

README for:

The GLASS inspection GUI (GiG) & The GLASS inspection GUI for redshifts (GiGz)

GLASS Website: <http://glass.astro.ucla.edu/>
GLASS Public Data: <https://archive.stsci.edu/prepds/glass/>

Kasper B. Schmidt

December 18, 2015

Contents

1 The GLASS inspection GUI (GiG)	2
1.1 ‘Install’ and Requirements	2
1.2 An Overview of the GiG	2
1.3 Running The GiG	3
1.3.1 GiG ‘features’	4
1.4 Opening png and fits Files With The GiG	5
1.5 The GiG Output file	5
1.6 Select Objets Based on the GiG Output	6
1.6.1 A GiG Selection: Example 1	7
1.6.2 A GiG Selection: Example 2	7
1.7 GiG FAQ	7
1.7.1 I have ds9 installed but when I run GiG i get ”ds9: command not found”	7
1.7.2 Why do I get a “C module” import error (PIL error) when launching GiG	8
1.7.3 A Window Seems to be Opened and Dies Immediately when I Start GiG - Anything Wrong?	8
1.7.4 Why do I Get a ”SystemExit” When Launching GiG?	8
1.7.5 Why Isn’t GiG Closing the DS9 and PNG Windows When Advancing to the Next Object?	8
1.7.6 Why are the Keyboard Shortcuts Not Working?	8
1.7.7 GiG returns an error that ”x11/xlib.h’ file not found”	8
1.7.8 What do the red circles overlaid the SCI-CONTAM image in DS9 mean?	8
1.7.9 How do spectra with mild, moderate, and severe contamination look?	9
1.7.10 The GiG doesn’t open and I get a weird error message (after updating Mac OSX)	9
2 The GLASS inspection GUI for redshifts (GiGz)	11
2.1 ‘Install’ and Requirements	11
2.2 An Overview of the GiGz	11
2.3 Running The GiGz	12
2.3.1 GiGz ‘features’	13
2.4 Opening png and fits Files With The GiGz	14
2.5 Inspecting Redshift Fits with GiGz	14
2.6 The GiGz Output file	15
2.7 Select Objets Based on the GiGz Output	15
2.7.1 A GiGz Selection: Example 1	15
2.7.2 A GiGz Selection: Example 2	15
2.8 GiGz FAQ	15
2.8.1 Should I always give a by-hand-redshift?	15
2.8.2 Why does it take forever to advance to the next object (and plot the 1D curves)	16
3 Appendix	17
3.1 Examples of contamination	17

1 The GLASS inspection GUI (GiG)

The following describes the GLASS inspection graphical user face (GUI), or GiG for short, used to inspect the grism spectroscopy products from the Grism Lens-Amplified Survey from Space (**GLASS**), identifying emission lines and rating contamination from neighboring objects in the dispersed *HST* WFC3 grism field-of-views of GLASS. The GiG output can be used to select objects of a certain type and contamination. Also, the GiG output works as an optional input for the GLASS inspection GUI for redshifts (GiGz) described in Section 2.

1.1 ‘Install’ and Requirements

To ‘install’ the GiG and GiGz simply download the script `visualinspection.py` from [the GiG and GiGz GitHub repository](#). When positioned in the directory where `visualinspection.py` was downloaded to, or by adding the path of that directory to the system’s (PYTHON)PATH variable, the GiG can be launched.

To run the GiG and GiGz successfully the following python modules should be available (most of which come with most default python installations):

```
os
re
sys
PIL
pdb
glob
time
pyfits
numpy
Tkinter
datetime
commands
subprocess
scipy.ndimage
matplotlib.pyplot
```

If not accessible these can be installed with, e.g., `pip`.

Furthermore a functioning *command line* version of **DS9** should be available in order to display the fits files, i.e., the following command should open `fitsimage.fits` without errors:

```
ds9 -geometry 1200x600 -scale zscale fitsimage.fits
```

The GiG can in principle be run without DS9 installed but that disables the “open fits files” button and only the png images can be inspected.

The GiG and GiGz were build on MacOS 10.8.5 (Mountain Lion) using Python 2.7. It was run successfully run on a linux machine with the STScI Ureka version of Python 2.7 available at <http://ssb.stsci.edu/ureka/>. It was also successfully run on Mac OS X 10.9 (Yosemite), 10.10 (Mavericks), and 10.11 (El Capitan), with a few exceptions as described in the GiG FAQ in Section 1.7. A few problems running on Mac OS X 10.6.8 with Python 2.6 were found and fixed.

For instructions on how to launch the GiG and GiGz and start inspecting objects see Section 1.3 and 2.3

1.2 An Overview of the GiG

The GiG eases visual inspection of the GLASS data products. It is a set of python scripts and classes that opens a graphical user interface (GUI) when launched, which can be used to inspect and classify 2D grism spectra. Keywords can be assigned to each object’s spectrum and will be stored to an ascii output file (see Section 1.5). As described in Section 1.6 this output file simplifies searches for specific objects in the GLASS data.

In Figure 1 the main window of the actual GUI is shown with a few short explanations of the sections used when inspecting objects. The GiG consists of several main parts:

- **P.A. division:** The top of the GiG window is divided into two identical parts, one for each of the two position angles (P.A.) of the GLASS data. If only files for one position angle are found for a given object the bottom half of keywords are un-responsive and the bottom comment field is detached from the output.
- **Keywords:** Two sets of boxes (one for each P.A.) to indicate features and characteristics of the displayed object. By default no keywords are set. Note that the keywords are split between keywords referring to the G102 and G141 spectra as well as the direct image of the object for the main WFC3 IR pointings. They keywords can also be set with the keyboard shortcuts in ‘()’.

- **Comments:** The comments fields allow the user to add comments not covered by the keywords. These will be appended the object’s row in the ascii output file. Note that when a ‘comments’ field is active the keyboard shortcuts are disabled. Use the ‘tab’-key to change the focus and re-enable the keyboard shortcuts.
- **Wavelengths:** Four comment fields ($2 \text{ grisms} \times 2 \text{ P.A.s}$) allowing the user to specify the wavelength of any detected emission lines or interesting features occurring in the 2D spectra. The wavelengths are obtained by opening the fits files and then ‘hovering’ over the location of the emission line with the mouse curser and reading off the wavelength value (in Ångstrom) from the DS9 window.
- **‘Open Fits Files’ button:** This button opens the 2D fits files available for the current object using DS9 from the command line. It opens the direct image (fits extension DSCI), the 2D spectrum (fits extension SCI), and the 2D contamination model (fits extension CONTAM) for the available grisms and position angles. It also opens an image of the 2D spectrum subtracted the contamination model if it exists (and overlays the corresponding region file if that also exists). If the image of the contamination-subtracted spectrum does not already exists, it is created and displayed.
- **Object Controls:** At the bottom of the GiG there is a set of buttons to control which object is displayed. Note, that when moving between objects, unedited inspections will also be saved to the output file, i.e., and empty entry for the given object. This behavior can be changed with by setting `skipempty=True` as described in Section 1.3.1.

‘Previous Object’: Moves back to the previously displayed object. It saves the current inspection to the output file while doing so. Hence, if the ‘previous object’ is inspected again, this object and its inspection will appear twice in the output catalog. This can be prevented either by editing the output file by hand, or running the GiG with the `check4duplicates=True` (see Section 1.3.1). This keyword will search the output file every time a new inspection is written to it for duplicates. If it finds matching inspections it will delete these and only store the new inspection.

‘Quit GUI’: Quits the GUI. Note that quitting the GiG this way *does not* save the inspection of the current object, i.e. it is similar to aborting the current inspection.

‘Skip Object’: Advances to the next object without saving the inspection result to the output file.

‘Next object (save)’: Advances to the next object saving the inspection result to the output file.

- **Figure in GiG window:** The bottom part of the GiG window by default displays the `zfit` figure summarizing the automatic redshift fits, used to evaluate the redshift fits with the GiGz as described in Section 2. This figure contains three panels which show: (*left*) The 4(2) extracted 1D spectra for the 2(1) PAs in cgs units with the `zfit` models over plotted (thin black lines). (*center*) The photometry (blue dots) and the best-fit SED to the photometry with the spectra from the left panel over plotted. (*right*) The $p(z)$ for the 4(2) redshift fits compared to the $p(z)$ for the EAZY photo-z fit (blue curve). Alternatively, the user can choose to display a figure of the stacked 2D spectra similar to the one shown in Figure 2 using the ‘`inGUImage`’ keyword when launching GiG. If the selected figure is not available a G102 1D.png figure is displayed.

1.3 Running The GiG

The GiG is run on a directory containing the reduction products of the GLASS reduction. These can be downloaded from the public data repository at the STScI MAST server at <https://archive.stsci.edu/prepds/glass/>. For members of the GLASS collaboration, data can also be downloaded as instructed on [the internal part of the GLASS website](#).

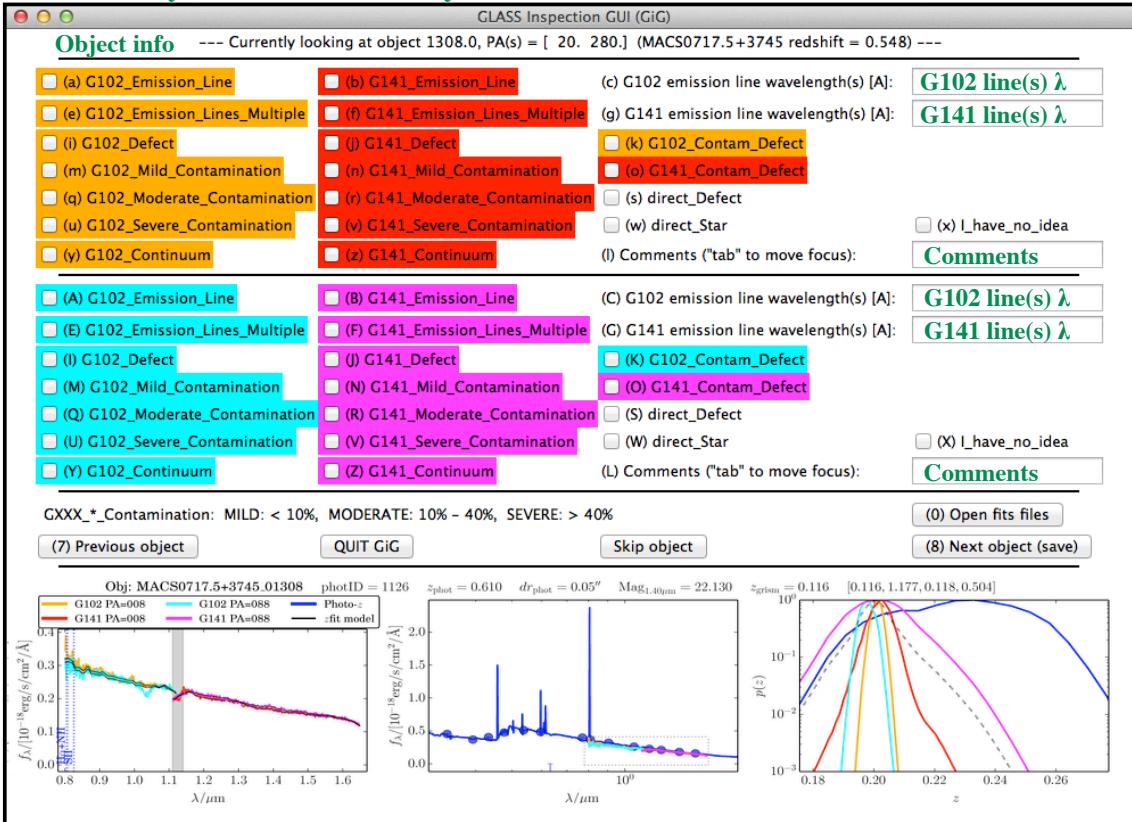
Having downloaded the GLASS data files for the objects to inspect either go to the directory containing `visualinspection.py` or add it to your (PYTHON)PATH. The GiG can then be run by starting a Python session and executing:

```
import visualinspection as vi
datadir = 'name/of/directory/to/run/GiG/on/'    # NB: End with '/'
objlist = None
outputGiG = 'GiG_output_test.txt'
name = 'YourName'
vi.launchgui(directory=datadir,objlist=None,verbose=True,outputfile=outputGiG,inspectorname=name,
             MASTfiles=True)
```

Setting `objlist` to `None` will run the GiG on the reduction products of all the objects it finds in the data directory. The `objlist` can also be set to a list of IDs if specific objects are to be inspected.

G102 Keywords

G141 Keywords



**The 4(2) fluxed spectra:
2(1) PAs x 2 grisms**

**Photometry and best-fit SED
with spectra from left panel**

**$p(z)$ for photometry and
individual grism spectra**

1st PA

2nd PA

Figure 1: An overview and brief description of the GLASS inspection GUI (GiG) window. The bottom panel figure can be changed to showing the G102 or G141 stacked spectra using the keyword ‘inGUIimage’

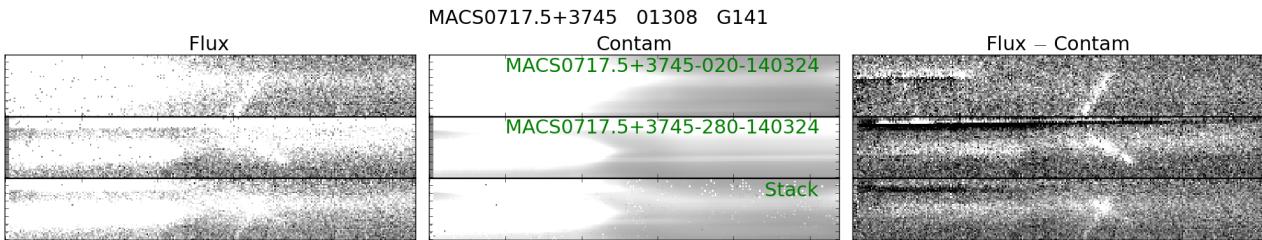


Figure 2: Example of the figures showing the stacked 2D spectra. The first two rows show the 2D spectra of the two position angles and the bottom row shows the stack. When available this figure (for G102 or G141) can be included shown in the GiG window instead of the default zfit figure shown in Figure 1 using the keyword ‘inGUIimage’ when launching GiG. All figures are opened in the PNG viewer when available.

1.3.1 GiG ‘features’

In this section a couple of the features available when launching the GiG are presented.

- **MASTfiles=True** (default value **False**)
Set this keyword to **True** if running the GiG on files downloaded from the MAST GLASS data product repository at <https://archive.stsci.edu/prepds/glass/> (filenames named `h1sp_glass_hst_wfc3_CLUSTER*`).
- **clobber=True** (default value **False**)
This enables the output file to be overwritten if it already exists instead of starting after the last classified object and appending the results. The GiG will prompt the user for confirmation that the file should be overwritten.
- **ds9xpa=True** (default value **False**)

If this keyword is set the ds9 window displaying the fits images of the individual objects will be updated using XPA (<http://hea-www.harvard.edu/RD/xpa/>). If this keyword is set it is implicitly assumed that XPA is installed on the system. When updating ds9 with XPA the ds9 window only has to be opened the first time a set of fits files are inspected. Note that the XPA-update changes the settings of already opened ds9 sessions, hence XPA should be avoided if you want to preserve the content of existing ds9 windows not related to the GiG.

