Advanced Astronomical Observations 2021
Session 06: Using Ginga

Kinoshita Daisuke

26 March 2021
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Advanced Astronomical Observations" (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2021 to June 2021.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we use Ginga to view FITS images.

1 About Ginga

Ginga is an astronomical image viewer. Ginga can be used to view FITS images. Ginga is also a toolkit and allow us to build our own FITS viewer software. Ginga is developed and maintained mainly by software engineers at NAOJ (National Astronomical Observatory of Japan) and STScI (Space Telescope Science Institute). For the details of Ginga, visit the official website of Ginga.

- Ginga
  - `https://ginga.readthedocs.io/` (Fig. 1)
  - `https://pypi.org/project/ginga/`
  - `https://ejeschke.github.io/ginga/`

2 Installation of Ginga

First, check whether you have Ginga installed on your computer. Try following. If Ginga is installed on your computer, then you see a window like Fig. 2.

```
% ginga
```

If Ginga is not installed on your computer, you probably see a message something like below.

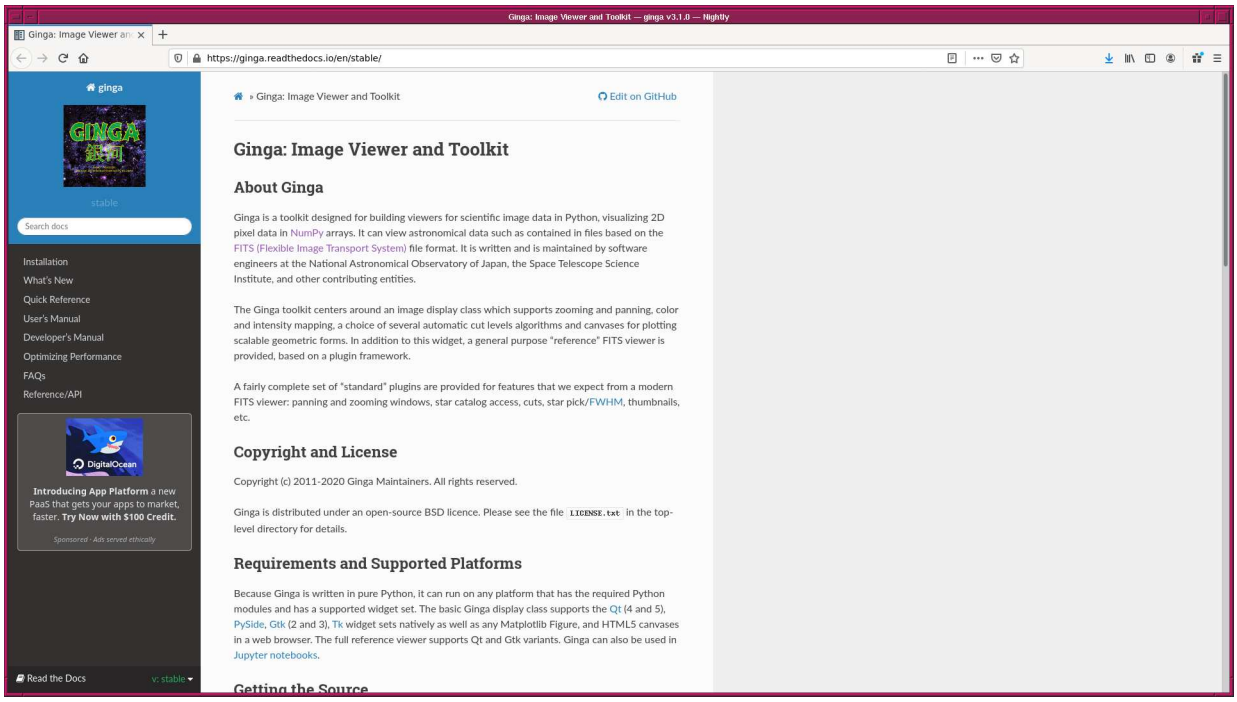

Figure 1: The official website of Ginga.

Figure 2: Ginga's graphical interface right after the start-up.

```
% ginga
ginga: Command not found.
```

If you do not have Ginga installed on your computer, visit following web page, read the installation instruction, and install all the necessary software.

- https://ginga.readthedocs.io/en/stable/install.html

3 Downloading a FITS image

Download a FITS image to play with Ginga.

3.1 Astroquery module

Try following to check whether you have Astroquery module on your computer. If you do not see any error message, then you have Astroquery installed on your computer.

```
% python3.9
Python 3.9.2 (default, Feb 21 2021, 12:39:42)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroquery
>>>
```

If you see an error message like below, you do not have Astroquery on your computer.

```
% python3.9
Python 3.9.1 (default, Dec 19 2020, 11:47:08)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroquery
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'astroquery'
>>>
```

If you do not have Astroquery on your computer, visit the official website of Astroquery and install Astroquery.

- Astroquery
  - https://astroquery.readthedocs.io/ (Fig. 3)
  - https://pypi.org/project/astroquery/
  - https://github.com/astropy/astroquery

3.2 Using name resolver

Try name resolver using Astroquery. Here is an example.

Python Code 1: ao2021_s06_01.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse
```

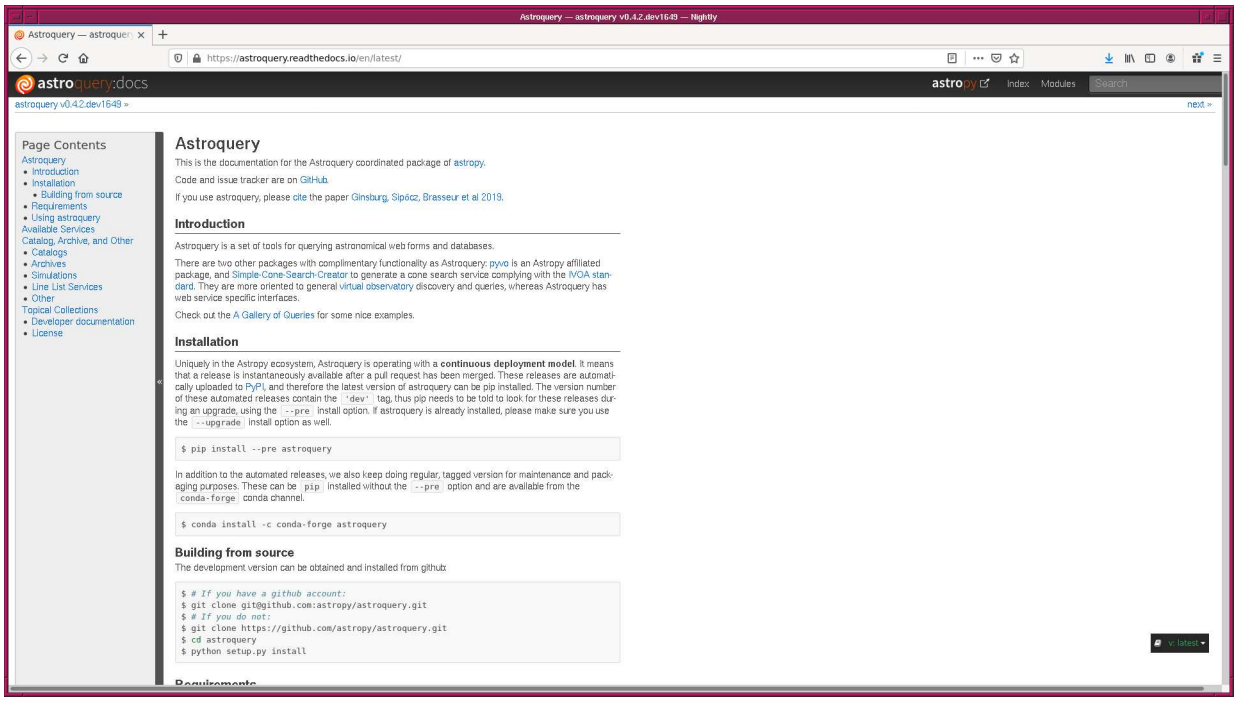

Figure 3: The official website of Astroquery.

```
# importing sys module
import sys

# importing astroquery module
import astroquery.simbad
import astroquery.ned

# constructing parser object
desc = "getting coordinate from given target name"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_resolver = ['simbad', 'ned']
parser.add_argument ('-r', '--resolver', choices=list_resolver, \
                     default='simbad', help='choice of name resolver')
parser.add_argument ('-t', '--target', default='', help='target name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
name_resolver = args.resolver
target_name = args.target

# checking target name
if (target_name == ''):
    # printing error message
    print ("No target name is given!")
    # exit
    sys.exit ()

# using name resolver
if (name_resolver == 'simbad'):
```

```
    query_result = astroquery.simbad.Simbad.query_object (target_name)
elif (name_resolver == 'ned'):
    query_result = astroquery.ned.Ned.query_object (target_name)

# printing result of the query
print (query_result.pprint (show_unit=True))
```

Run the script, and show the information of Acrux.

```
% chmod a+x ao2021_s06_01.py
% ./ao2021_s06_01.py -t Acrux
 MAIN_ID         RA           DEC       ... COO_WAVELENGTH     COO_BIBCODE
             "h:m:s"       "d:m:s"     ...
--------- ------------- ------------- ... -------------- -------------------
* alf Cru 12 26 35.8952 -63 05 56.734 ...              O 2007A&A...474..653V
None
```

Try to use NED instead of Simbad.

