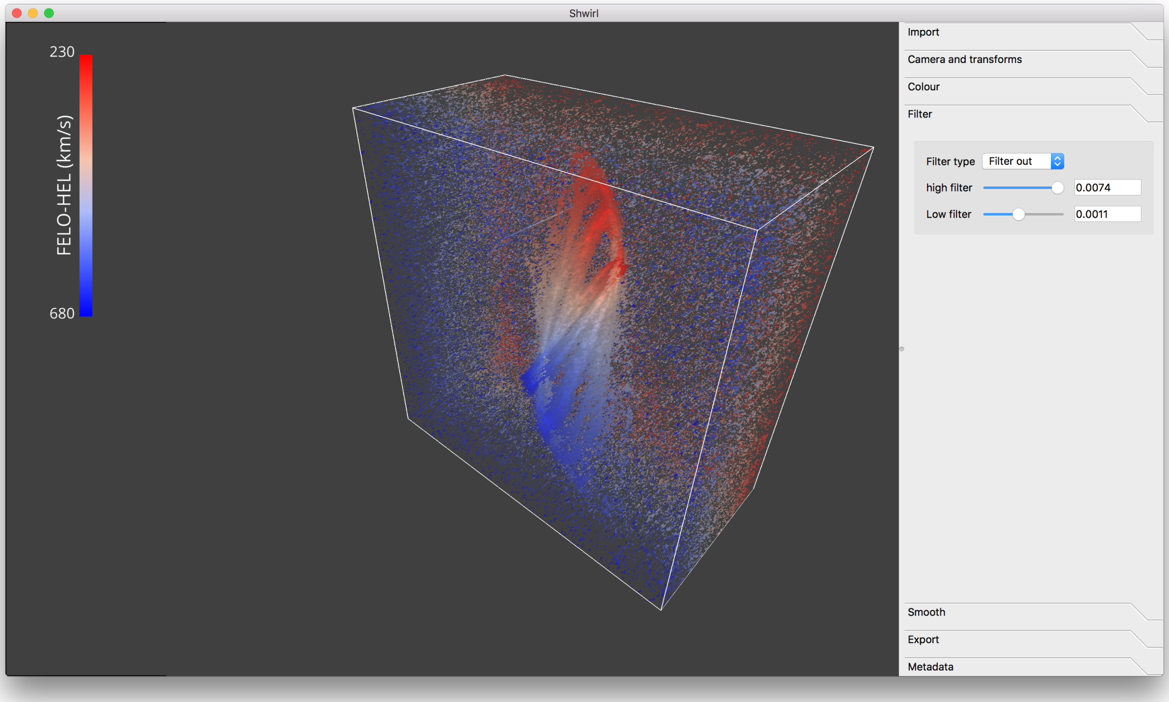
Transfer function: mip

Colour method: Moment 1

Interpolation method: linear

Colour map: BlueRed

Intensity clipping:



Filter type: Filter out

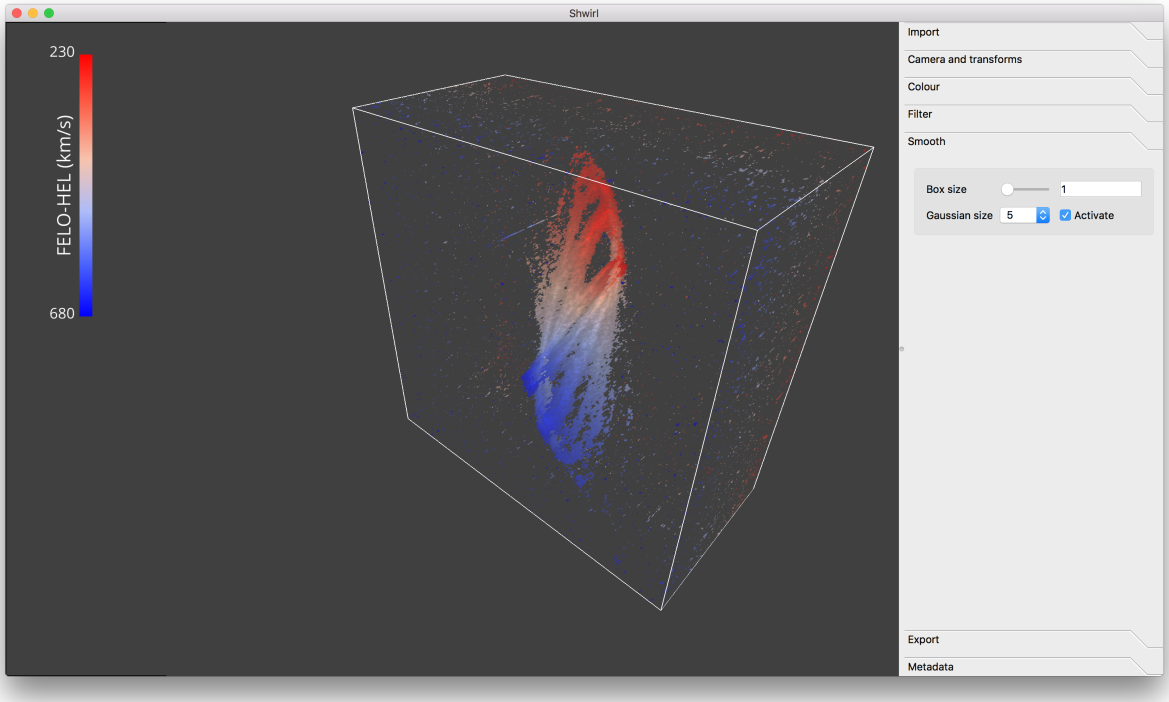
high filter: 0.0074

Low filter: 0.0011

3.7 Smooth

- Box smoothing
- Gaussian smoothing

Intensity clipping + Gaussian smoothing

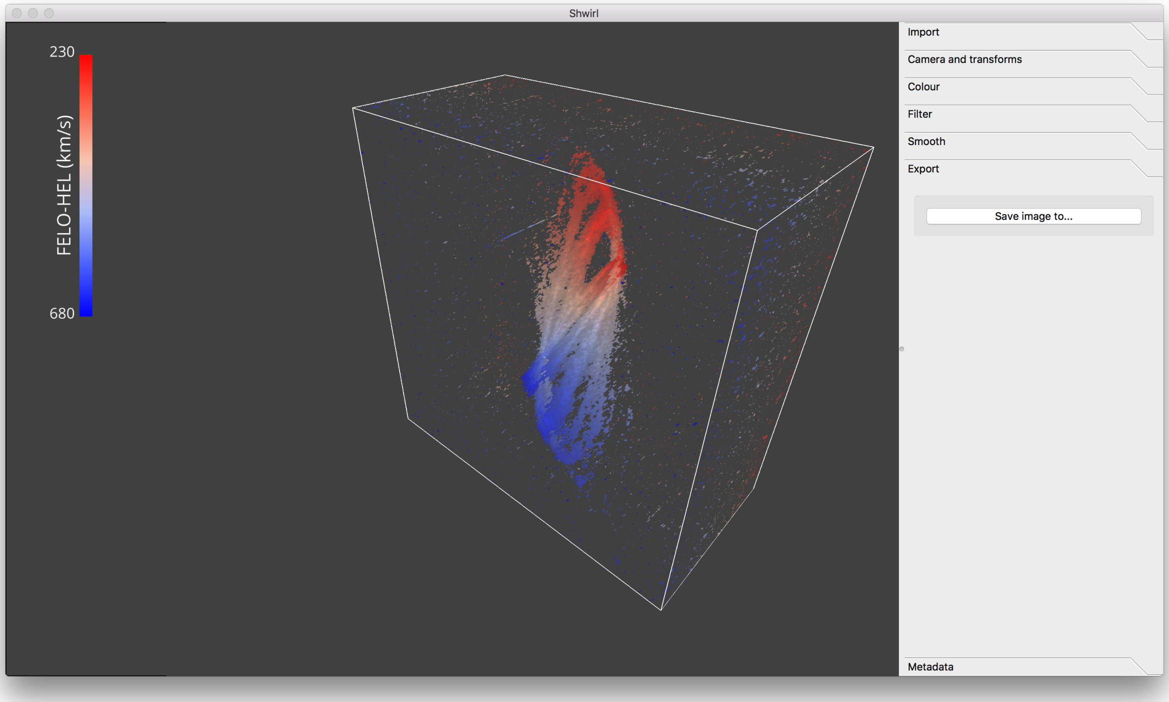


Box size 1

Gaussian size 5 ☑ Activate

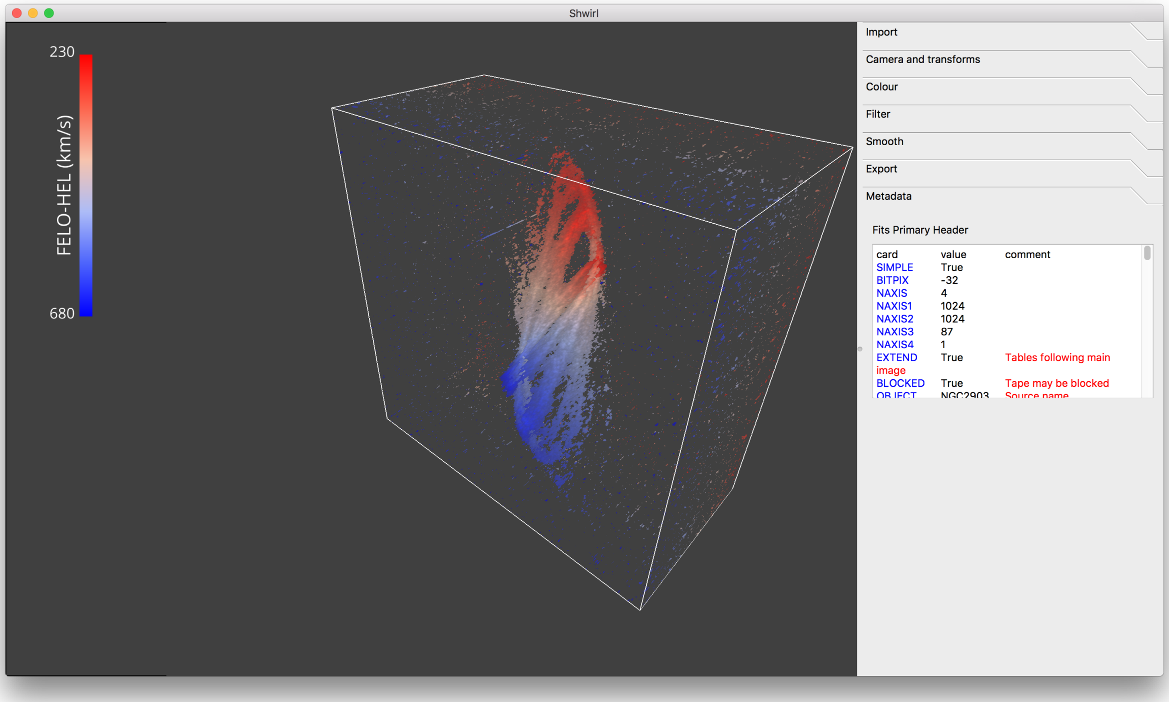
3.8 Export

The visualisation can be exported to an image:



3.9 Metadata

The information of the Primary Header is available in the metadata tab:



Fits Primary Header

card	value	comment
SIMPLE	True	
BITPIX	-32	
NAXIS	4	
NAXIS1	1024	
NAXIS2	1024	
NAXIS3	87	
NAXIS4	1	
EXTEND	True	Tables following main image
BLOCKED	True	Tape may be blocked
OBJECT	NGC2903	Source name

3.10 Computing an emission line ratio in 3D space

The functionalities to compute an emission line ratio can currently be found in the *dev* branch on github. See Vohl, Fluke, Barnes & Hassan (2017) for more details. More to come soon.

3.11 Axis labels

Axis labels are currently under development. Coming soon.

CHAPTER 4

Example data

Spectral cubes used in Vohl, Fluke, Barnes & Hassan (Submitted) are the following:

4.1 Neutral Hydrogen (HI) cube of NGC 2903

The HI spectral cube of NGC 2903 (as shown in usage examples) is taken from The HI Nearby Galaxy Survey (THINGS). This cube, along with all other cubes from THINGS can be downloaded at http://www.mpia.de/THINGS/Data.html.

4.2 Cube of the carbon monoxide (CO) in the Antennae galaxies

The CO spectral cube of the Southern mosaic pattern taken from the ALMA Science Verification data targeting the CO 3-2 line in the Antennae galaxy can be downloaded at https://almascience.nao.ac.jp/alma-data/science-verification/antennae-galaxies