- **openfitsauto=True** (default value **False**)

Setting **openfitsauto=True** will automatically open the fits files for each individual object. If used together with **ds9xpa=True** this does not add any considerable object loading time.

- **inGUIimage='G102stack'** (default value '**zfit**')

Setting **inGUIimage='G102stack'** or **inGUIimage='G141stack'** will display a figure similar to the one shown in Figure 2 showing the stack of the G102 or G141 spectra in the GUI window instead of the default **zfit** figure. If the stack figure is not available a G102 1D.png figure will be shown as 'filler' .

- **check4duplicates=True** (default value **False**)

If this feature is enabled the GiG will check the output file of existing inspections of the current object-P.A. pair before writing to the file. If an existing inspection(s) is found in the output file GiG replaces that with the new inspection. Hence, this keyword guarantees that no duplicate inspections exist in the output file (for the objects inspected with the keyword set).

- **outputcheck=True** (default value **False**)

When set the output file will be checked when the GiG is quit. The objects and P.A.s in the output file will be counted and compared to the number of objects provided in the **objlist** keyword. This makes it easy to check that all the objects to be inspected exist in the output file.

- **skipempty=True** (default value **False**)

By default unedited objects are written to the output file. Setting **skipempty=True**, output will only be stored if the default flags are changed or a comment/wavelength is provided.

1.4 Opening png and fits Files With The GiG

When running the GiG the png files for each object will automatically be opened when navigating between the objects. Among the pngs opened for each position angle and each grism is a 1D representation of the extracted spectrum and the contamination. and a mosaic of the fits extensions of the extracted 2D spectra. Hence, for a given object with data available in both the G102 and G141 grisms as well as for both P.A.s at least 8 pngs will be opened. An example of the main pngs for a single grism and a single PA are shown and explained in Figure 3. If desired the corresponding 2D fits images can be opened with ds9 from within the GiG using the 'Open fits files' button or the **openfitsauto** keyword as described above. The default fits window is shown and explained in Figure 4.

1.5 The GiG Output file

The GiG outputs an ascii file with a short header containing a time stamp, the inspector name, and the column names. The first column gives the object ID and the second column gives the P.A. of the reduction products inspected. The remaining columns indicate whether a given key was set (1) or not (0). Each row is trailed by the input to the emission line wavelength fields and the comment field. Hence the GiG output file has the following format:

```
# Results from Visual Inspection initiated on 2015-10-21 16:29:00
# Inspector: John Doe
# ID PA G102_Emission_Line G141_Emission_Line G102_Spectral_Coverage G141_Spectral_Coverage
G102_Emission_Lines_Multiple G141_Emission_Lines_Multiple G102_Contamination_Level
G141_Contamination_Level G102_Defect G141_Defect G102_Contam_Defect Spectral_Coverage_Type
G102_Mild_Contamination G141_Mild_Contamination G141_Contam_Defect Contamination_Level_Type
G102_Moderate_Contamination G141_Moderate_Contamination direct_Defect empty7
G102_Severe_Contamination G141_Severe_Contamination direct_Star I_have_no_idea G102_Continuum
G141_Continuum
00011 020 1 1 0.99722 0.99679 0 1 0.09003 0.15606 0 0 0 1 0 0 0 1 1 1 0 -1 0 0 0 0 1 1 #G102wave# 9146
#G141wave# 12304, 16047 #C#
00011 280 1 1 0.53333 0.72115 0 1 0.00000 0.15614 0 0 0 1 1 1 0 1 0 0 0 -1 0 0 0 0 1 1 #G102wave# 9134
#G141wave# 12292 #C#
00018 020 0 0 1.00000 0.96474 0 0 0.73849 0.91960 0 0 0 1 1 1 0 1 0 0 0 -1 0 0 0 0 1 1 #G102wave#
#G141wave# #C#
```

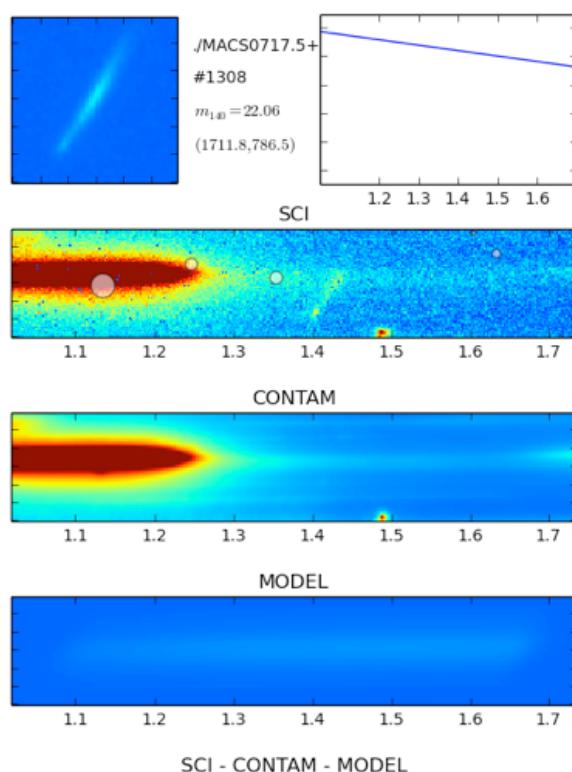
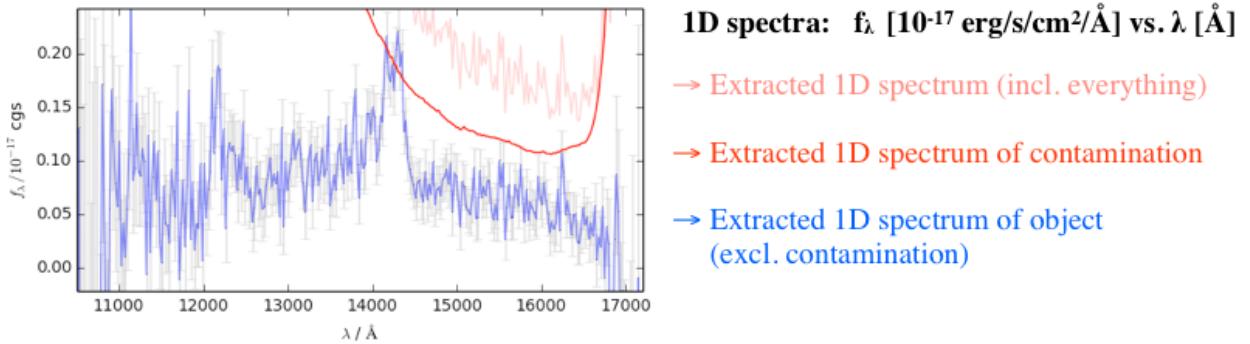


Figure 3: The default .png files opened for each position angle and each grism.

```

00018 280 0 0 0.50278 0.65385 0 0 0.53650 0.87775 0 0 0 1 1 1 0 1 0 0 0 -1 0 0 0 0 1 1 #G102wave#
#G141wave# #C#
01308 020 1 1 1.00000 0.99038 0 0 0.45569 0.63834 0 0 1 1 0 0 0 1 1 1 0 -1 0 0 0 0 1 1 #G102wave# 10622
#G141wave# 14222 #C# This is a comment
01308 280 1 1 1.00000 0.98077 0 0 0.21756 0.36848 0 0 0 1 0 0 0 1 0 0 0 -1 1 1 0 0 1 1 #G102wave# 10624
#G141wave# 14290 #C# Extended lines
.
.
.
```

1.6 Select Objects Based on the GiG Output

The output from a GiG inspection is as mentioned a simple ascii file where each column represents a given keyword. Hence, this file can be used to select objects based on these keyword. Below are a couple of examples on how objects can be selected using the GiG output.

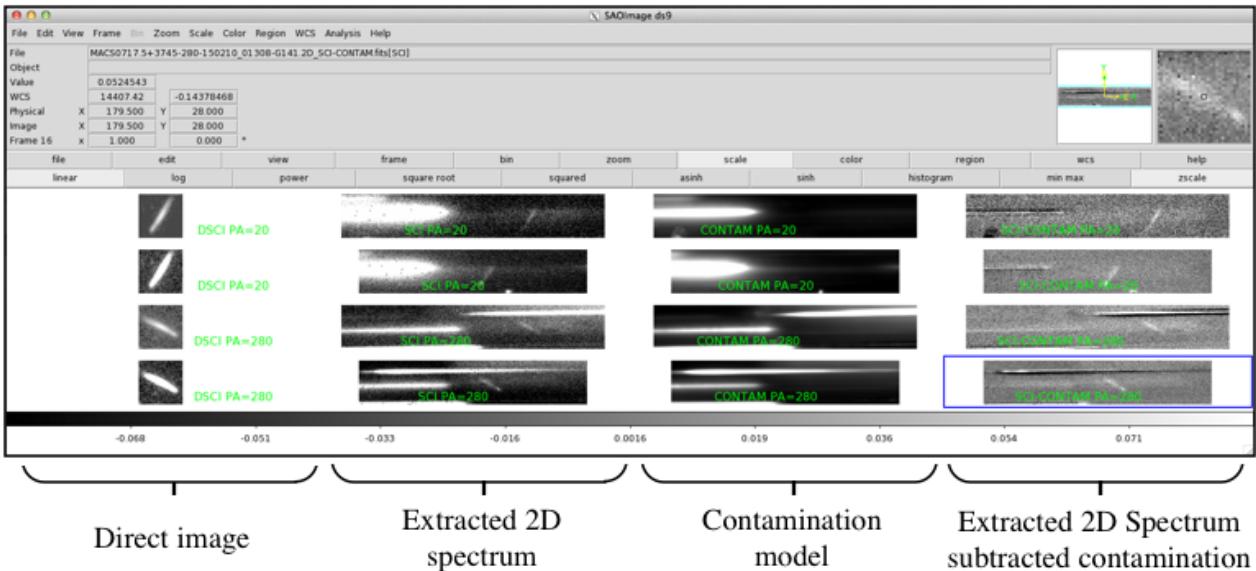


Figure 4: The default .fits files opened for each position angle and each grism. The first two rows correspond to the G102 and G141 spectra for the first P.A. and the bottom two rows for the second P.A. (if it exists). Red circular markers (not shown here) in the forth column *do not* indicate contamination. These markers are the result from running SExtractor on the SCI-CONTAM spectrum (see FAQ Section 1.7.8 for further explanation).

1.6.1 A GiG Selection: Example 1

To select objects with emission lines in G102 from one of the pointings (one P.A. given) do the following in Python:

```
GiGoutput = '/GiG/output/file/to/read/from.txt'
data = np.genfromtxt(GiGoutput, comments='#', skip_header=2, names=True)
PA = 88.0
selection = data[(data['G102_Emission_Line'] == 1) & (data['PA'] == PA)]
print '\nThe IDs of objects with PA =',PA,' and emission lines in G102:\n',selection['ID']
```

1.6.2 A GiG Selection: Example 2

To select objects looking irregular or looking like a merger in the direct images with multiple emission lines and no severe contamination in G141 do the following in Python:

```
GiGoutput = '/GiG/output/file/to/read/from.txt'
data = np.genfromtxt(GiGoutput, comments='#', skip_header=2, names=True)
PA = 88.0
selection = data[np.logical_or(data['direct_Merger'] == 1, data['direct_Irregular'] == 1) &
                (data['G141_Emission_Lines_Multiple'] == 1) & (data['G141_Severe_Contamination'] == 0)]
print '\nThe IDs of irregular/merger-like objects with ELs and no severe contamination in G141:\n',\
      selection['ID']
```

1.7 GiG FAQ

In this section you'll find answers to some of the frequently asked questions about the GiG. Feel free to email [Kasper](#) with any questions or issues you might run into, if you are not helped by the FAQ.

1.7.1 I have ds9 installed but when I run GiG i get "ds9: command not found"

This error most probably occurs because ds9 is not installed in the `sh` shell which is what GiG uses to run commands parsed to the command line (including ds9 commands). You can test this by typing `sh` and then executing `ds9`.

To make ds9 available from the `sh` command line create the file `/bin/ds9` with the following content (adjust the ds9 path as needed):

```
#!/bin/sh
/Applications/ds9.darwinsnowleopard.7.2/ds9 "$@"
```

This should enable the `ds9` command in `sh` and hence enable GiG to use it.

1.7.2 Why do I get a “C module” import error (PIL error) when launching GiG

If you get an error similar to:

```
ImportError: The _imaging C module is not installed
```

encountered in PIL/Image.py there seems to be a problem with the PIL installation. Try installing PIL using pillow (<https://pillow.readthedocs.org>) instead.

1.7.3 A Window Seems to be Opened and Dies Immediately when I Start GiG - Anything Wrong?

No. What you see is a DS9 windows being opened and closed again. This is done to get the version of DS9 to decide whether the ‘lock’ keyword is available for displaying fits files (it’s only available in DS9 version 7 or newer).

1.7.4 Why do I Get a ”SystemExit” When Launching GiG?

The output file you are trying to write to already exists and the last classified object in the file corresponds to the last object in the list of objects created when launching GiG with `objlist=None` or in the list you provided. Either you can remove or rename the output file or you can provide a list to the `objlist` keyword instead of ‘None’. **NB!** Make sure to close the empty GiG window before starting a new GiG session.

The iPython traceback will be similar to:

```
- The file /Users/kasperborellschmidt/work/GLASS/MACS0717test/vanzellaOBJ/inspection_output_yymmdd.txt  
already exists (Resuming after last objects in output)  
An exception has occurred, use %tb to see the full traceback.
```

```
SystemExit: - The last object in outputfile is the last in "objlist" --> ABORTING
```

To exit: use ‘exit’, ‘quit’, or Ctrl-D.

1.7.5 Why Isn’t GiG Closing the DS9 and PNG Windows When Advancing to the Next Object?

The PNG files and the fits files (opened with DS9) are opened in new windows and the process ids are stored in memory. When advancing to the next object GiG tries to terminate these process ids which should close the windows. If the process IDs for some reason changed(?) or are not the ones returned to python when opening the windows, python can’t close the windows and they stay open.

Try closing GiG as well as all the opened windows and start over. Make sure the last classified object in the output list is as intended as GiG will start with the object after that in the list.

1.7.6 Why are the Keyboard Shortcuts Not Working?

The keyboard input is ignored if the GiG window is not selected/active or if a comment field is active. In the latter case you shift the focus between the comment fields and checkboxes with the ‘tab’-key.