```
% ./ao2021_s06_01.py -r ned -t M44
No. Object Name     RA      ... Redshift Points Diameter Points Associations
                  degrees   ...
--- ----------- ---------- ... --------------- --------------- ------------
  1 MESSIER 044  130.09249 ...               0               0            2
None
```

Modify the script to print RA and Dec only.

Python Code 2: ao2021_s06_02.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing astroquery module
import astroquery.simbad
import astroquery.ned

# constructing parser object
desc = "getting coordinate from given target name"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_resolver = ['simbad', 'ned']
parser.add_argument ('-r', '--resolver', choices=list_resolver, \
                     default='simbad', help='choice of name resolver')
parser.add_argument ('-t', '--target', default='', help='target name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
```

```
name_resolver = args.resolver
target_name = args.target

# checking target name
if (target_name == ''):
    # printing error message
    print ("No target name is given!")
    # exit
    sys.exit ()

# using name resolver
if (name_resolver == 'simbad'):
    query_result = astroquery.simbad.Simbad.query_object (target_name)
elif (name_resolver == 'ned'):
    query_result = astroquery.ned.Ned.query_object (target_name)

# RA and Dec
RA  = query_result['RA'][0]
Dec = query_result['DEC'][0]

# printing result of the query
print ("%s" % target_name)
if (name_resolver == 'simbad'):
    print ("  RA  = %s (in hhmmss)" % RA)
    print ("  Dec = %s (in ddmmss)" % Dec)
elif (name_resolver == 'ned'):
    print ("  RA  = %s (in deg)" % RA)
    print ("  Dec = %s (in deg)" % Dec)
```

Run the script.

```
% chmod a+x ao2021_s06_02.py
% ./ao2021_s06_02.py -r simbad -t Regulus
Regulus
  RA  = 10 08 22.3109 (in hhmmss)
  Dec = +11 58 01.951 (in ddmmss)
% ./ao2021_s06_02.py -r ned -t Regulus
Regulus
  RA  = 152.11708 (in deg)
  Dec = 12.30639 (in deg)
```

Use Astropy to convert the coordinate.

Python Code 3: ao2021_s06_03.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing astroquery module
import astroquery.simbad
import astroquery.ned

# importing astropy module
```

```
import astropy.coordinates
import astropy.units

# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# constructing parser object
desc = "getting coordinate from given target name"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_resolver = ['simbad', 'ned']
parser.add_argument ('-r', '--resolver', choices=list_resolver, \
                     default='simbad', help='choice of name resolver')
parser.add_argument ('-t', '--target', default='', help='target name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
name_resolver = args.resolver
target_name = args.target

# checking target name
if (target_name == ''):
    # printing error message
    print ("No target name is given!")
    # exit
    sys.exit ()

# using name resolver
if (name_resolver == 'simbad'):
    query_result = astroquery.simbad.Simbad.query_object (target_name)
elif (name_resolver == 'ned'):
    query_result = astroquery.ned.Ned.query_object (target_name)

# RA and Dec
RA  = query_result['RA']
Dec = query_result['DEC']

# coordinate
if (name_resolver == 'simbad'):
    coord = astropy.coordinates.SkyCoord (RA[0], Dec[0], unit=(u_ha, u_deg))
elif (name_resolver == 'ned'):
    coord = astropy.coordinates.SkyCoord (RA[0], Dec[0], unit=(u_deg, u_deg))

coord_str = coord.to_string (style='hmsdms')
(coord_ra_str, coord_dec_str) = coord_str.split ()
coord_ra_deg  = coord.ra.deg
coord_dec_deg = coord.dec.deg

# printing coordinate
print ("Target Name: %s" % target_name)
print ("  RA:  %s = %f deg" % (coord_ra_str, coord_ra_deg) )
print ("  Dec: %s = %f deg" % (coord_dec_str, coord_dec_deg) )
```

Execute the script.

```
% chmod a+x ao2021_s06_03.py
% ./ao2021_s06_03.py -r ned -t M35
Target Name: M35
  RA:  06h09m05.0592s = 92.271080 deg
  Dec: +24d20m19.104s = 24.338640 deg
```

3.3 Downloading image

Here is a script to download an astronomical image from the data archives.

Python Code 4: ao2021_s06_04.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing astroquery module
import astroquery.simbad
import astroquery.ned
import astroquery.skyview

# importing astropy module
import astropy.coordinates
import astropy.units

# importing datetime module
import datetime

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# date/time
now = datetime.datetime.now ().isoformat ()

# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# constructing parser object
desc = "getting coordinate from given target name"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_resolver = ['simbad', 'ned']
list_survey   = ['DSS1 Blue', 'DSS1 Red', 'DSS2 Blue', 'DSS2 Red', 'DSS2 IR', \
                 'SDSSu', 'SDSSg', 'SDSSr', 'SDSSi', 'SDSSz']
parser.add_argument ('-r', '--resolver', choices=list_resolver, \
                     default='simbad', help='choice of name resolver')
parser.add_argument ('-s', '--survey', choices=list_survey, \
                     default='DSS2 Red', help='choice of survey')
parser.add_argument ('-t', '--target', default='', help='target name')
```

```
parser.add_argument ('-f', '--fov', type=int, default=1024, \
                     help='field-of-view in pixel')
parser.add_argument ('-o', '--output', default='', help='output file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
name_resolver = args.resolver
survey        = args.survey
target_name   = args.target
fov_pix       = args.fov
file_output   = args.output

# checking target name
if (target_name == ''):
    # printing error message
    print ("No target name is given!")
    # exit
    sys.exit ()

# checking output file name
if (file_output == ''):
    # printing error message
    print ("No output file name is given!")
    # exit
    sys.exit ()
elif not (file_output[-5:] == '.fits'):
    # printing error message
    print ("Output file must be FITS file!")
    # exit
    sys.exit ()

# using name resolver
if (name_resolver == 'simbad'):
    query_result = astroquery.simbad.Simbad.query_object (target_name)
elif (name_resolver == 'ned'):
    query_result = astroquery.ned.Ned.query_object (target_name)

# RA and Dec
RA  = query_result['RA']
Dec = query_result['DEC']

# coordinate
if (name_resolver == 'simbad'):
    coord = astropy.coordinates.SkyCoord (RA[0], Dec[0], unit=(u_ha, u_deg))
elif (name_resolver == 'ned'):
    coord = astropy.coordinates.SkyCoord (RA[0], Dec[0], unit=(u_deg, u_deg))

coord_str = coord.to_string (style='hmsdms')
(coord_ra_str, coord_dec_str) = coord_str.split ()
coord_ra_deg  = coord.ra.deg
coord_dec_deg = coord.dec.deg

# printing coordinate
print ("Target Name: %s" % target_name)
print ("  RA:  %s = %f deg" % (coord_ra_str, coord_ra_deg) )
print ("  Dec: %s = %f deg" % (coord_dec_str, coord_dec_deg) )
```

```
# searching image
list_image = astroquery.skyview.SkyView.get_image_list (position=coord, \
                                                         survey=survey)

# printing image list
print ("Available images:")
print (" ", list_image)

# getting image
image = astroquery.skyview.SkyView.get_images (position=coord, survey=survey, \
                                               pixels=fov_pix)

# header and data
image0 = image[0]
header = image0[0].header
data   = image0[0].data

# adding comments in header
header['history'] = "image downloaded from %s" % survey
header['history'] = "image saved on %s" % now

# saving to a FITS file
astropy.io.fits.writeto (file_output, data, header=header)
```

Try above script. Download an image of M13 from DSS (Digitized Sky Survey[1]).

```
% ./ao2021_s06_04.py -h
usage: ao2021_s06_04.py [-h] [-r {simbad,ned}]
                        [-s {DSS1 Blue,DSS1 Red,DSS2 Blue,DSS2 Red,DSS2 IR,SDSSu
,SDSSg,SDSSr,SDSSi,SDSSz}]
                        [-t TARGET] [-f FOV] [-o OUTPUT]

getting coordinate from given target name

optional arguments:
  -h, --help            show this help message and exit
  -r {simbad,ned}, --resolver {simbad,ned}
                        choice of name resolver
  -s {DSS1 Blue,DSS1 Red,DSS2 Blue,DSS2 Red,DSS2 IR,SDSSu,SDSSg,SDSSr,SDSSi,SDSS
z}, --survey {DSS1 Blue,DSS1 Red,DSS2 Blue,DSS2 Red,DSS2 IR,SDSSu,SDSSg,SDSSr,SD
SSi,SDSSz}
                        choice of survey
  -t TARGET, --target TARGET
                        target name
  -f FOV, --fov FOV     field-of-view in pixel
  -o OUTPUT, --output OUTPUT
                        output file name
% ./ao2021_s06_04.py -t M13 -o m13_dss.fits -f 1536
Target Name: M13
  RA:  16h41m41.634s = 250.423475 deg
  Dec: +36d27m40.75s = 36.461319 deg
Available images:
  ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv14766003969435.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv14766562087116.fits
|=========================================| 9.4M/9.4M (100.00%)        23s

% ls -l *.fits
```

[1] https://archive.stsci.edu/cgi-bin/dss_form

```
-rw-r--r--  1 daisuke  taiwan  9449280 Mar 25 20:45 m13_dss.fits
```

4 Viewing a FITS image using Ginga

Use Ginga to view a FITS image. Type following command. You see a window like Fig. 4.

```
% ginga --opencl m13_dss.fits
```

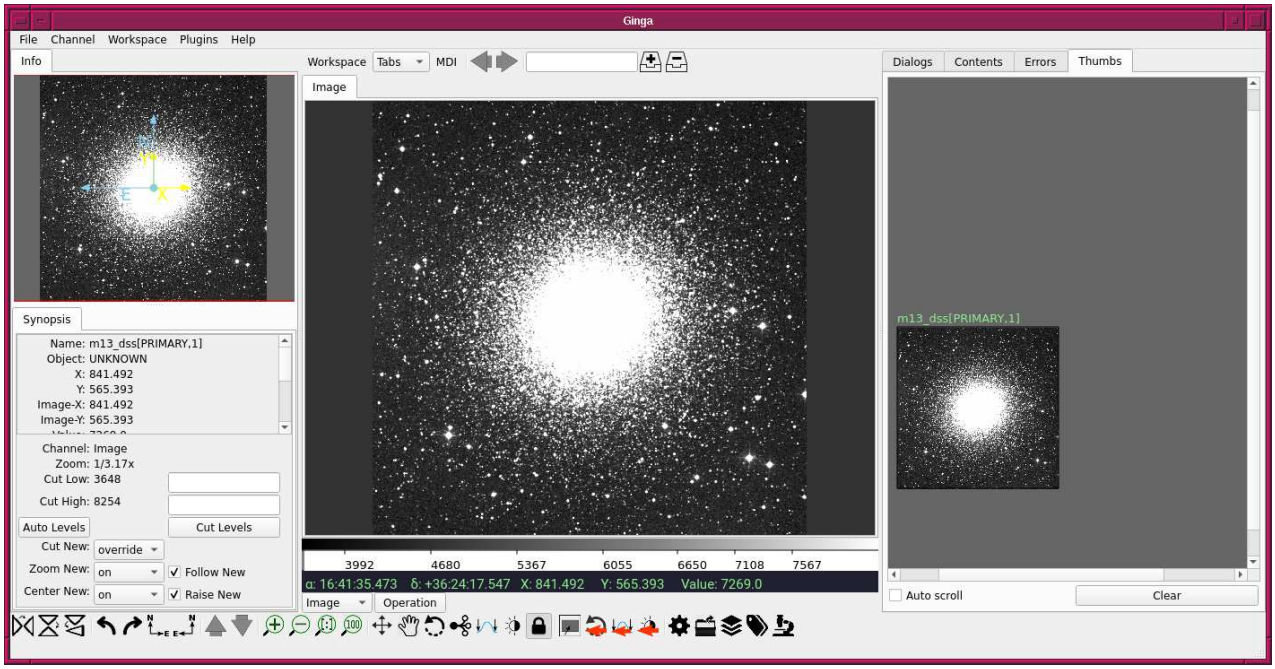

Figure 4: Image of M13 shown by Ginga.