1.7.7 GiG returns an error that ”x11/xlib.h’ file not found”

If GiG is failing to open and dies with an error containing ”fatal error: ‘X11/Xlib.h’ file not found” this might be a problem with the X11 installation running on the Mac. Try opening your terminal and execute: `xcode-select –install`

If this doesn’t work try some of the other suggestions [on stackoverflow.com](#)

1.7.8 What do the red circles overlaid the SCI-CONTAM image in DS9 mean?

The red circular markers with numbers overlaid on the SCI-CONTAM spectra in DS9 when displaying an object’s fits files are the result from running SExtractor on the SCI-CONTAM spectrum. Hence, the red circles mark clusters of high-flux pixels and therefore indicate potential emission lines, spurious features, or residuals from contamination subtraction. Irrespective of what they mark they are only meant as an extra tool of pinpointing emission lines when inspecting spectra.

An example from the GiG inspection trainings set, where these markers are useful to locate a line is shown in Figure 5. Here, the left-most markers in the G141 frames (second and fourth row of plots) mark the position of an emission line. The markers in the top G102 panel seem to have caught contamination not properly accounted for in the model, as these features do not show up in the second G102 SCI-CONTAM frame.

Note that these SExtractor region files are not produced by GiG. Hence, if they do not exist in the data directory nothing is displayed.

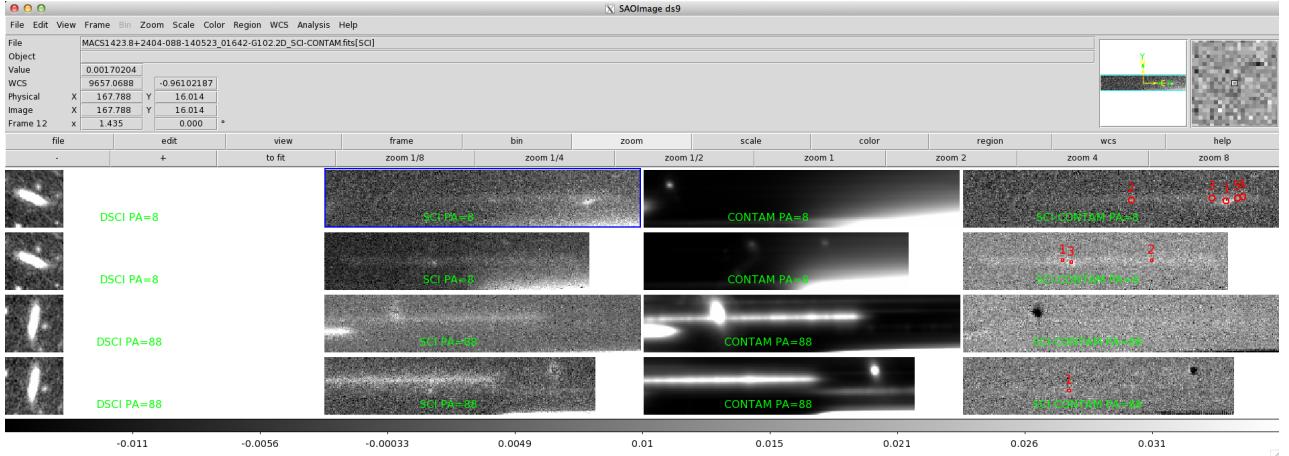


Figure 5: Illustration of the usefulness of the red circular markers from the SExtractor region file created on the SCI-CONTAM frame. The markers “1” in the second and fourth row in the SCI-CONTAM column mark a G141 emission line. One can even appreciate the rotation of the line as the P.A. changes.

1.7.9 How do spectra with mild, moderate, and severe contamination look?

The contamination level is judged mainly based on the ”CONTAM” frame of the figures showing the 2D spectra. As noted in Figure 3 this panel shows the contamination model and the 0th order images. Judging the contamination is done by estimating how much contaminating flux overlaps/contaminate the central rows of the 2D spectrum.

The general rule of thumb is:

- MILD: Very small fraction of the area is contaminated (5-10% at most)
- MODERATE: Smallish fraction contaminated at high intensity (< 40%) or larger fraction at low surface brightness
- SEVERE: More than 40% contaminated at high surface brightness

In Figures 8-22 (Starting Page 18) a sample of objects with different degrees of contamination are shown for reference. The individual figures are sorted according to increasing degree of contamination.

1.7.10 The GiG doesn’t open and I get a weird error message (after updating Mac OSX)

People have experienced problems with launching the TKinter package (the package that controls the GiG and GiGz graphical interface) and this is known to have potential ties to Mac OSX system upgrades (e.g., <https://www.python.org/download/mac/tcltk/>). The problem produces an error message similar to:

```
i.launchgui(directory=datadir, outputfile=output, inspectorname=username)
2015-12-11 10:46:22.847 Python[19385:3381498] -[NSApplication _setup:] unrecognized selector sent to instance 0x7fe4115c0e00
2015-12-11 10:46:22.851 Python[19385:3381498] An uncaught exception was raised
2015-12-11 10:46:22.851 Python[19385:3381498] -[NSApplication _setup:] unrecognized selector sent to instance 0x7fe4115c0e00
2015-12-11 10:46:22.851 Python[19385:3381498] (
0 CoreFoundation          0x000007fff9f90c5e32 __exceptionPreprocess + 178
1 libobjc.A.dylib          0x000007fff9837bf0 objc_exception_throw + 48
2 CoreFoundation          0x000007fff9f12f34d-[NSObject(NSObject) doesNotRecognizeSelector:] + 205
3 CoreFoundation          0x000007fff9f036661 __forwarding__ + 1009
4 CoreFoundation          0x000007fff9f0361e8 _CF_forwarding_prep_0 + 120
5 libtk8.6.dylib          0x0000000010e8518af TkpInit + 395
6 libtk8.6.dylib          0x0000000010e7c4fc2 Initialize + 2015
7 _tkinter.so              0x0000000010e629bc4 Tcl_AppInit + 84
8 _tkinter.so              0x0000000010e62947e Tkinter_Create + 1132
9 Python                  0x0000000010e82e503c PyEval_EvalFrameEx + 15692
10 Python                 0x0000000010e82e0ffa PyEval_EvalCodeEx + 1690
11 Python                 0x0000000010e82e6ccf function_call + 364
12 Python                 0x0000000010e8247023 PyObject_Call + 99
13 Python                 0x0000000010e8254056 instanceMethod_call + 182
14 Python                 0x0000000010e8247023 PyObject_Call + 99
15 Python                 0x0000000010e82e7e65 PyEval_CallObjectWithKeywords + 165
16 Python                 0x0000000010e8252076 PyInstance_New + 134
17 Python                 0x0000000010e8247023 PyObject_Call + 99
18 Python                 0x0000000010e82e3ff9 PyEval_EvalFrameEx + 11529
19 Python                 0x0000000010e82e0ffa PyEval_EvalCodeEx + 1690
20 Python                 0x0000000010e82e83e6 fast_function + 118
21 Python                 0x0000000010e82e4736 PyEval_EvalFrameEx + 13382
22 Python                 0x0000000010e82e0ffa PyEval_EvalCodeEx + 1690
23 Python                 0x0000000010e82e0956 PyEval_EvalCode + 54
24 Python                 0x0000000010e830996 PyRun_InteractiveOneFlags + 406
25 Python                 0x0000000010e8309656 PyRun_InteractiveLoopFlags + 206
26 Python                 0x0000000010e8309508 PyRun_AnyFileExFlags + 136
27 Python                 0x0000000010e831f6e2 Py_Main + 3186
28 libdyld.dylib           0x000007ffffa6b75ad start + 1
29 ???
0x00000000000000001 0x0 + 1
)
2015-12-11 10:46:22.852 Python[19385:3381498] *** Terminating app due to uncaught exception 'NSInvalidArgumentException', reason:
'-[NSApplication _setup:] unrecognized selector sent to instance 0x7fe4115c0e00'
*** First throw call stack:
(

```

```

0  CoreFoundation          0x000007fff9f0c5e32 __exceptionPreprocess + 178
1  libobjc.A.dylib         0x000007fff9837bdfa objc_exception_throw + 48
2  CoreFoundation          0x000007fff9f12f34d -[NSObject(NSObject) doesNotRecognizeSelector:] + 205
3  CoreFoundation          0x000007fff9f036661 ___forwarding___ + 1009
4  CoreFoundation          0x000007fff9f0361e8 _CF_forwarding_prep_0 + 120
5  libtk8.6.dylib          0x0000000010e8518af TkpInit + 395
6  libtk8.6.dylib          0x0000000010e7c4fc2 Initialize + 205
7  _tkinter.so              0x0000000010e629bc4 Tcl_AppInit + 84
8  _tkinter.so              0x0000000010e62947e Tkinter_Create + 1132
9  Python                  0x0000000010e82e503c PyEval_EvalFrameEx + 15692
10 Python                 0x0000000010e82e0ff PyEval_EvalCodeEx + 1690
11 Python                 0x0000000010e826ccfc function_call + 364
12 Python                 0x0000000010e8247023 PyObject_Call + 99
13 Python                 0x0000000010e8254056 instanceMethod_call + 182
14 Python                 0x0000000010e8247023 PyObject_Call + 99
15 Python                 0x0000000010e82e7e65 PyEval_CallObjectWithKeywords + 165
16 Python                 0x0000000010e8252076 PyInstance_New + 134
17 Python                 0x0000000010e8247023 PyObject_Call + 99
18 Python                 0x0000000010e82e3ff9 PyEval_EvalFrameEx + 11529
19 Python                 0x0000000010e82e0ff4 PyEval_EvalCodeEx + 1690
20 Python                 0x0000000010e82e83e6 fast_function + 118
21 Python                 0x0000000010e82e4736 PyEval_EvalFrameEx + 13382
22 Python                 0x0000000010e82e0ff4 PyEval_EvalCodeEx + 1690
23 Python                 0x0000000010e82e0956 PyEval_EvalCode + 54
24 Python                 0x0000000010e8309b96 PyRun_InteractiveOneFlags + 406
25 Python                 0x0000000010e830965c PyRun_InteractiveLoopFlags + 206
26 Python                 0x0000000010e8309508 PyRun_AnyFileExFlags + 136
27 Python                 0x0000000010e831f6e2 Py_Main + 3186
28 libdyld.dylib           0x000007ffff16b75fad start + 1
29 ???
0x00000000000000000000000000000001 0x0 + 1
)
libc++abi.dylib: terminating with uncaught exception of type NSException
Abort trap: 6

```

The problem appears to arise because the default backend of `matplotlib` is set to Mac OSX. Changing the `backend` variable in the `matplotlibrc` setup file to “`backend : TkAgg`” will make the TKinter package, and hence the GiG and the GiGz launch as intended. The location of the currently active `matplotlibrc` can be displayed with:

```

In [1]: import matplotlib
In [2]: matplotlib.matplotlib_fname()

```

For details on the `matplotlibrc` setup file see <http://matplotlib.org/users/customizing.html>.

2 The GLASS inspection GUI for redshifts (GiGz)

In the following the GLASS inspection graphical user faces (GUI) for redshifts, or GiGz for short, is described. The GiGz is used to inspect the grism redshift fits for GLASS objects (potentially pre-selected based on the GiG output described in Section 1) and assign a manual redshift based on these as well as photometric priors to each object.

2.1 ‘Install’ and Requirements

If GiG has been successfully imported and run from within python fulfilling the requirements listed in Section 1.1, GiGz will also be available and can be launched.

2.2 An Overview of the GiGz

The GiGz enables inspection of the automatic grism redshift fits `zfit`, and to manually estimate the redshift of the objects (potentially) using the inspection output from the GiG including the marked emission lines. As GiG, GiGz is a set of python scripts and classes that opens a GUI when launched, as well as an interactive python plotting window displaying the 1D spectra, which can be used to inspect and refined the redshift fits. The quality of the individual redshift fits, as well as the by-hand redshift are estimated on a scale from 0 to 4. It also enables marking any particular lines detected in the spectra. The results will be stored to an ascii output file (see Section 2.6). As described in Section 2.7 this output file simplifies searches for specific GLASS objects based on their redshift fits and/or their emission lines.

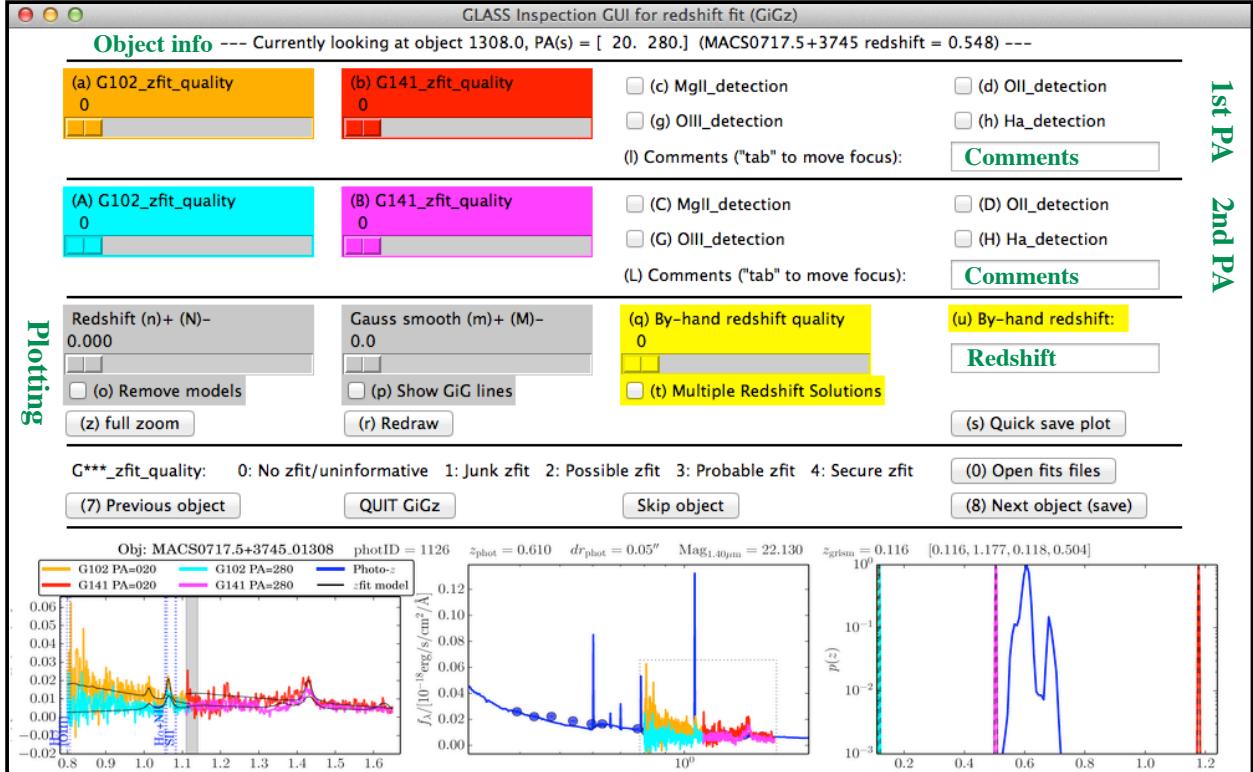
In Figure 6 the main window of the GiGz is shown with a few short explanations of the sections used when inspecting objects. The GiGz consists of several main parts:

- **P.A. division:** The top of the GiGz window is also divided into two identical parts, one for each of the two GLASS P.A.s. If only files for one P.A. are found for a given object, the bottom half of keywords are un-responsive and the bottom comment field is detached from the output like in the GiG window.
- **1D Plotting:** When loading an object in the GiGz an interactive plotting window displaying the 1D spectra of the individual GLASS spectra will also be opened. These plots are similar to the examples shown in Figure 7. This window allows the detailed inspection of the extracted GLASS 1D spectra to determine the best-fit redshift manually for the given object. The curves can be smoothing from the GiGz main window and be zoomed, panned and scaled using the controls in the interactive window. To guide-the-eye when identifying emission lines, an emission line list is over-plotted in green when a redshift is provide (bottom panel of Figure 7). If the spectra have been automatically fitted to obtain a redshift, these models will be over plotted, with an emission line list corresponding to the automatic best-fit redshift (top panel of Figure 7). If a GiG output file is provide when launching the GiGz, and an emission line has been marked for the given object, this information can be displayed on the 1D spectra (using the `Show GiG lines` box in the GiGz main window) as filled circles similar to the ones shown in the bottom panel of Figure 7.
- **Emission Line Boxes:** If a redshift can be determined and prominent emission lines determined, these can be indicated with the emission line boxes in the upper right corner of the GiGz main window.
- **Comments:** The comments fields allow the user to add comments to each object. These will be appended the object’s row in the ascii output file. Note that when a ‘comments’ field is active the keyboard shortcuts are disabled. Use the ‘tab’-key to change the focus.
- **`zfit` quality:** Sliders used to rate the quality of the redshift fits (see description below) are located in the upper left corner of the main GUI window. These fits were obtained from combining the 4 individual grism spectra with the ancillary (CLASH) photometry. The spectra are rated from 0 to 4 (see Figure 6). See Section 2.5 for details)
- **‘Open Fits Files’ button:** This button opens the 2D fits files available for the current object using DS9 from the command line. It opens the direct image (fits extension `DSCI`), the 2D spectrum (fits extension `SCI`), and the 2D contamination model (fits extension `CONTAM`) for the available grisms and position angles. It also open an image of the 2D spectrum subtracted the contamination model if it exists (and overlays the corresponding region file if that also exists). If the image of the contamination-subtracted spectrum does not already exists, it is created and displayed.
- **Object Controls:** At the bottom of the GiGz window there is a set of buttons to control which object is displayed. Note, that when moving between objects, unedited inspections will also be saved to the output file, i.e., and empty entry for the given object. This behavior can be changed with by setting

G102 z-fit rating

G141 z-fit rating

Emission Line Keywords



**The 4(2) fluxed spectra:
2(1) PAs x 2 grisms**

**Photometry and best-fit SED
with spectra from left panel**

**p(z) for photometry and
individual grism spectra**

Figure 6: An overview and brief description of the GLASS inspection GUI for redshifts (GiGz) window. The bottom panel figure can be changed to show the G102 or G141 stacked spectra using the keyword ‘inGUIimage’ (see Figure 2). The gray-shaded buttons next to “plotting” are used to control the plots of the 1D spectra shown in Figure 7.

skipempty=True as described in Section 2.3.1. For details on the individual buttons see description of objet controls in Section 1.2.

- **Figure in GiG window:** The bottom part of the GiG window by default displays the `zfit` figure used to evaluate the redshift fits. This figure contains three panels which show: (*left*) The 4(2) extracted 1D spectra for the 2(1) PAs in cgs units with the `zfit` models over plotted (thin black lines). (*center*) The photometry (blue dots) and the best-fit SED to the photometry with the spectra from the left panel over plotted. (*right*) The $p(z)$ for the 4(2) redshift fits compared to the $p(z)$ for the EAZY photo-z fit (blue curve). Alternatively, the user can choose to display a figure of the stacked 2D spectra similar to the one shown in Figure 2 using the ‘`inGUIimage`’ keyword when launching GiG. If the selected figure is not available a G102 1D.png figure is displayed.

2.3 Running The GiGz

Like the GiG, the GiGz is run on a directory containing the reduction products of the GLASS reduction. As described in Section 1.3 these can be downloaded from the STScI MAST server at <https://archive.stsci.edu/prepds/glass/>. (or from the internal part of the GLASS website).

Having downloaded the GLASS data products for the objects to inspect either go to the directory containing `visualinspection.py` or add it to your (PYTHON)PATH. The GiGz can then be run by starting a Python session and executing:

```
import visualinspection as vi
datadir = 'name/of/directory/to/run/GiGz/on/' # NB: End with '/'
objlist = None
```

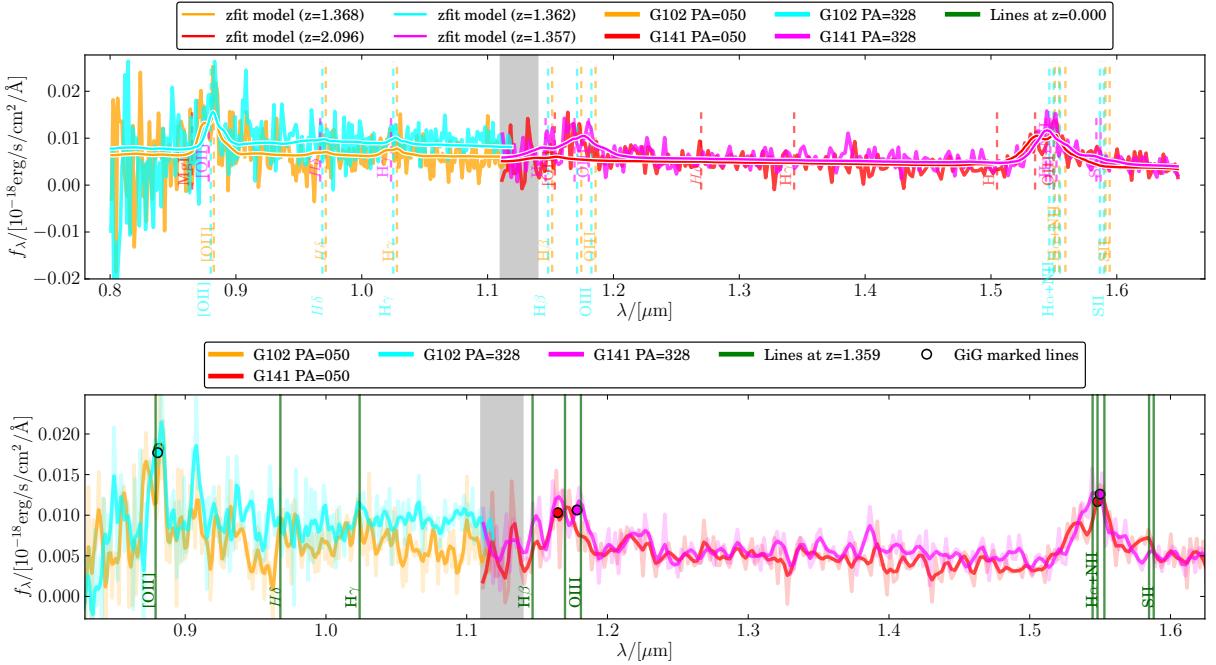


Figure 7: Examples of the interactive plotting windows displayed when running GiGz. Each panel displays the extracted 1D spectra of the individual grism spectra. In **the top panel** the redshift models from automatically fitting for the object redshift (also displayed in the bottom panel of the GiGz window shown in Figure 7) and the corresponding position of emission lines (vertical dashed lines) are overlaid the 1D spectra. **The bottom panel** shows a modified version of the top panel (modified using gray-shaded buttons in the GiGz window shown in Figure 7). In particular, the redshift models have been removed, the emission lines identified in the GiG output catalog have been overlaid (black circles), the 1D spectra have been smoothed (original 1D spectrum shown in light colors), and the position of emission lines at redshift 1.359 are indicated by the green vertical lines.

```
outputGiGz = 'GiGz_output_test.txt'
name = 'YourName'
vi.launchgui_z(directory=datadir, outputfile=outputGiGz, GiGfile=GiGfile, inspectorname=name, objlist=None,
                MASTfiles=True)
```

Setting `objlist` to `None` will run the GiGz on the reduction products for all the objects it finds in the data directory. `objlist` can also be set equal to a list of IDs if specific objects are to be inspected.

2.3.1 GiGz ‘features’

In this section a couple of the features available when launching the GiGz as described above are presented.

- `MASTfiles=True` (default value `False`)
See Section 1.3.1
- `clobber=True` (default value `False`)
See Section 1.3.1
- `GiGfile='filename.txt'` (default value `None`)
Provide a file containing the output from a GiG inspection (see Section 1.5) and any identified emission lines where wavelengths were provided in the GiG inspection, will be marked when plotting the 1D spectra with the GiGz when `Show GiG lines` is set. This is illustrated by the black filled circles in the bottom panel of Figure 7. Note that by default (if `objlist=None`) the GiGz will only inspect objects with emission lines marked in the GiG file if a GiG output file is provided. Otherwise use `GiGfile = None` or change the default value of the `GiGselection` keyword. If a list of ids is provided to `objlist` any GiG-based selection is ignored.
- `GiGselection='selectionstring'` (default value `'emissionlineobjects'`)
Determines the selection of objects from the GiG output file provided with the `GiGfile` keyword which are inspected (will be ignored if `objlist != None`). The options available are the default option and `'all'` and `'allentries'`. The latter two options will return all the IDs found in the `GiGfile`. To inspect a more specific sub-sample of objects use the `objlist` keyword.

- **latexplabel=True** (default value `False`)
When setting this keyword the plotting labels will be rendered with LATEX. This requires an installed LATEX compiler which is compatible with python.
- **clobber=True** (default value `False`)
See Section 1.3.1
- **ds9xpa=True** (default value `False`)
See Section 1.3.1
- **openfitsauto=True** (default value `False`)
See Section 1.3.1
- **inGUIimage='G102stack'** (default value '`zfit`')
See Section 1.3.1
- **check4duplicates=True** (default value `False`)
See Section 1.3.1
- **outputcheck=False** (default value `False`)
See Section 1.3.1
- **skipempty=True** (default value `False`)
See Section 1.3.1
- **autosaveplot=True** (default value `False`)
The GiGz interface allows you to save the interactive plot to a pdf by the push of a button. To automatically save the interactive plot to such a file, when advancing to the next object, set `autosaveplot=True`. Note that any pre-existing files will be overwritten. To save these, rename them or move them to a different location before running the GiGz with `autosaveplot=True`.

2.4 Opening png and fits Files With The GiGz

When running the GiGz the png files for each object will automatically be opened when navigating between the objects similar to when running the GiG. An example of the default 1D.png and 2D.png for a single grism and a single PA are shown and explained in Figure 3. The fits files can also be opened from GiGz if desired. These are shown and explained in Figure 4.

2.5 Inspecting Redshift Fits with GiGz

The GiGz main interface shown in Figure 6 allows the user to rate the quality of the automatic redshift fits for each individual object. This is done with the colored sliders for the two PAs, which rate the individual fits from 0 to 4 in quality. Above the control buttons a guideline for the redshift inspections is provided. A rule of thumb is that:

- 0) No automatic redshift exists or the redshift fits were uninformative
- 1) The automatic redshift fit failed miserably and is of no use
- 2) The redshift is possible, but it's hard to tell how trustworthy it is
- 3) The obtained redshift is probably the correct redshift. However, there is still some uncertainty regarding the line(s) or continuum feature(s) used to constrain the redshift.
- 4) A secure redshift, obtained from multiple lines, clear continuum features, marginally resolved doublets etc.

The quality of each redshift fit is ideally assigned by including knowledge about the photometric redshift prior (the blue points and curves in the in-GUI plot), the appearance of emission lines, and the estimated continuum after contamination subtraction. These quantities are stored in the GiGz output when advancing to the next object.

Furthermore, it allows the user to assign a redshift value to each individual object using the ‘By-hand’ redshift controls marked in yellow in the GiGz window shown in Figure 6. It is this redshift which will be written to the output GiGz file. If no by-hand redshift is provided a redshift of -99 will be written to the output catalog.

In cases where only a single line is detected, it can be hard to determine what lines are observed (given that the photometry has a broad $p(z)$). In this case a keyword like ‘Multiple Redshift Solutions’ becomes useful. Multiple redshift solutions can also occur with emission line doublets, which are not, or only marginally resolved, due to the intermediate resolution of the HST grisms, and therefore cannot be told apart from single emission lines.

For good redshift solutions, the GiGz interface allows the user to indicate the presence of a few of the main galaxy emission lines, like [OIII] and H α .

2.6 The GiGz Output file

The GiGz outputs an ascii file similar to what GiG produces with a short header containing a time stamp, the inspector name, and the column names. The first column gives the object ID and the second column gives the P.A. of the reduction products inspected. The remaining columns indicate the quality of the redshift fits (0-4) and whether any obvious emission lines were identified as well as a by-hand ‘best redshift’. Each row is trailed by the input to the comment field. Hence the GiGz output file has the following format:

```
# Results from Visual Inspection of zfits initiated on 2015-10-21 16:29:00
# Inspector: John Doe
# ID PA G102_zfit_quality G141_zfit_quality MgII_detection OII_detection empty1 empty2
OIII_detection Ha_detection byhandredshift byhandredshift_quality multiple_redshift_solutions
00011 020 0 0 0 1 -1 -1 1 1 1.457 4 0 #C#
00011 280 0 0 0 1 -1 -1 1 1 1.457 4 0 #C#
00018 020 2 2 0 0 -1 -1 0 0 -99 0 0 #C#
00018 280 2 2 0 0 -1 -1 0 0 -99 0 0 #C#
01308 020 1 1 0 1 -1 -1 1 0 1.855 4 0 #C# This is a comment
01308 280 1 1 0 1 -1 -1 1 0 1.855 4 0 #C#
.
.
.
```

2.7 Select Objects Based on the GiGz Output

Similar to the GiG output, the output from a GiGz inspection is a simple ascii as described above. Hence, this file can also be used to select objects based on the provided keywords as exemplified in the following.

2.7.1 A GiGz Selection: Example 1

To select objects with [OIII] emission lines do the following in Python (to select lines in a given grism use the redshift as a constraint as well):

```
GiGzoutput = '/GiGz/output/file/to/read/from.txt'
data = np.genfromtxt(GiGzoutput, comments='#', skip_header=2, names=True)
selection = data[(data['OIII_detection'] == 1)]
print '\nThe IDs of objects with [OIII] emission:\n', np.unique(selection['ID'])
```

2.7.2 A GiGz Selection: Example 2

To select objects that fall in the [OIII]-H α sweet spot, i.e., objects with redshifts between 0.6 and 1.5, at a fairly high confidence, do the following in Python:

```
GiGzoutput = '/GiGz/output/file/to/read/from.txt'
data = np.genfromtxt(GiGzoutput, comments='#', skip_header=2, names=True)
selection = data[(data['byhandredshift'] > 0.5) & (data['byhandredshift'] < 1.5) & (data['byhandredshift_quality'] == 4)]
print '\nThe IDs of objects with [OIII] emission:\n', np.unique(selection['ID'])
```

2.8 GiGz FAQ

In this section you’ll find answers to some of the frequently asked questions about the GiGz (if they are not already answered in the GiG FAQ in Section 1.7). Feel free to email [Kasper](#) at any time, if you are not helped by the FAQs.

2.8.1 Should I always give a by-hand-redshift?

Not necessarily. If no by-hand-redshift is given the output file will just say -99 as illustrated in Section 2.6. However, the by-hand-redshift will represent the ‘best redshift’ in most cases, where the results from the grism redshift fits to each of the 1D spectra are not necessarily representative of the best redshift for the object.

2.8.2 Why does it take forever to advance to the next object (and plot the 1D curves)

Moving from one object to the next can be significantly slowed down if the 1D curves are plotted rendering the text with L^AT_EX (using `latexplotlabel=True`). Also redrawing the plots when finding the redshift for an individual object will be significantly slower when `latexplotlabel=True`. To make the plotting and moving between objects faster use `latexplotlabel=False`. To save the nicer looking L^AT_EX plots for an object post-inspection, one can re-run the GiGz (with a new output file!) providing the obtained redshift and adjusting the plot.

3 Appendix

3.1 Examples of contamination

In Figures 8-22 a sample of objects with different degrees of contamination are shown for reference. The individual figures are sorted according to increasing degree of contamination.