4.1 Changing colour map

Try following to show the "ColorMapPicker" menu.

1. Click a menu "Plugins" at the top of the window,
2. Move the mouse to "RGB",
3. Select "Set Color Map [G]".

The "ColorMapPicker" menu appears on the right-hand side of the window. (Fig. 5) Now, you can choose your favourite colour map. A new choice of colour map will be effective right after you click one of colour map at "ColorMapPicker". Fig. 6 shows an image of M13 after choosing "bone" as a colour map. Try some colour maps, and decide which colour map to use.

4.2 Changing colour distribution algorithm

Try following to show "Preferences" menu.

1. Click a menu "Plugins" at the top of the window,
2. Move the mouse to "Utils",
3. Select "Preferences".

Now, you see "IMAGE: Preferences" menu on the right-hand side of the window. (Fig. 7) Change the colour distribution algorithm and see what happens. Fig. 8 shows an example of choosing "asinh".

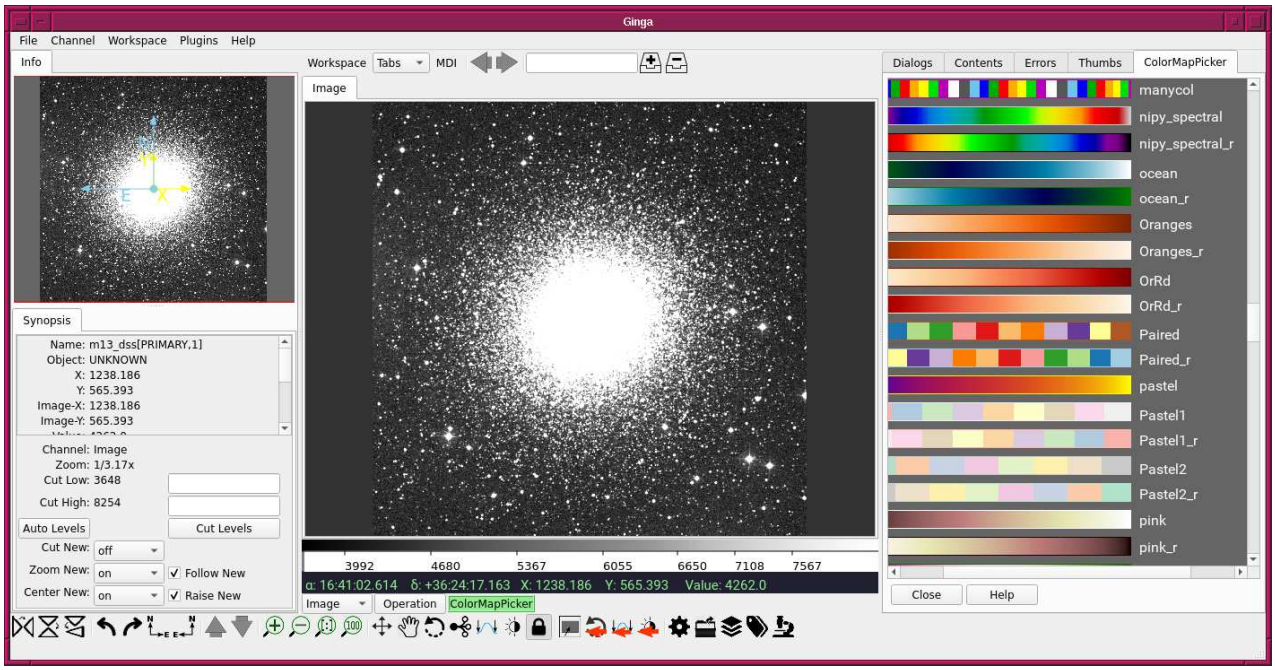

Figure 5: The "ColorMapPicker" menu of Ginga.

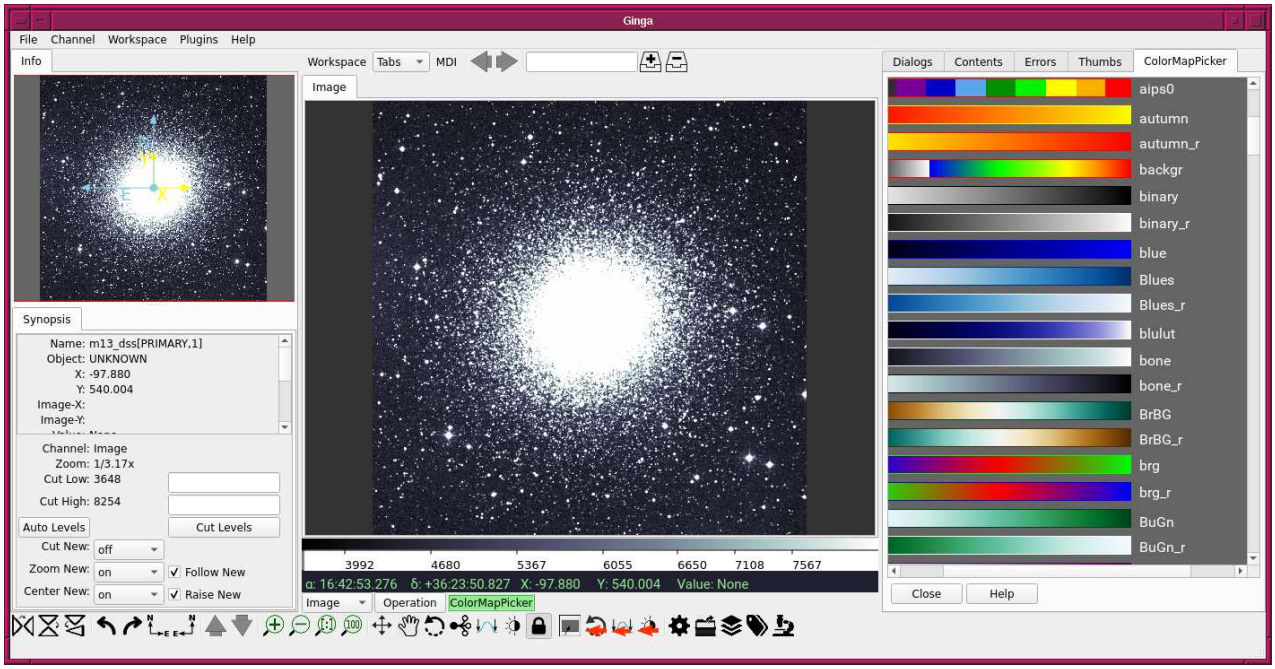

Figure 6: Image of M13 using the colour map "bone".

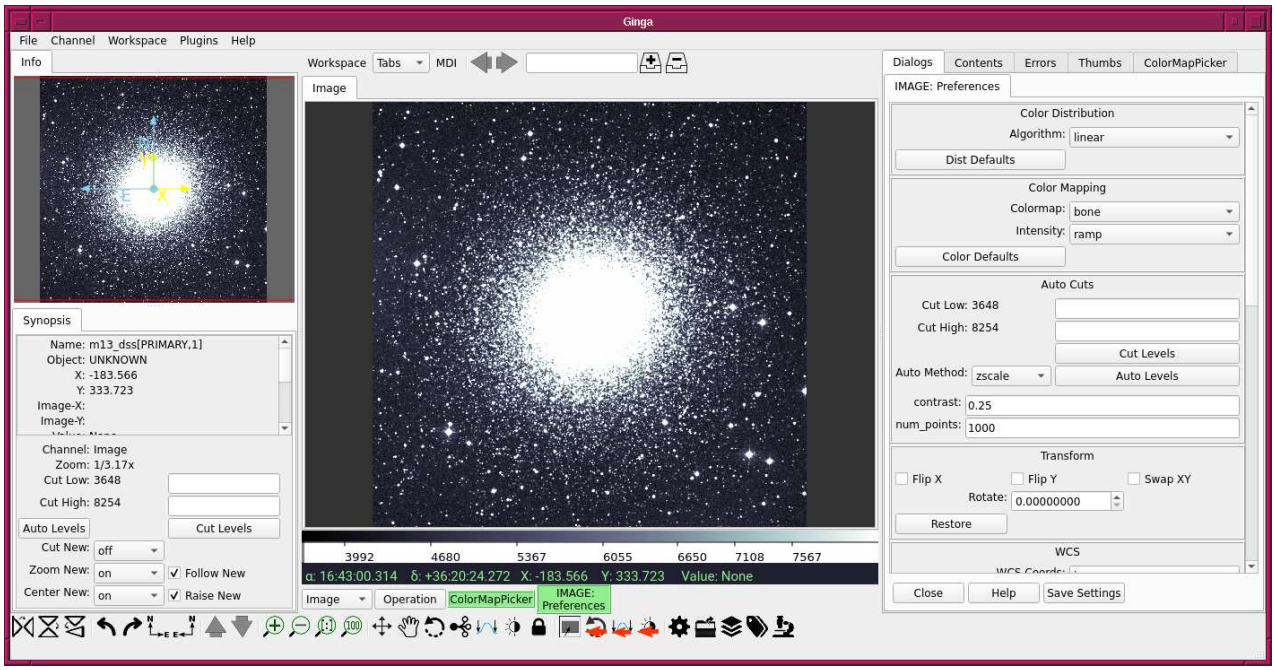

Figure 7: The "IMAGE: Preferences" menu of Ginga.