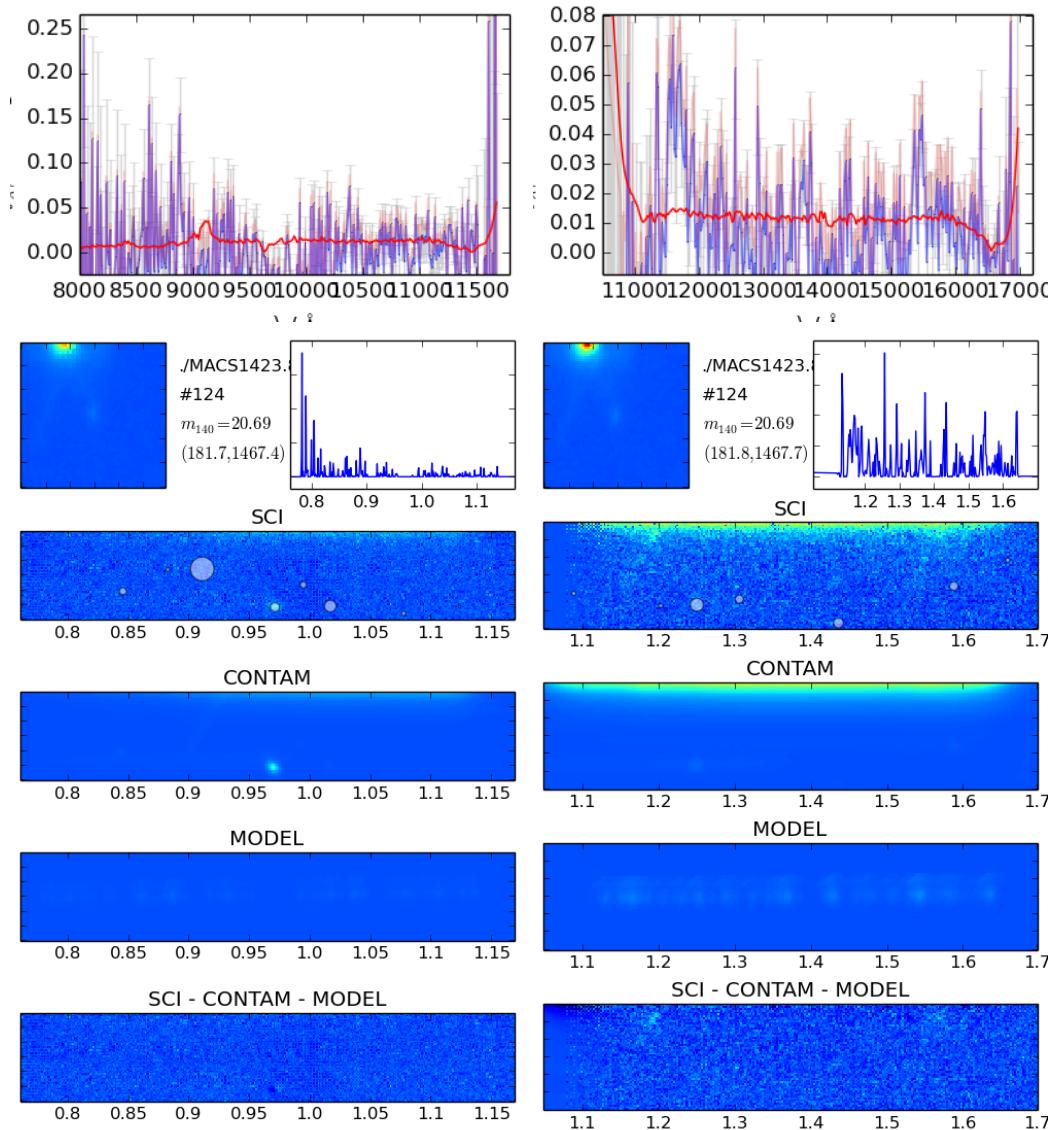


Figure 8: Example of contamination level **MILD**

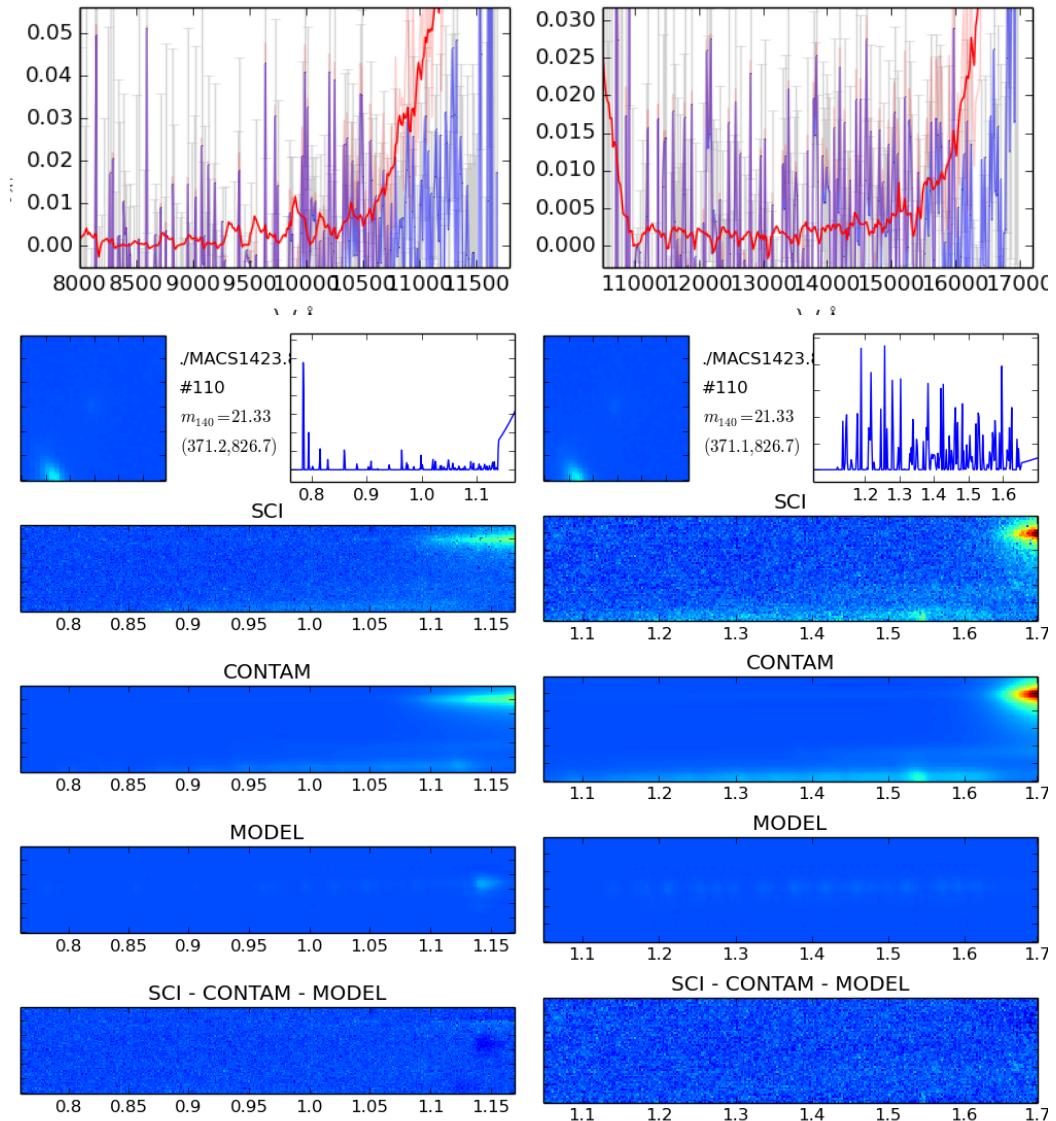


Figure 9: Example of contamination level **MILD**

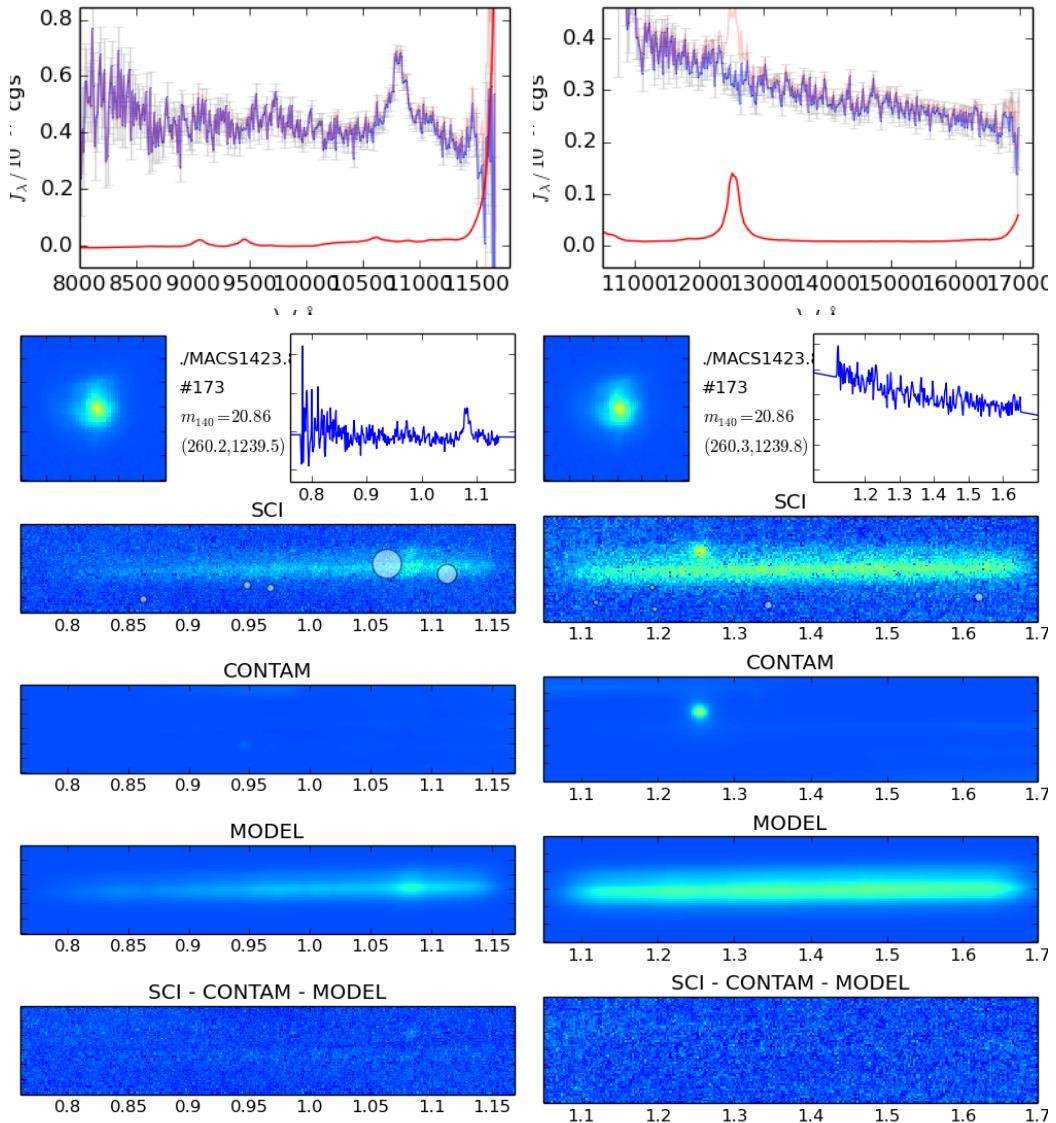


Figure 10: Example of contamination level **MILD**

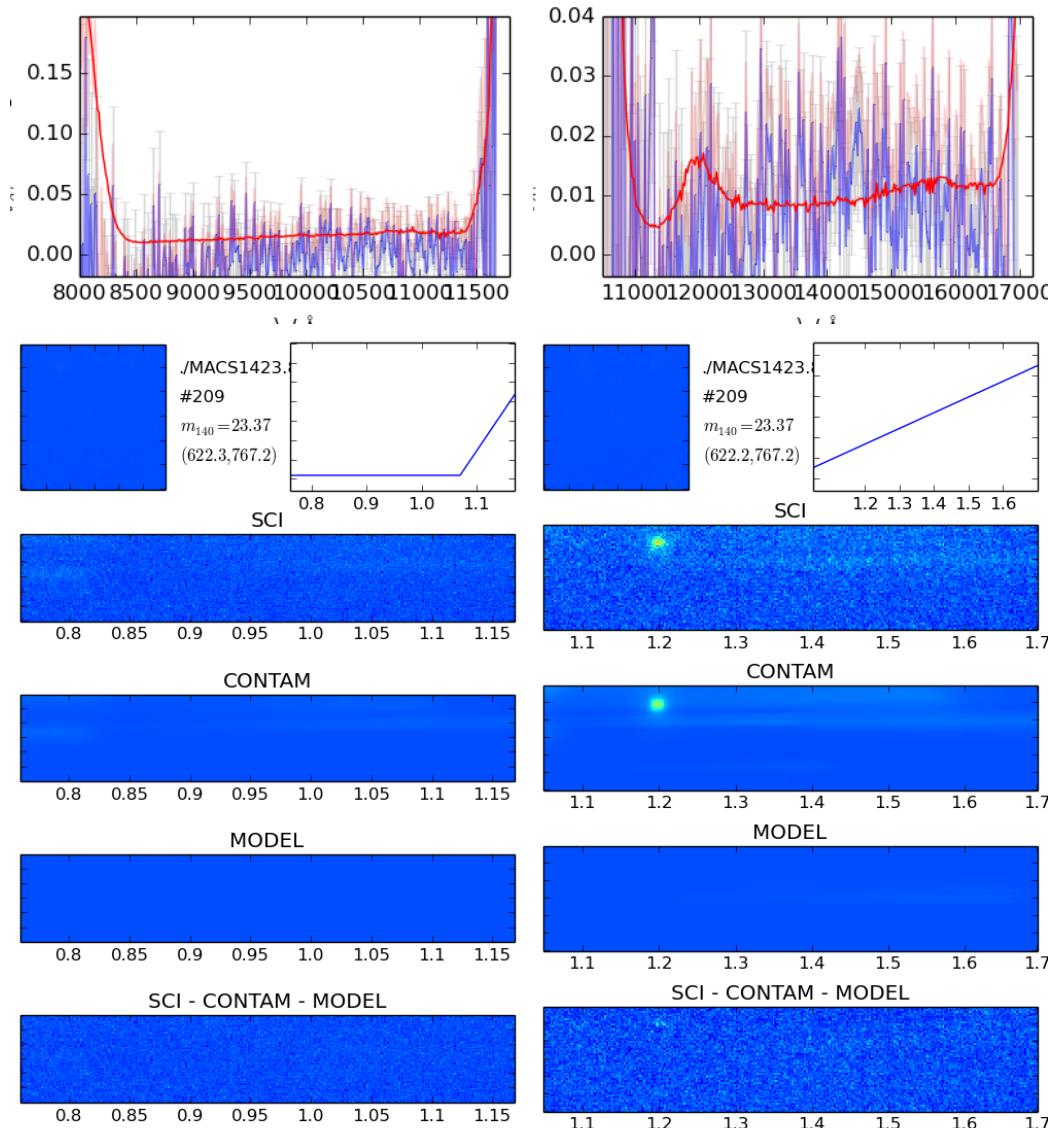


Figure 11: Example of contamination level **MILD**

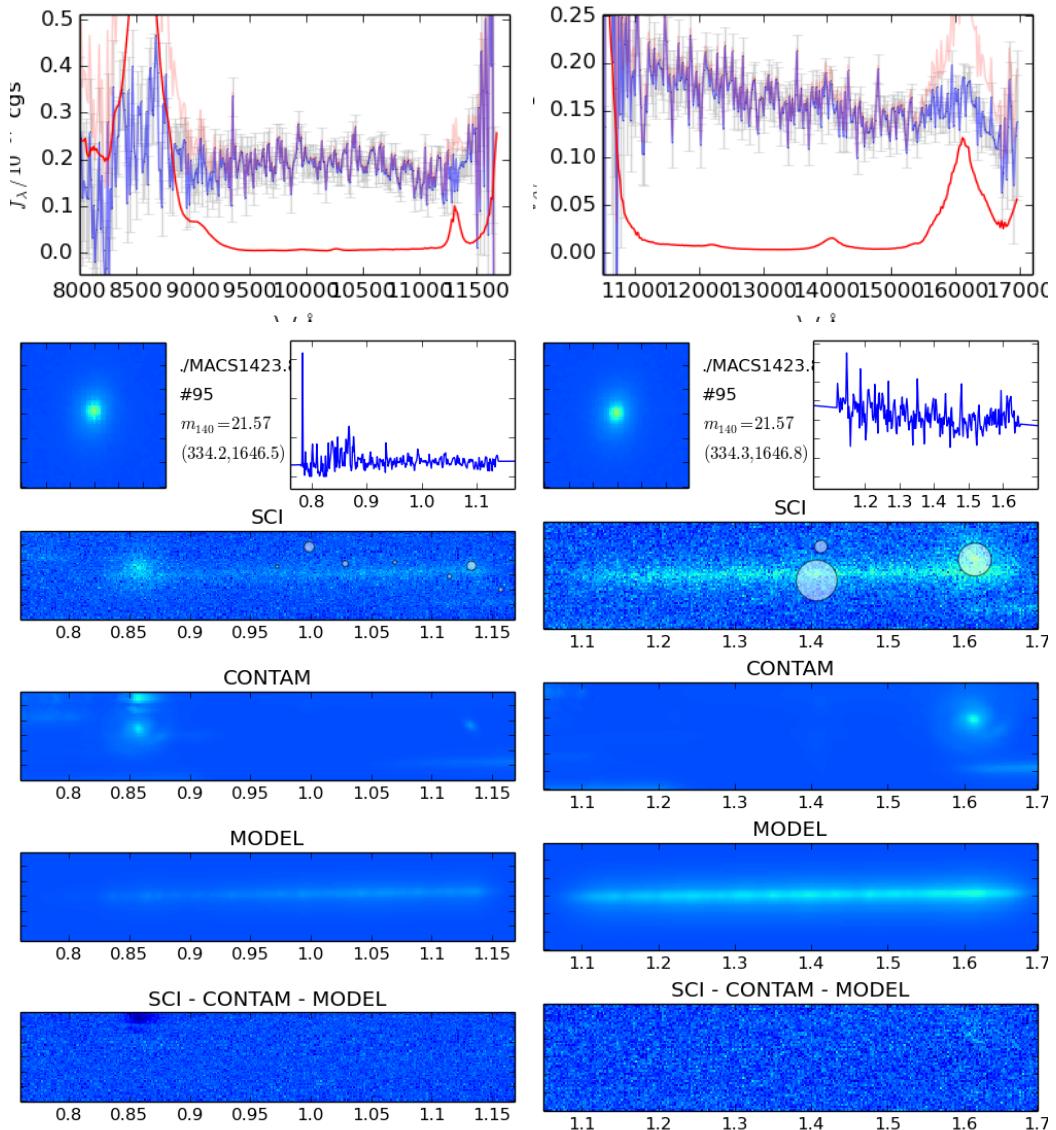


Figure 12: Example of contamination level **Moderate**