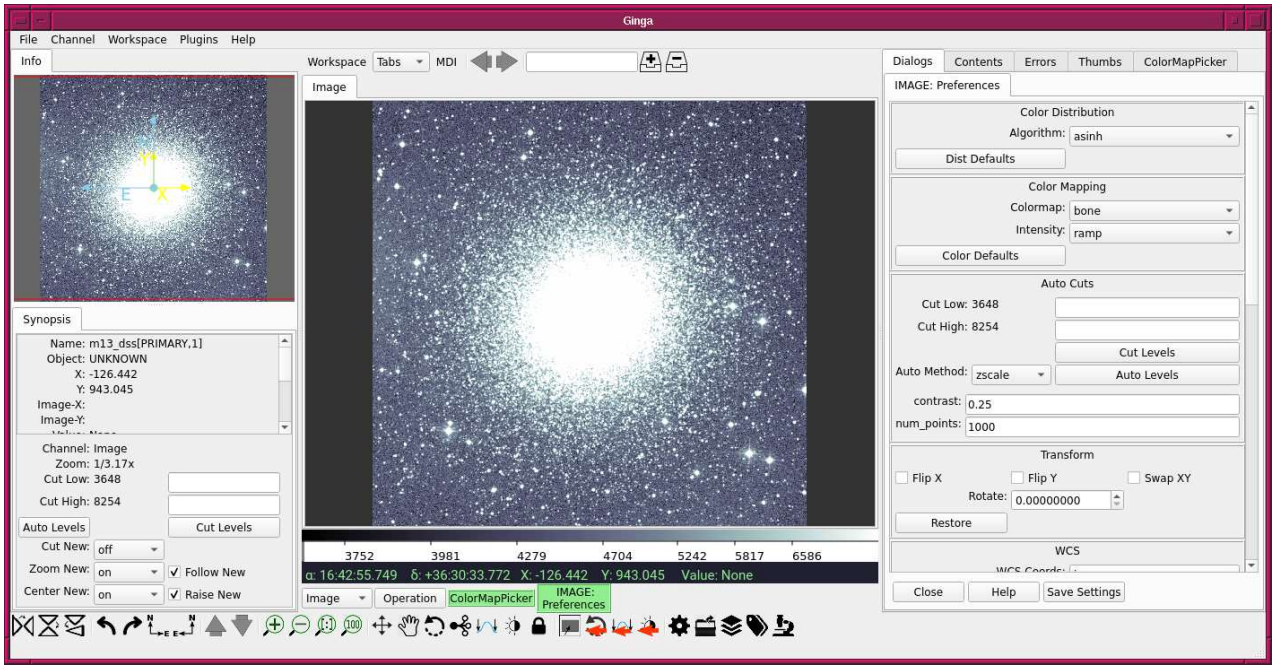

Figure 8: Image of M13 using the colour distribution algorithm "asinh".

4.3 Adjusting cut levels

Try following to change cut levels.

1. Find “Auto Cuts” menu in “IMAGE: Preferences”,
2. Type a number in the box for “Cut Low”,
3. Type a number in the box for “Cut High”,
4. Click the button “Cut Levels”.

Then, you see a new image at the middle of the window. (Fig. 9) You can also try “Auto Method”. Choose “Histogram” from “Auto Method” menu, for example. (Fig. 10)

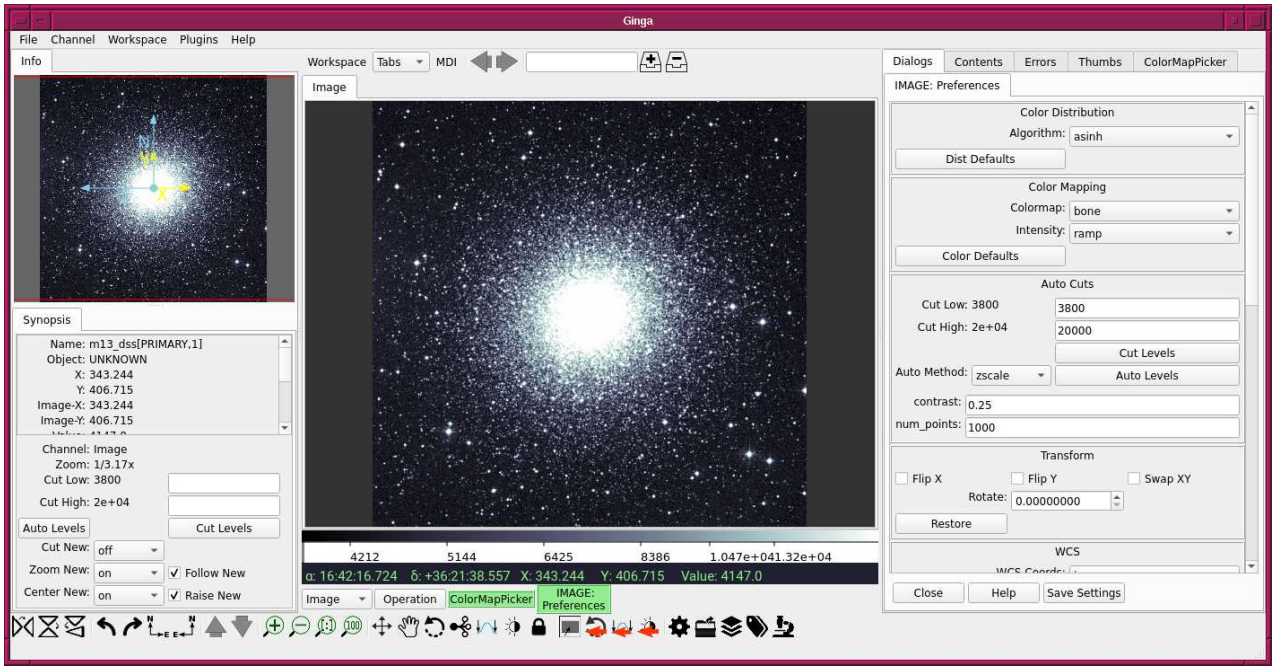

Figure 9: The image of M13 with manually set cut levels.

4.4 Zooming image

Try zooming by scrolling wheel of the mouse. If you prefer to use keyboard short-cut, then try to type “+” and “-” keys. Fig. 11 is the image after zoom-in.

4.5 Panning position

Try following to do panning.

1. Move the mouse to an arbitrary position on the image,
2. Click the middle button of the mouse.

Now you see the clicked position at the centre of the image panel. (Fig. 12)

5 Quick examination of image

5.1 Downloading image

Download one more image.

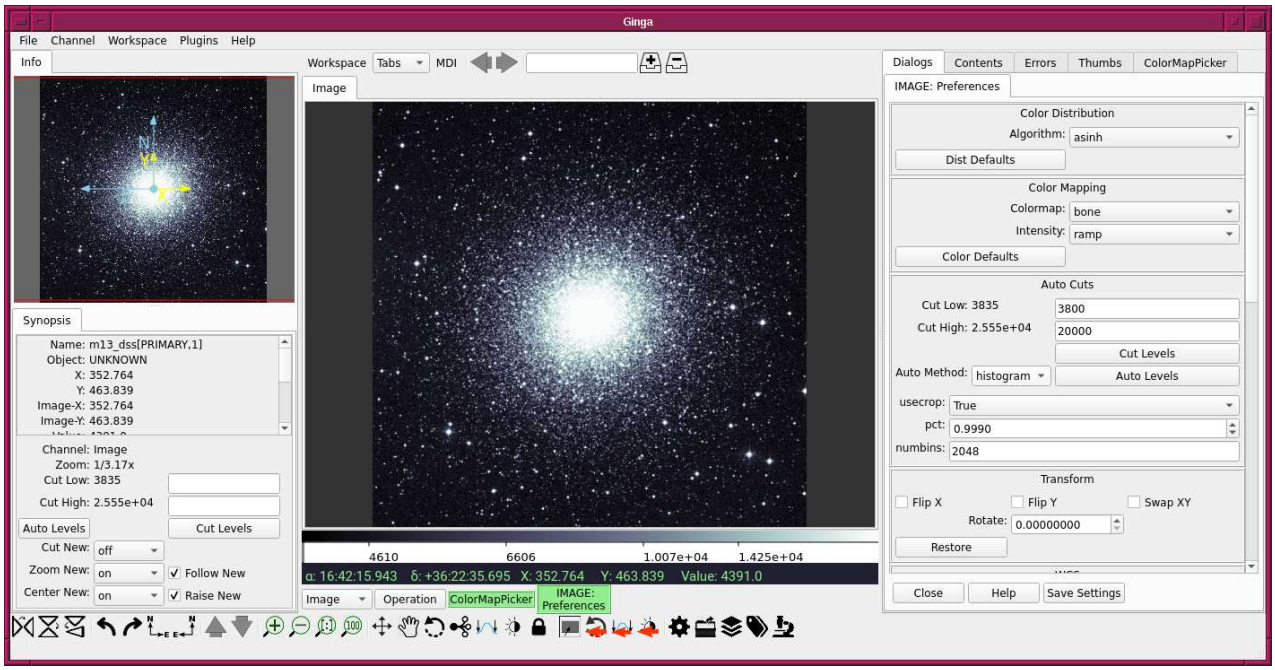

Figure 10: The image of M13 using auto method "histogram".

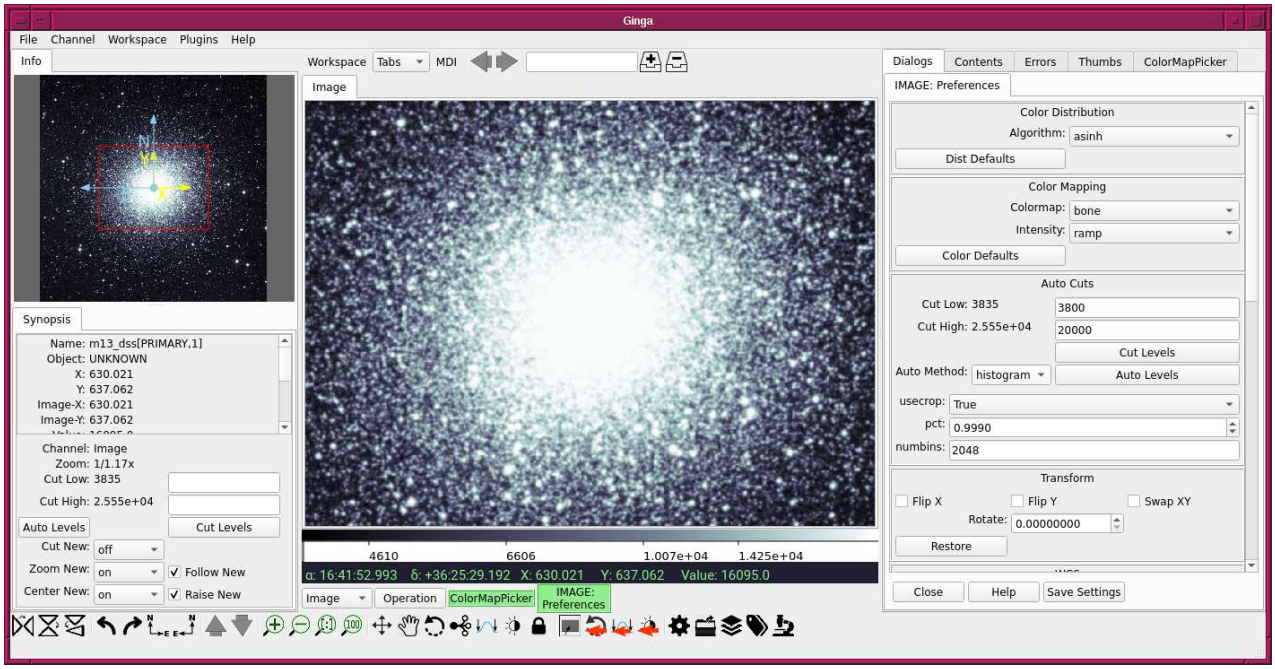

Figure 11: The image of M13 after zoom-in.

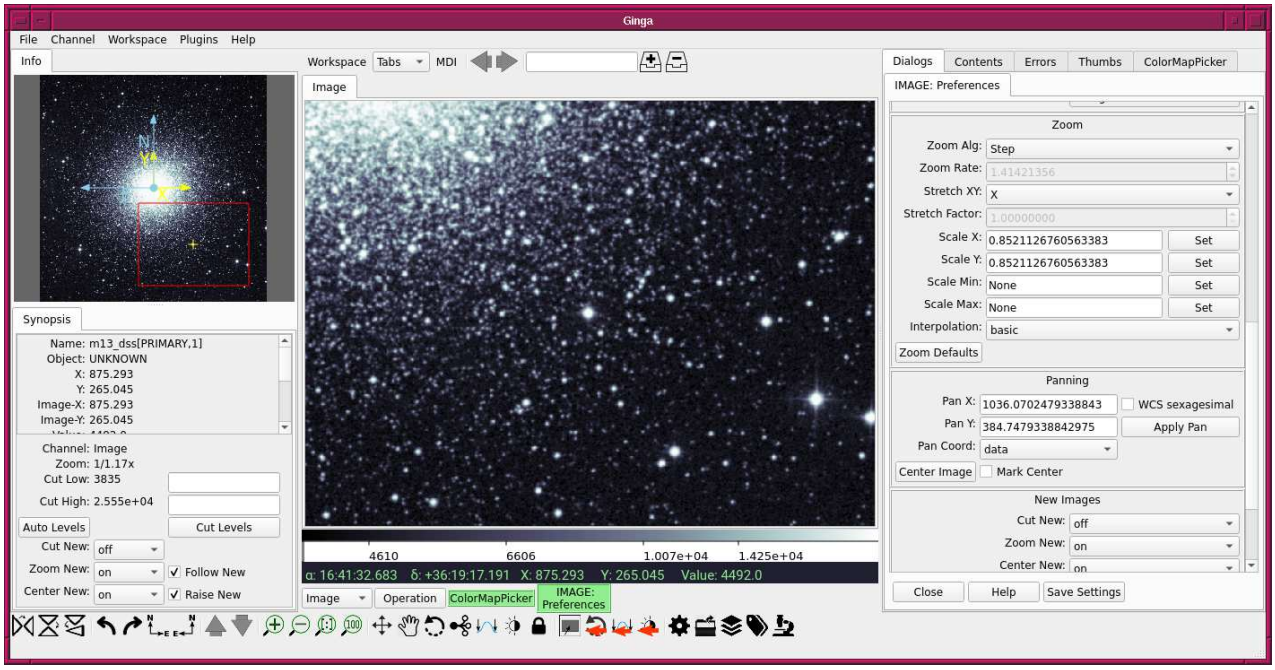

Figure 12: The image of M13 after panning.

```
% ./ao2021_s06_04.py -s SDSSr -t M51 -o m51_sdss.fits -f 1536
Target Name: M51
  RA:  13h29m52.698s = 202.469575 deg
  Dec: +47d11m42.93s = 47.195258 deg
Available images:
  ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv14772567031718.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv14772573045035.fits
|=========================================| 9.4M/9.4M (100.00%)         16s

% ls -l *.fits
-rw-r--r--  1 daisuke  taiwan  9449280 Mar 25 20:45 m13_dss.fits
-rw-r--r--  1 daisuke  taiwan  9449280 Mar 25 22:12 m51_sdss.fits
```

5.2 Viewing image

Load the image.

1. Click the menu “File”,
2. Choose the menu “Load Image”, (Fig. 13)
3. Select the file “`m51_sdss.fits`.

Choose “linear” for colour distribution algorithm, and “zscale” for “Auto Method” at “IMAGE: Preferences” menu. Then, click the “Auto Levels” button. Then, you see a window like Fig. 14.

5.3 Using “Pick”

Use “Pick” to carry out quick examination of image.

1. Use zooming and panning to show a star at the centre of the image panel, (Fig. 15)
2. Click the menu “Plugins”,
3. Move the mouse to “Analysis”,

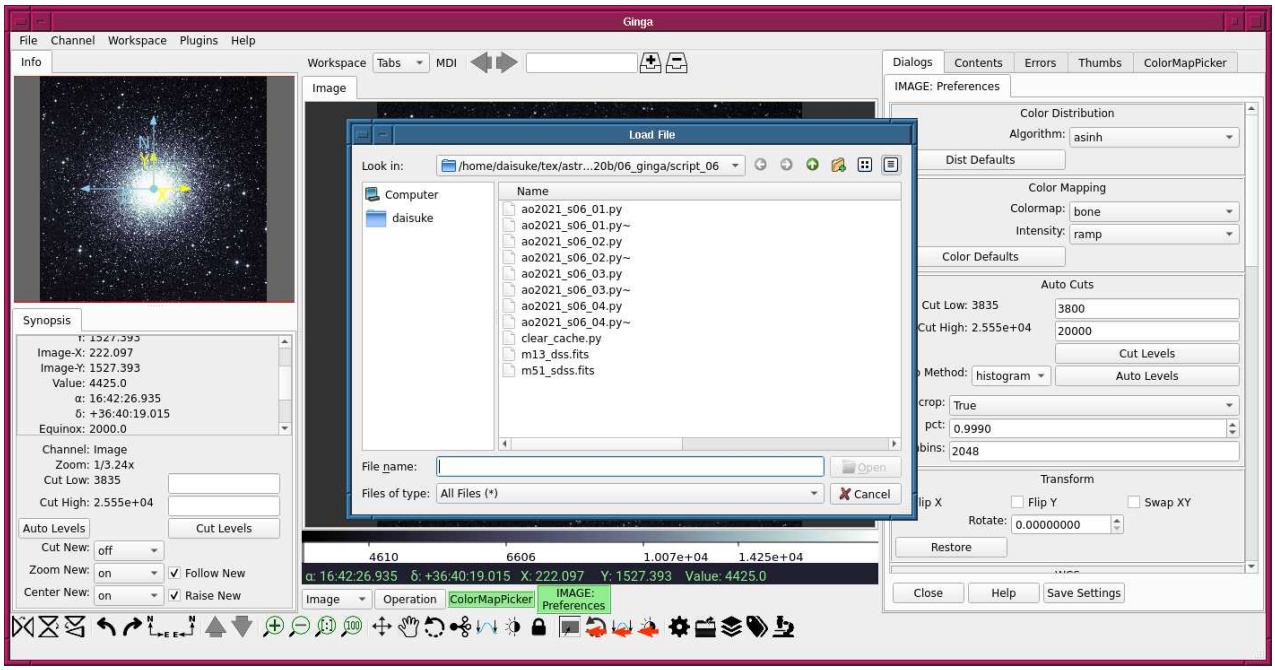

Figure 13: The file selection menu of Ginga.