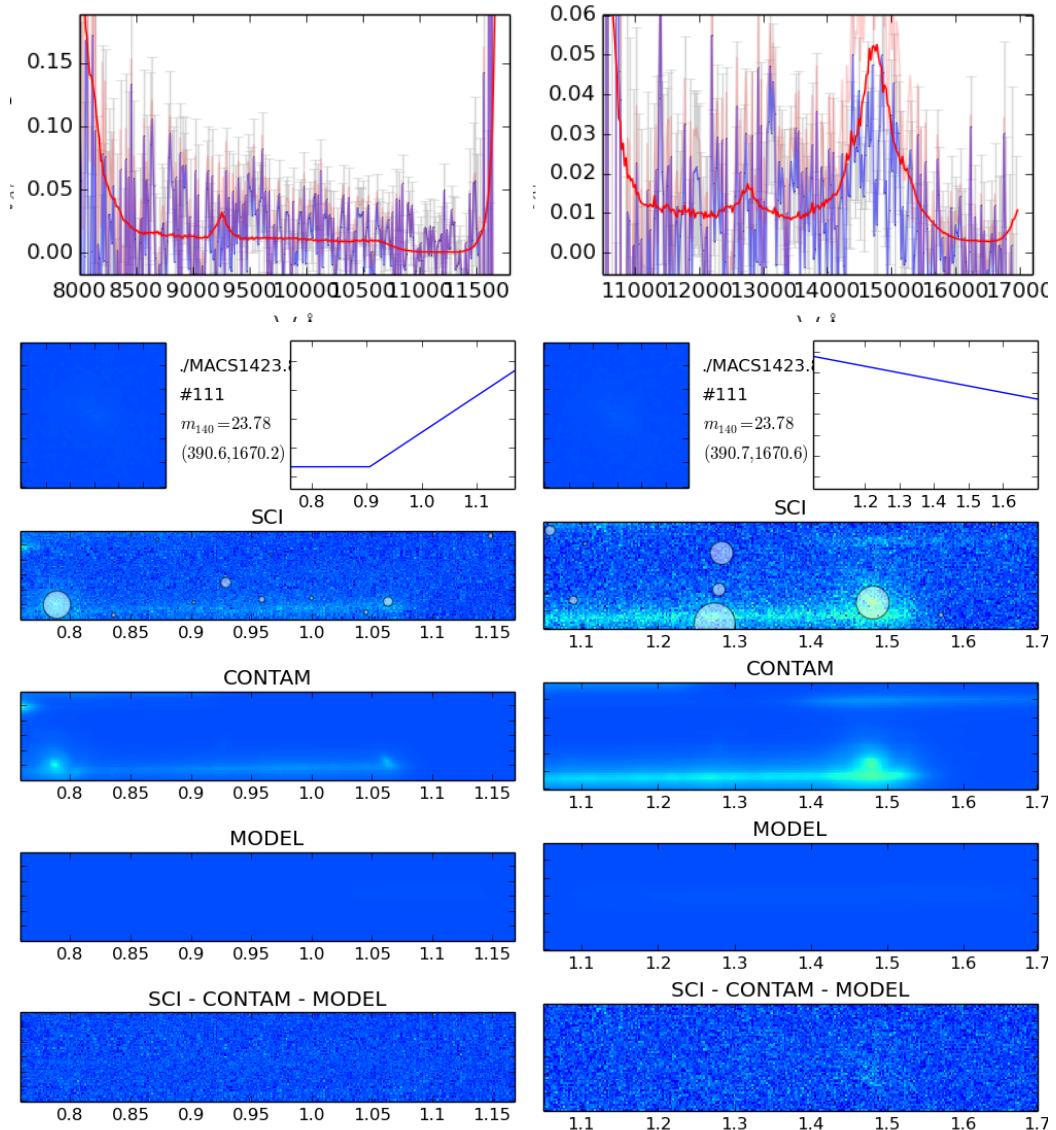


Figure 13: Example of contamination level **MODERATE**

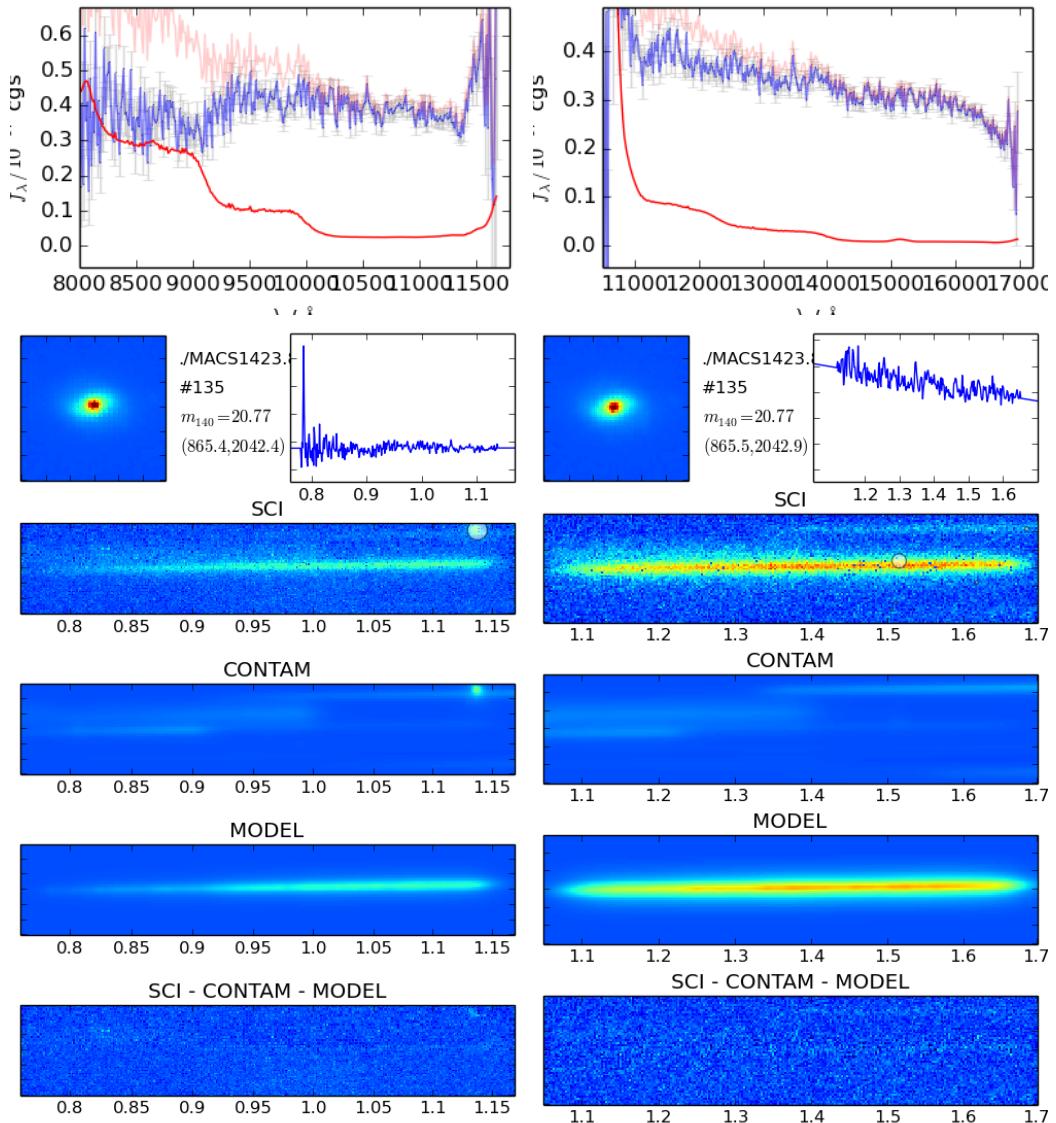


Figure 14: Example of contamination level **MODERATE**

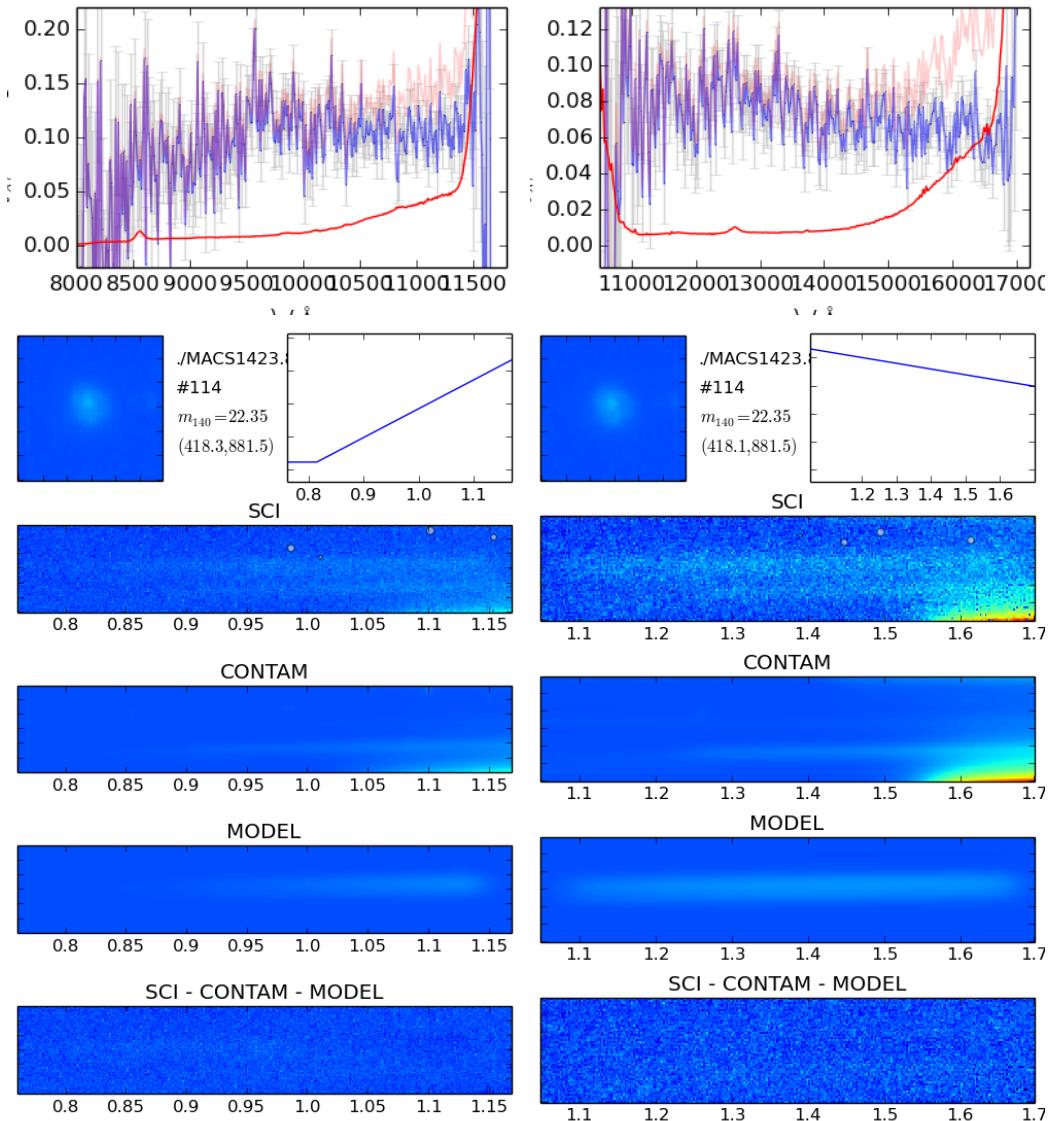


Figure 15: Example of contamination level **MODERATE**

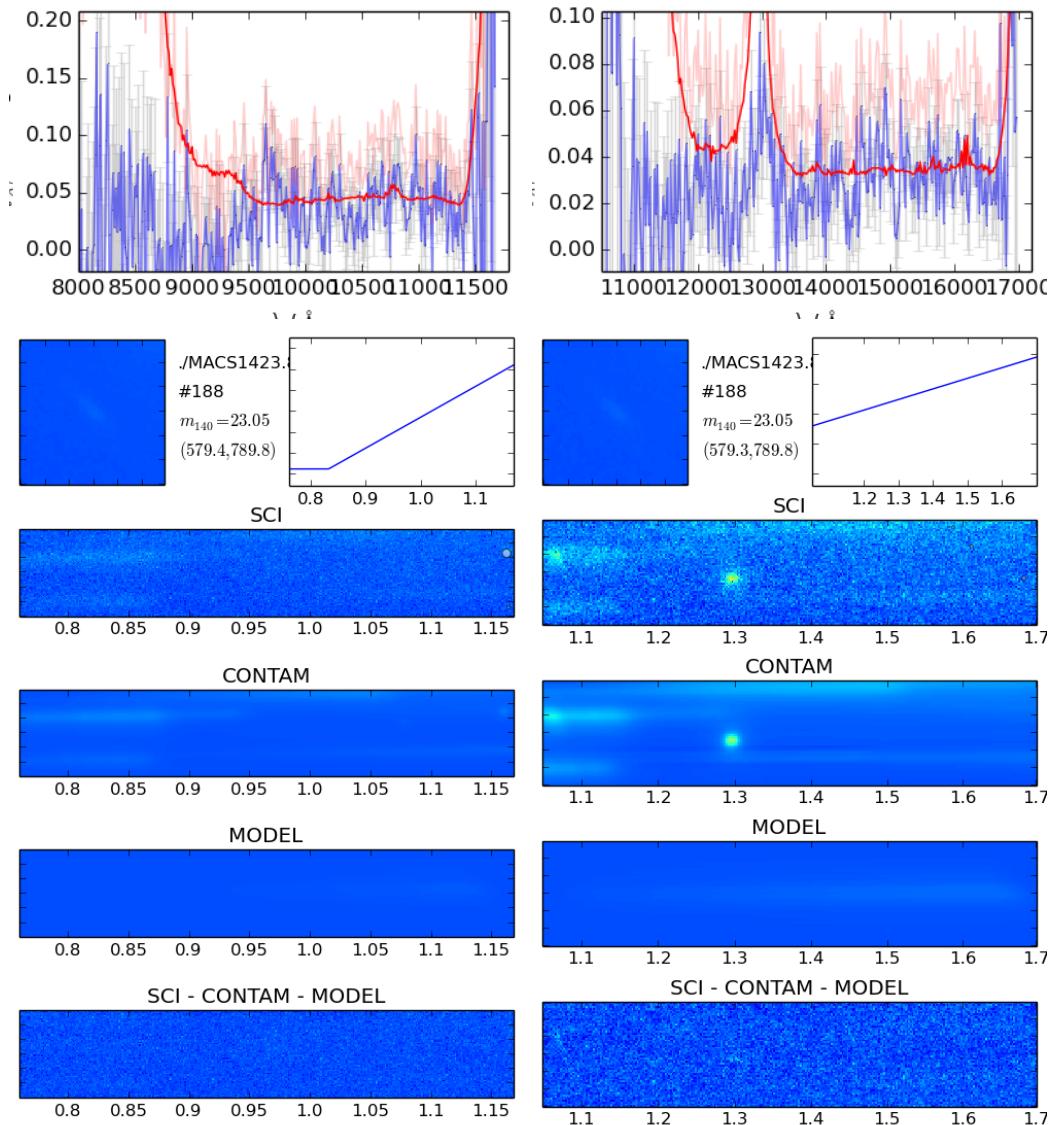


Figure 16: Example of contamination level **MODERATE**

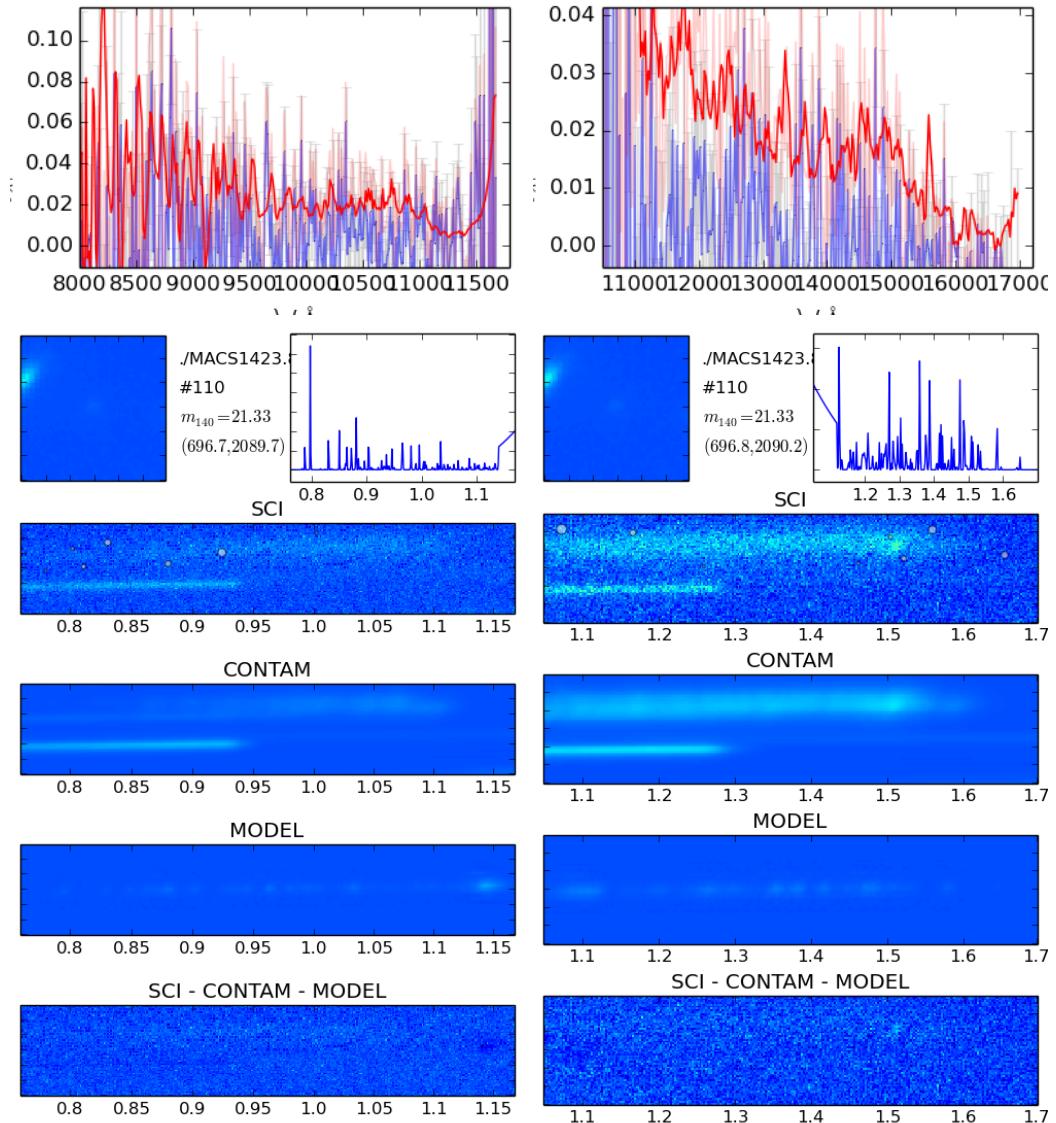


Figure 17: Example of contamination level **MODERATE**