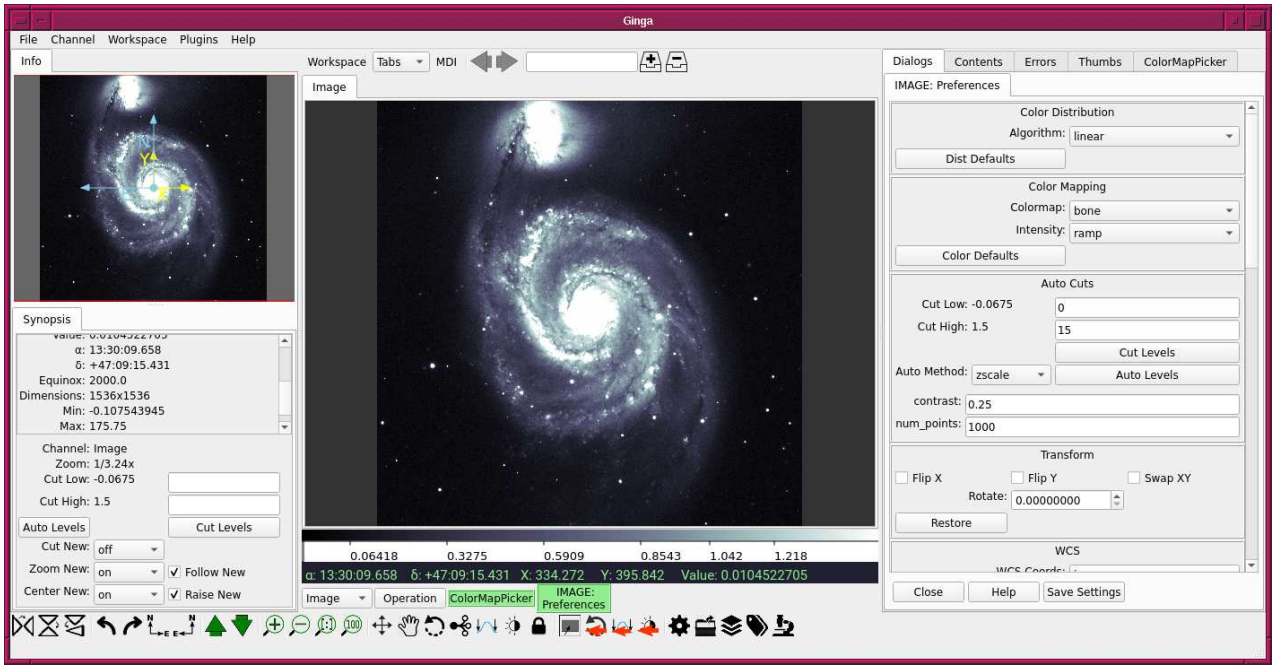

Figure 14: The image of M51 shown by Ginga.

4. Click the menu “Pick”,
5. Move the mouse cursor to the star of your interest,
6. Click the right button of the mouse.

Then, Ginga pick the star. The picked star is marked in the image panel, and the information of stellar image is shown at the right-hand side of the window. (Fig. 16) You can find FWHM (Full Width at Half Maximum) of stellar PSF (Point Spread Function).

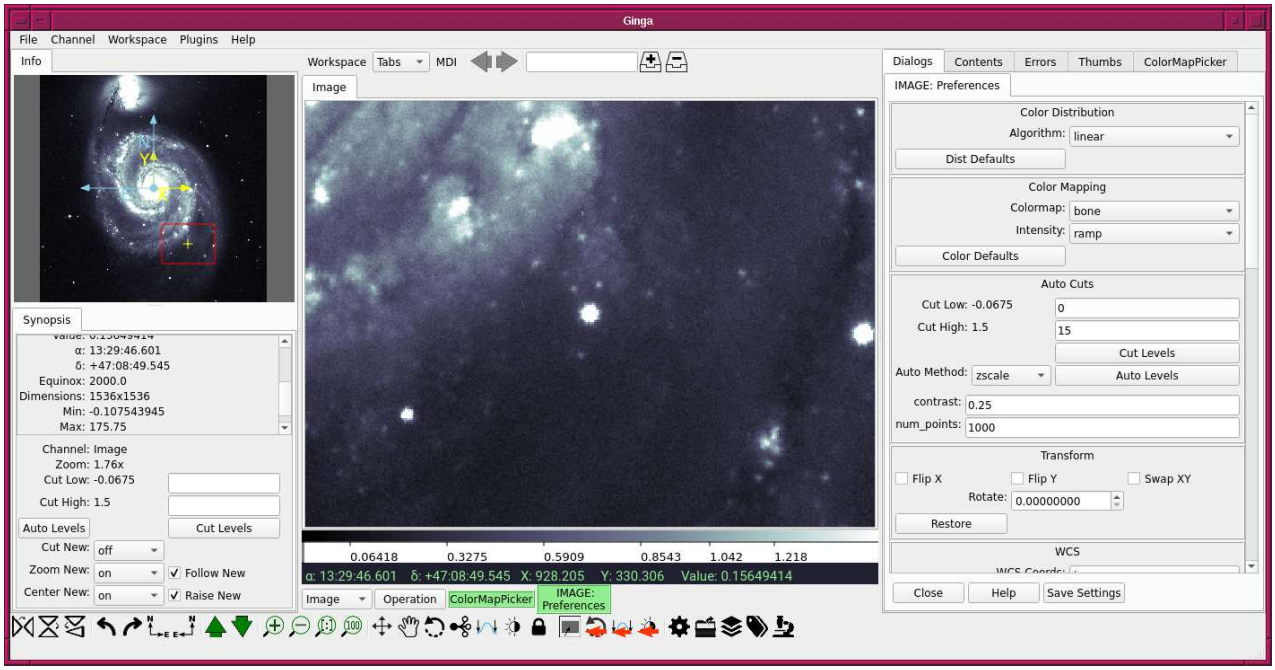

Figure 15: A star is centred at the image panel.

Click the “Radial” panel to show the radial profile of the stellar image. (Fig. 17)

5.4 Making histogram

Use “Histogram” plug-in to make a histogram.

1. Click the menu “Plugins”,
2. Move the mouse to “Analysis”,
3. Click the menu “Histogram”,
4. Use click and drag of the mouse to specify the region of your interest.

Then, you see a histogram on the right-hand side of the window. (Fig. 18)

5.5 Using “WCSAxes”

Use “WCSAxes” to show grids of coordinates.

1. Click the menu “Plugins”,
2. Move the mouse to “Utils”,
3. Click the menu “WCSAxes”.

Then, you see WCS axes superimposed on the image.

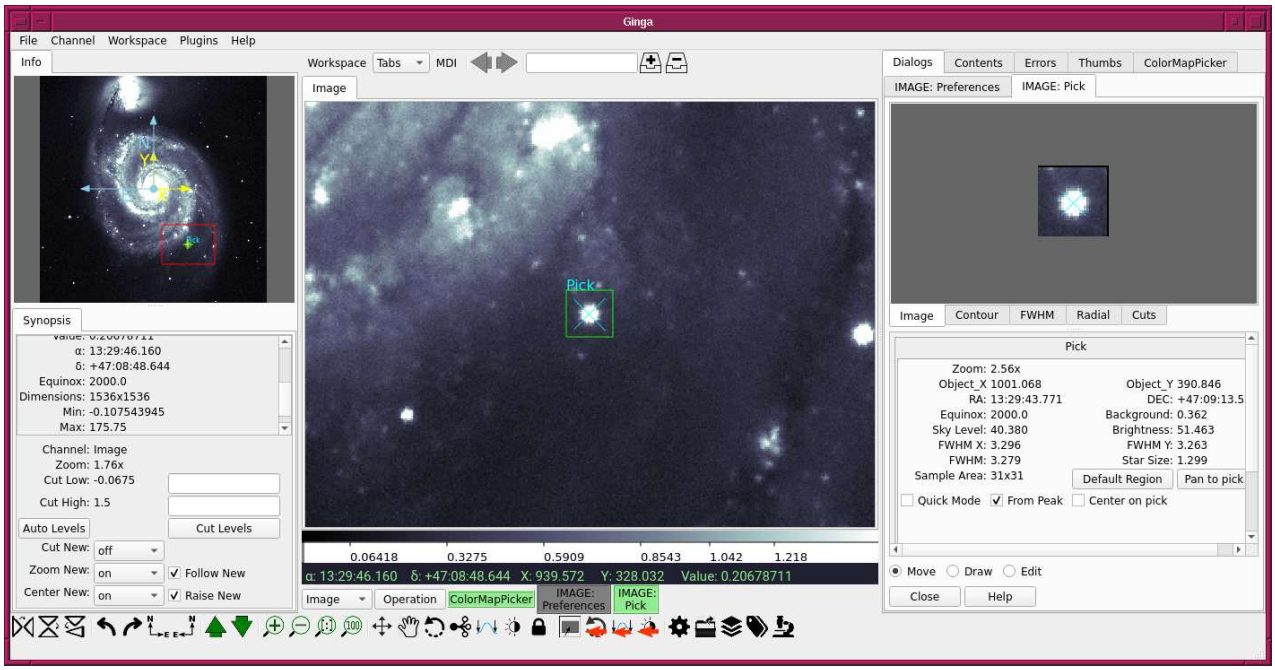

Figure 16: A star is picked by the plug-in “Pick”.

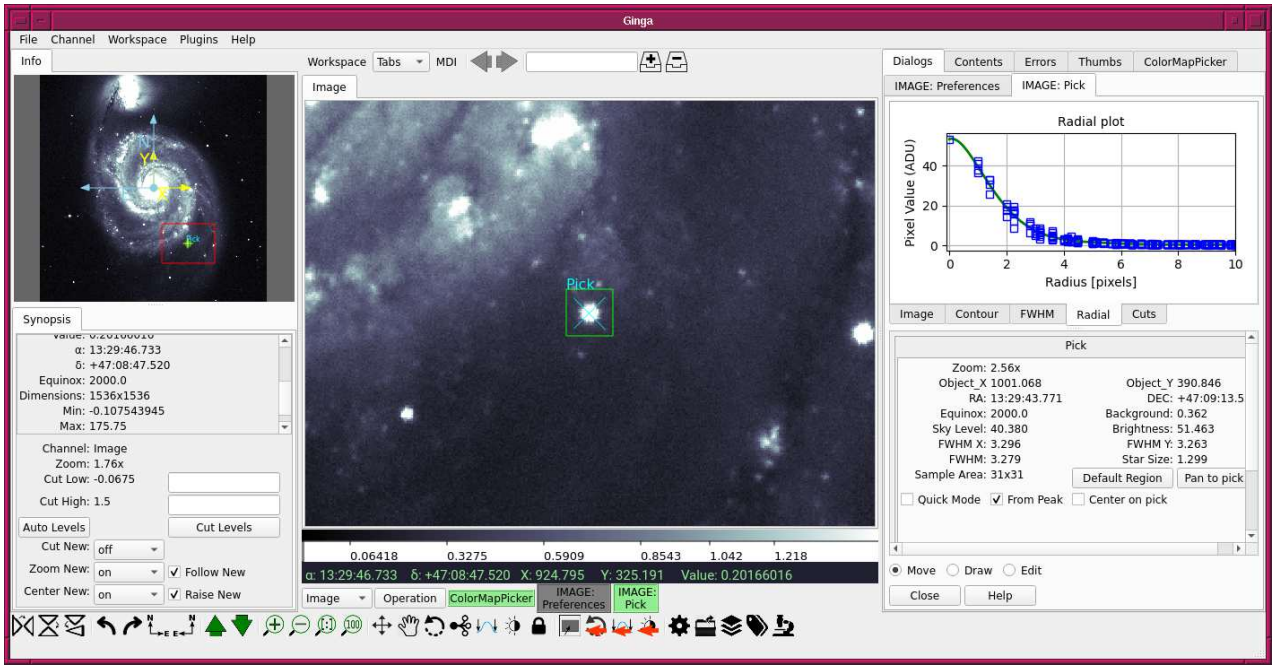

Figure 17: Radial profile of a star.

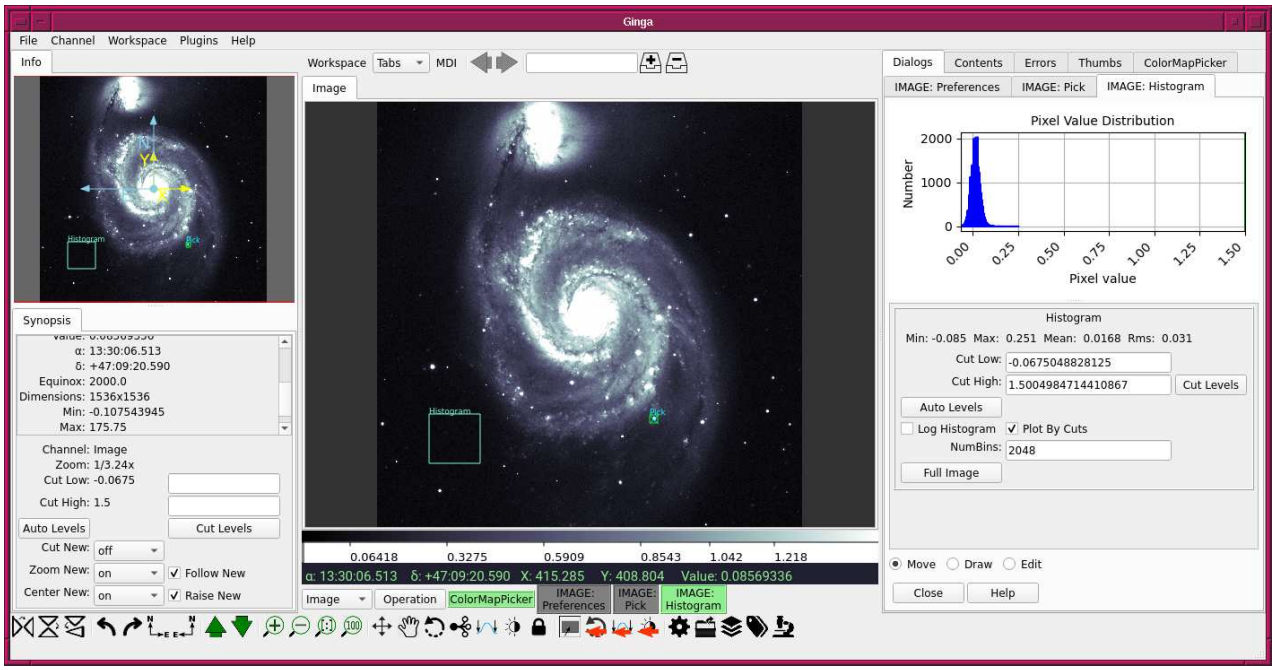

Figure 18: The histogram of the specified region in the image.

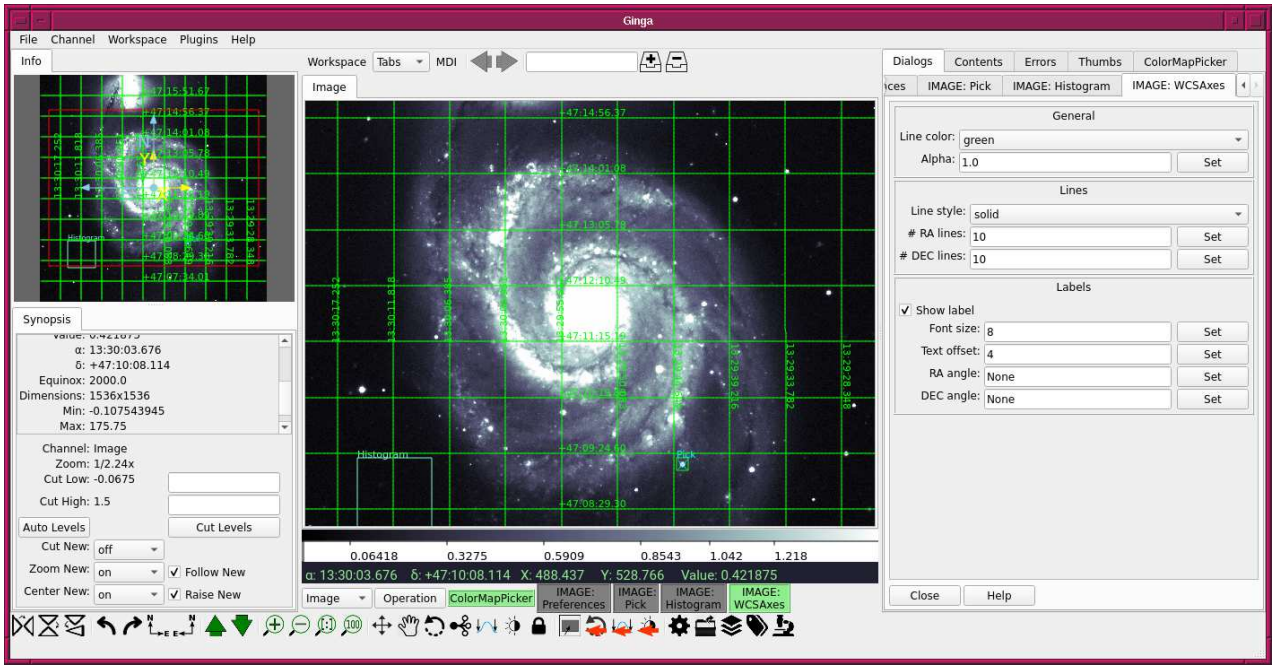

Figure 19: Showing WCS axes.

5.6 Making a line profile

Use “Cuts” to make a line profile.

1. Click the menu “Plugins”,
2. Move the mouse to “Analysis”,
3. Click the menu “Cuts”,
4. Click “Cut Width”, and set “Width Type” and “Width radius”,
5. Use right button of the mouse to define start point and end point for the line profile.

Then, the line profile is shown at the right-hand side of the window. (Fig. 20)

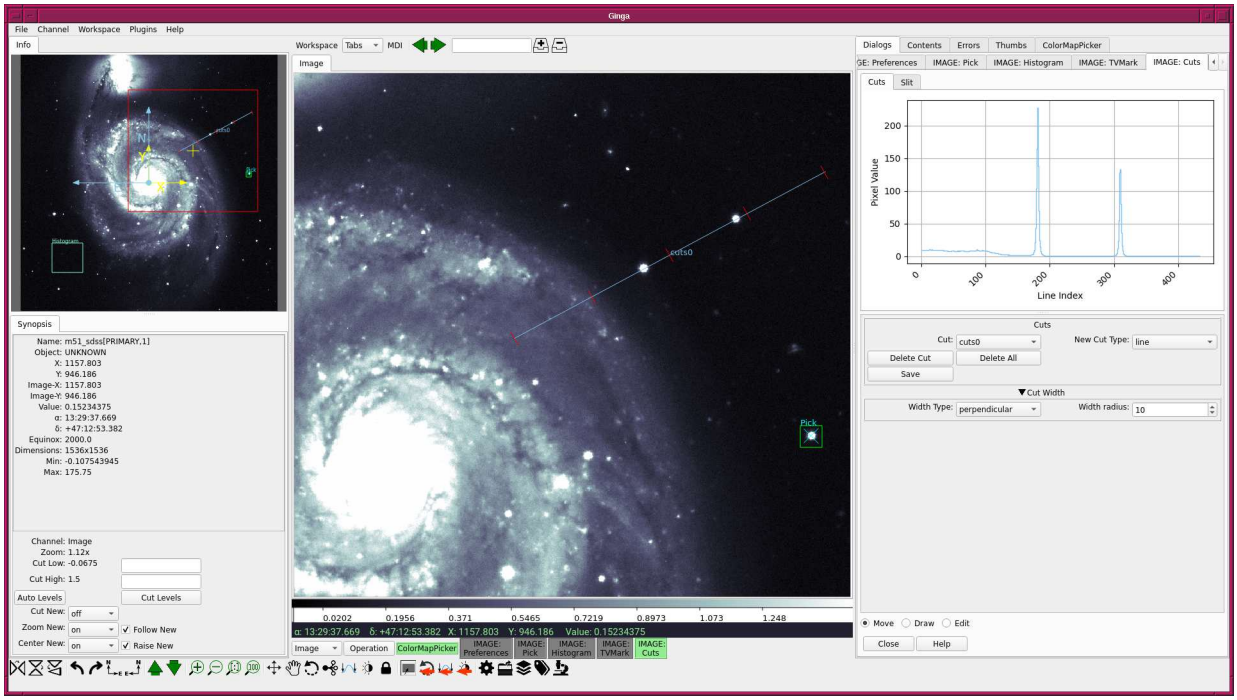

Figure 20: An example of line profile by “Cuts” plug-in.

5.7 Showing pixel table

Use “PixelTable” to show a pixel table.