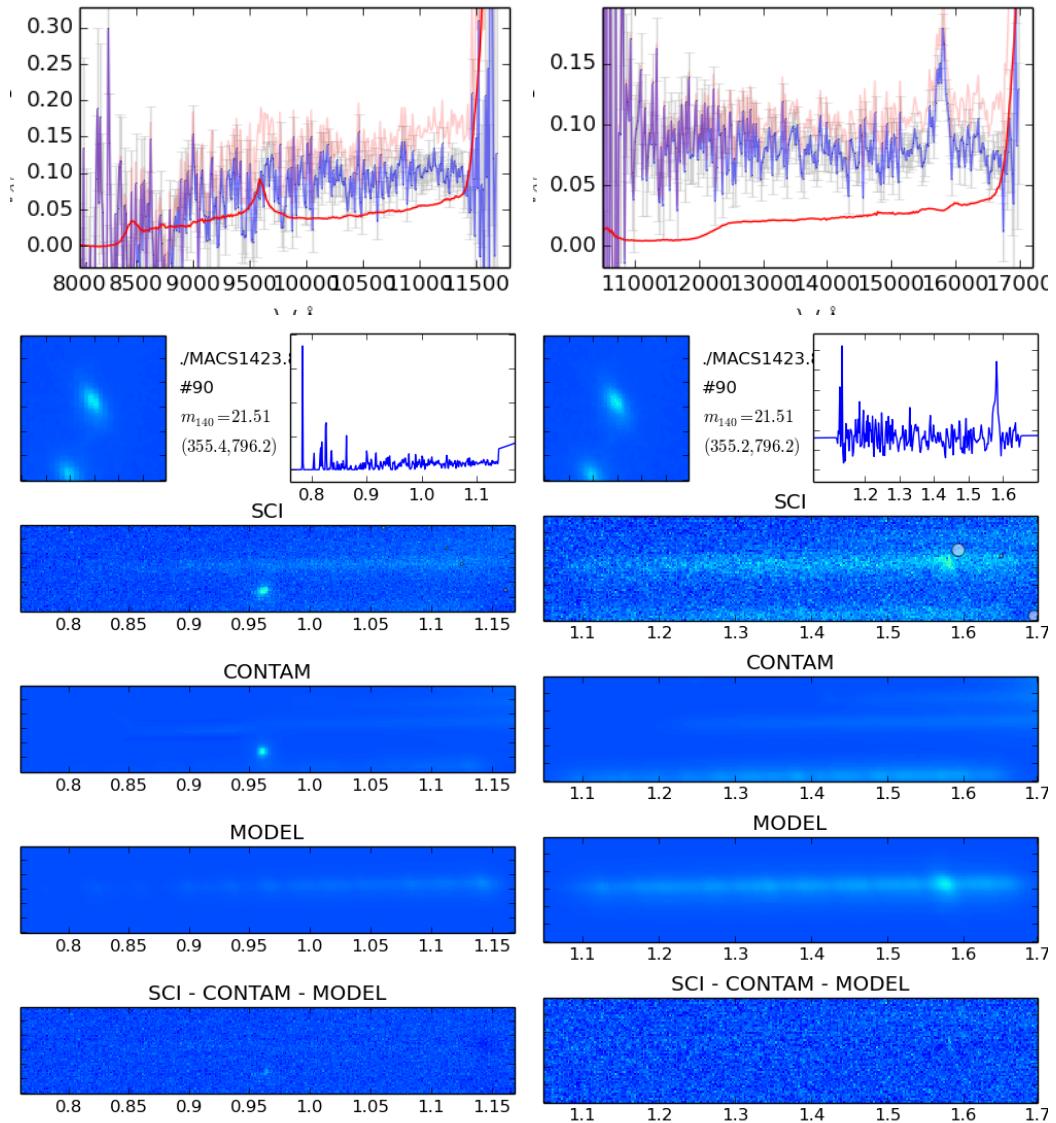


Figure 18: Example of contamination level **SEVERE**

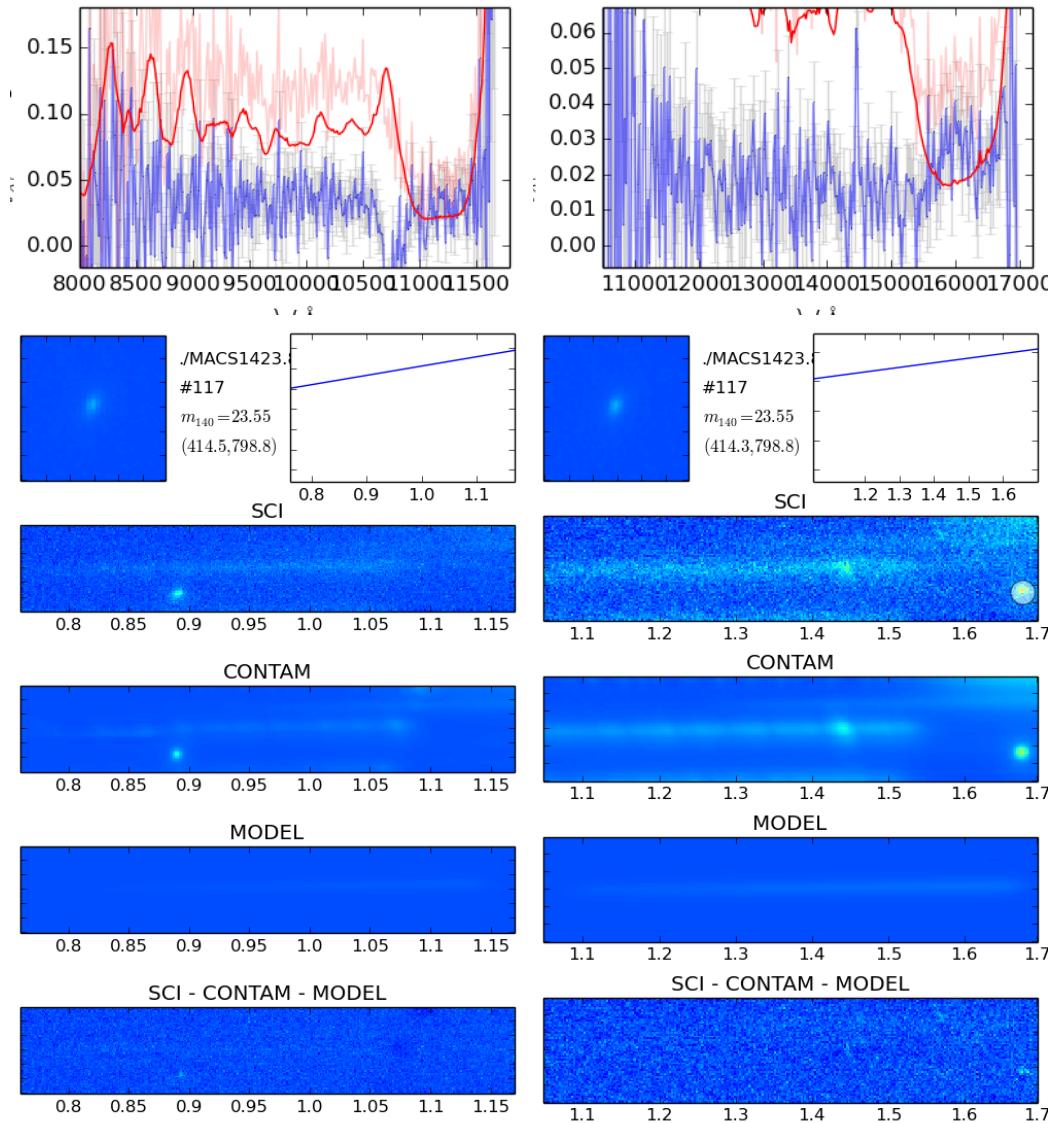


Figure 19: Example of contamination level **SEVERE**

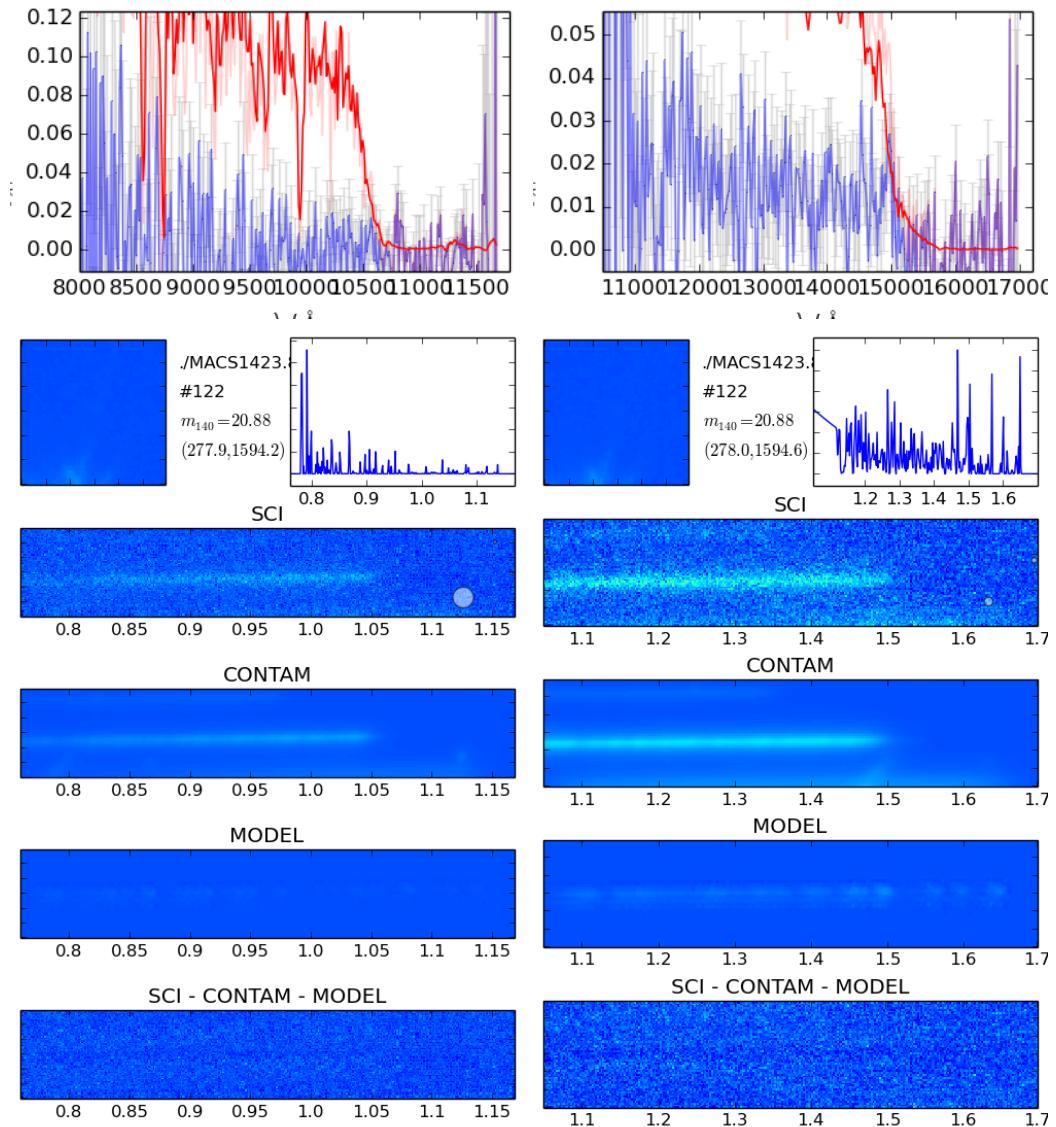


Figure 20: Example of contamination level **SEVERE**

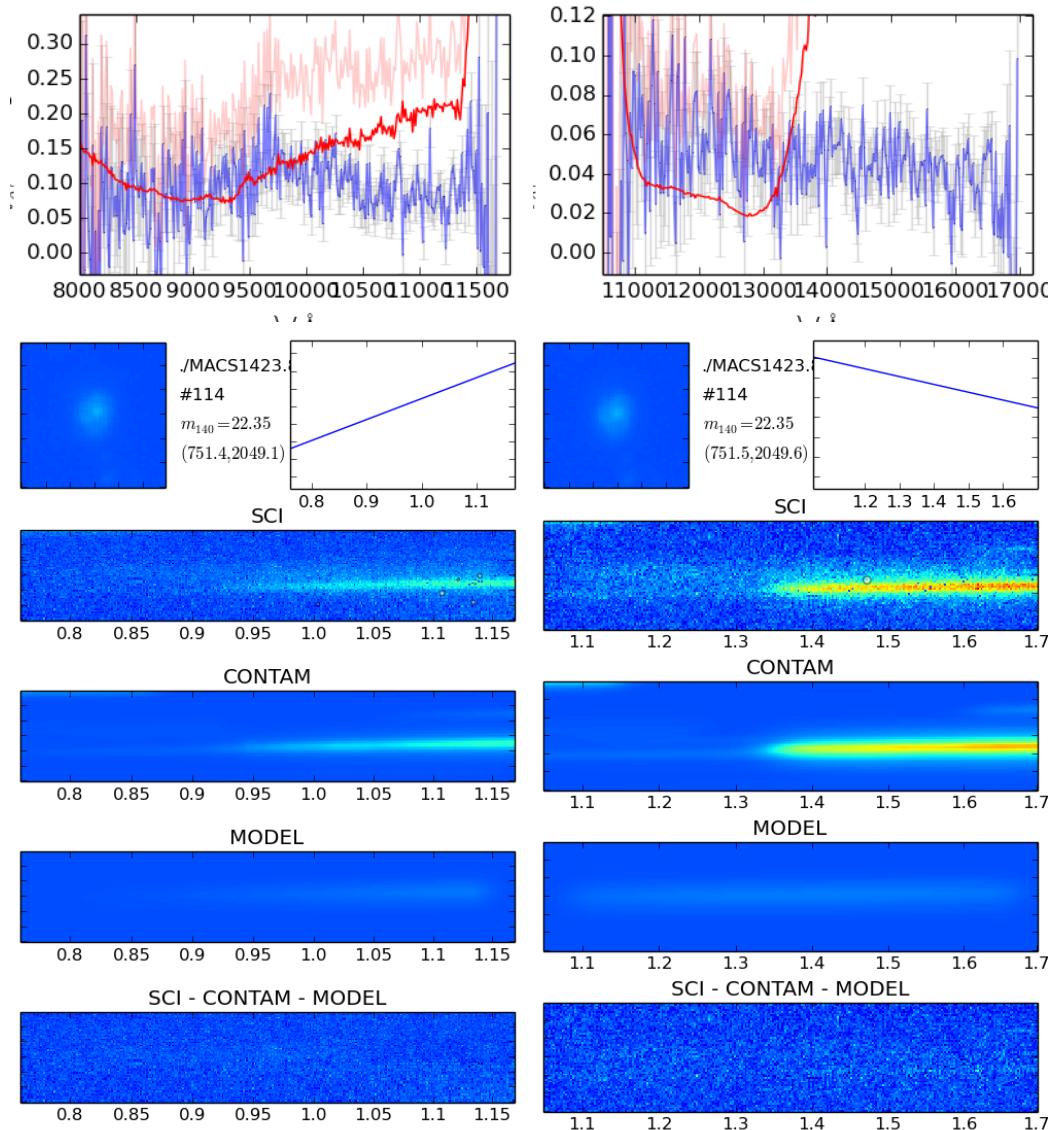


Figure 21: Example of contamination level **SEVERE**

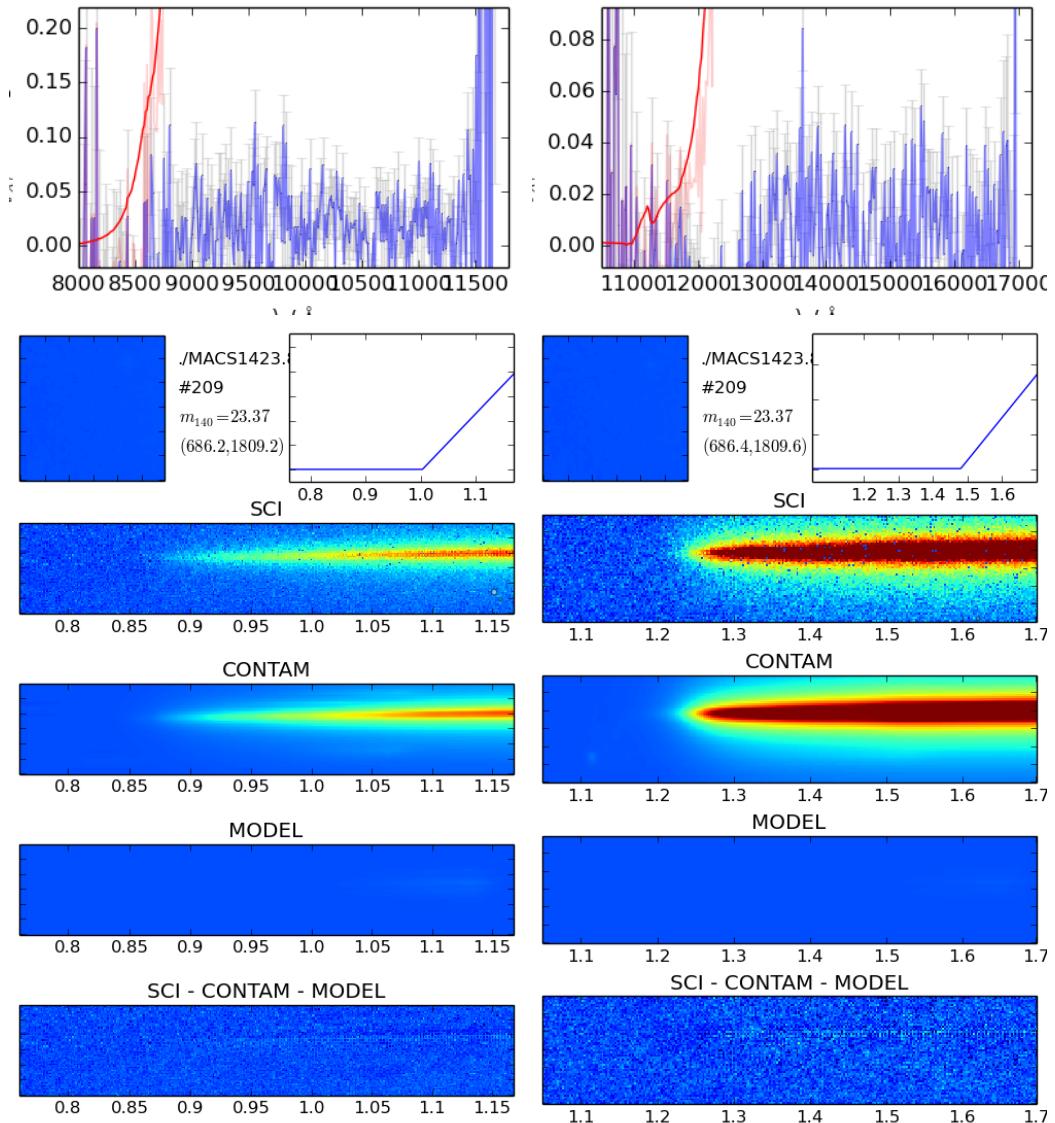


Figure 22: Example of contamination level **SEVERE**