1. Click the menu “Plugins”,
2. Move the mouse to “Analysis”,
3. Click the menu “PixelTable”,
4. Move the mouse cursor to the point of your interest.

Then, you see a pixel table at the right-hand side of the window. (Fig. 21)

6 Making marks on the image

Use “TVMark” to mark objects of your interest.

1. Click the menu “Plugins”,
2. Move the mouse to “Analysis”,

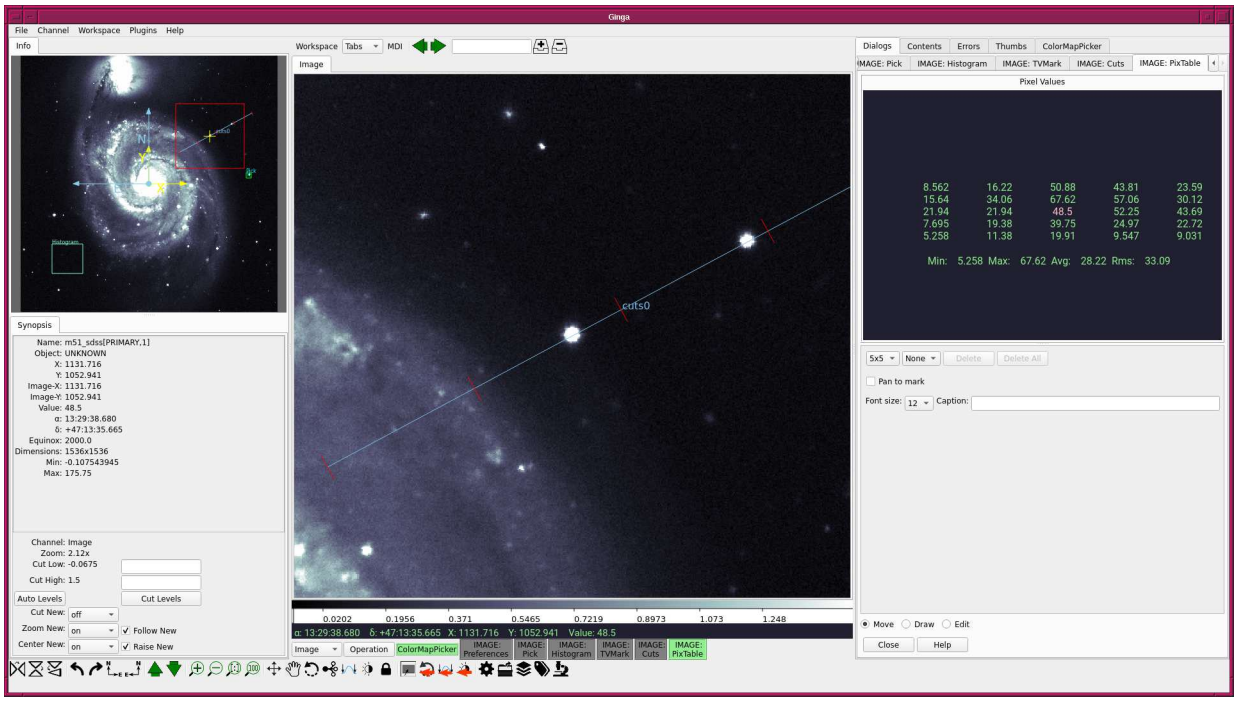

Figure 21: An example of pixel table by “PixelTable” plug-in.

3. Click the menu “Pick”,
4. Click the “Report” button and show the Report panel,
5. Move the mouse cursor to an object of your interest and click right button of the mouse,
6. Push “Add Pick” button,
7. Repeat steps 5 and 6 until you pick all the objects of your interest,
8. Push the button “Save Table”, (Fig. 22)
9. Click the menu “Plugins”,
10. Move the mouse to “Analysis”,
11. Click the menu “TVMark”,
12. Make sure that “Use RADEC” is unchecked,
13. Set your favourite “Mark”, “Color”, “Size”, and “Width”,
14. Push “Load Coords” button and select the file `pick_log.fits`,

Then, you see marks on the image. (Fig. 23)

7 For your training

- Read the official document of Astroquery module to learn about the usage of Astroquery.
  - `https://astroquery.readthedocs.io/en/latest/`
- Read the User’s Manual of Ginga to learn about the usage of Ginga.
  - `https://ginga.readthedocs.io/en/stable/manual/index.html`
- Download FITS images using Astroquery, and play with Ginga.

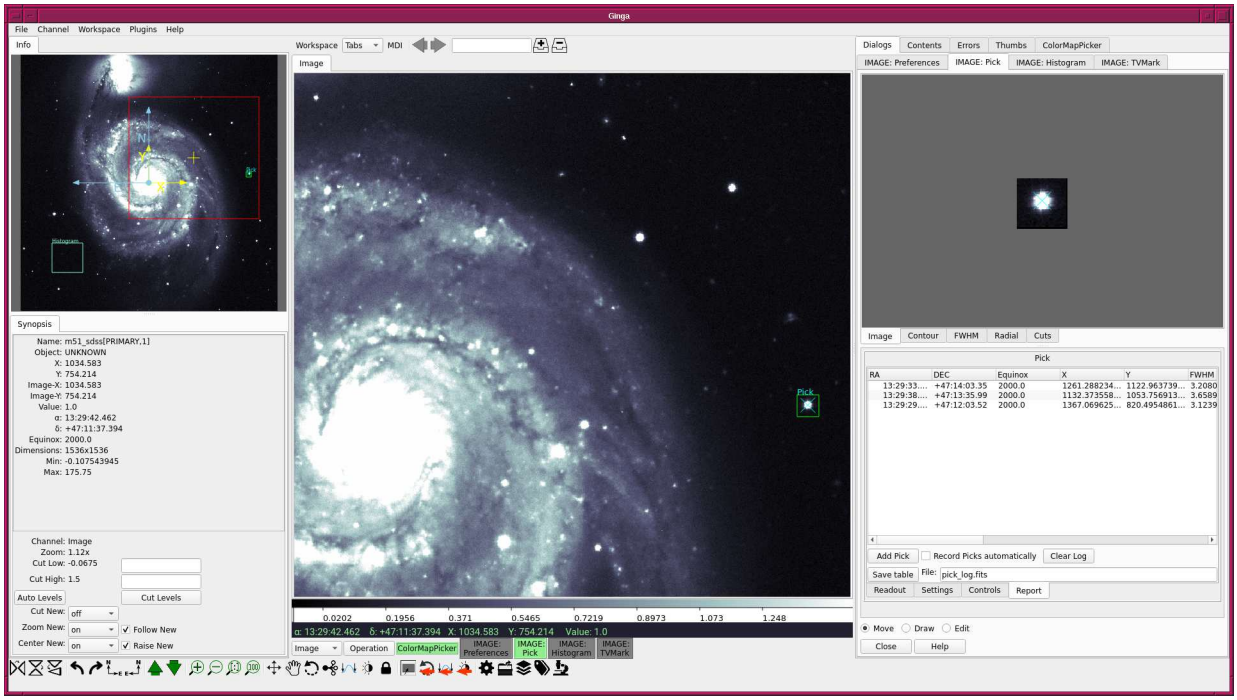

Figure 22: Picking objects and saving into a FITS table.

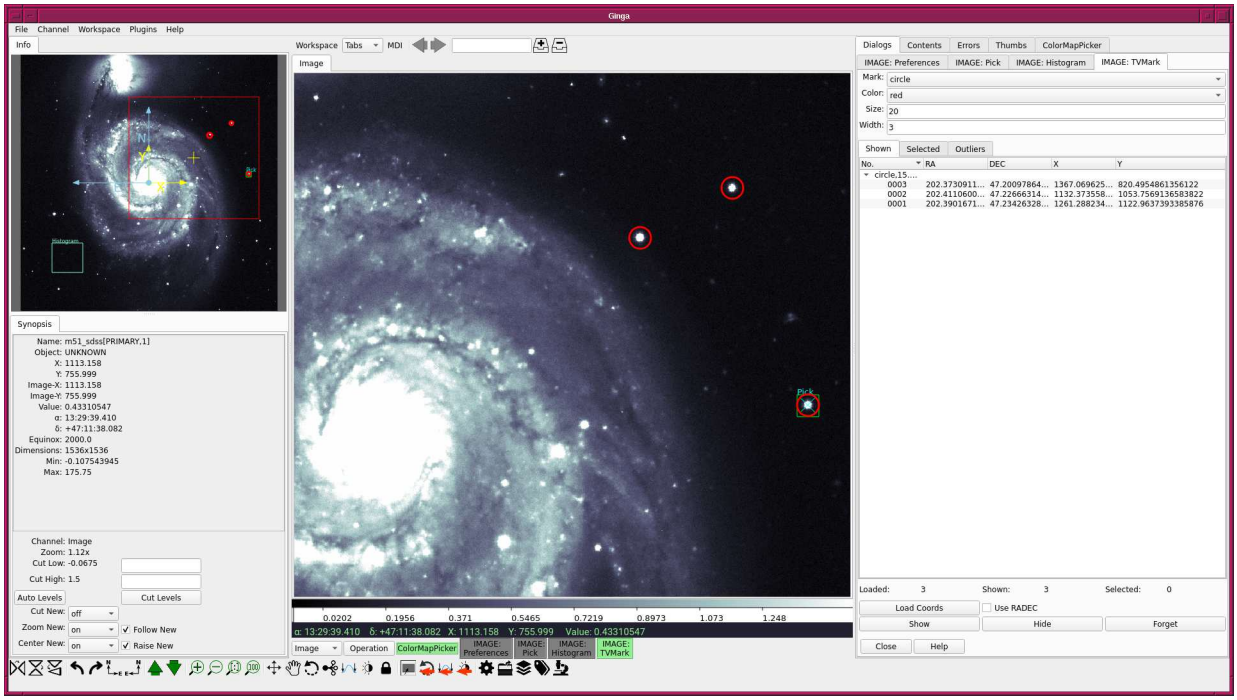

Figure 23: Showing marks on the image.

8 Assignment

There is no assignment for this session.