

Porting a Multi-Object Spectrograph Mask Design App from IDL to Python

Eric Jeschke,¹ Russell Kackley,¹ Makoto Tanaka,¹ and Minkyong Kim

¹*Subaru Telescope, National Astronomical Observatory of Japan*

Abstract. We present a case-study in migrating a mask design software application for a Multi-Object Spectrograph (MOIRCS, operating at Subaru Telescope) from an IDL (Interactive Data Language) implementation to a pure Python one. The port accomplished several goals, including: (1) freeing users from onerous licensing restrictions, and (2) improving the overall stability and responsiveness of the program, and (3) improving the prospects for future upgrades and maintenance. We explain the purpose and use of the program, describe the practical procedures used in the port, and finally show and describe the user interfaces of both the old and the new versions.

1. Introduction

MOIRCS (Multi-Object Infrared Camera and Spectrograph) provides imaging and low-resolution spectroscopy from 0.9-2.5 microns over a 4x7 arcmin field of view, operating on the Subaru Telescope at the summit of Maunakea, Hawaii. A popular feature of this instrument is the MOS mode, in which a custom-cut mask can be used to obtain spectra for multiple objects within the same field. MOIRCS slit masks consist of a number of slits and alignment holes cut into a thin, flexible mask. Several masks can be loaded into the instrument prior to observation and then inserted into the optical path during observation. The alignment holes are used to line up with stars of sufficient magnitude and an alignment procedure during observation shifts and rotates the focal plane so that stars appear in the holes and the slits are aligned precisely with the objects for which they are configured. After that, a series of exposures can be taken to obtain the necessary spectra.

When the instrument was commissioned in 2006 (Ichikawa et al. 2006) an IDL-based program was developed to allow potential users to create the mask design files necessary for the observatory to cut the needed masks prior to observation. The output of the mask design program is a simple, tabular file with coordinates and lengths measured in millimeters for the mask cutting process.

2. How the IDL version works

The IDL program works by loading a FITS image large enough to cover the MOIRCS field of view. The image needs to have a valid world coordinate system and should be rescaled to 0.117 arc seconds per pixel. The program provides a graphical overlay showing the MOIRCS detector outline and overall field of view, as shown in Figure 1 (left). The user then chooses a grism (and optionally can change some of the grism

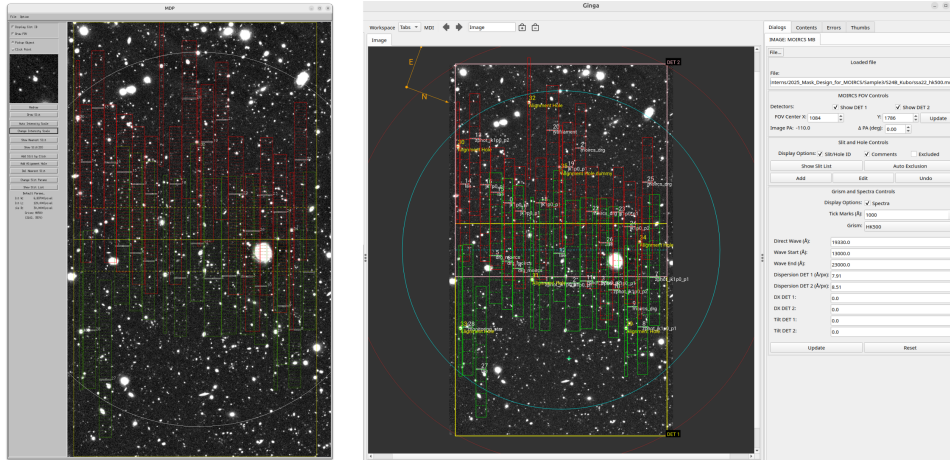


Figure 1. Screenshot showing the IDL version on the left, and the new MOIRCS Mask Builder plugin running in Ginga on the right.

parameters). Following this, the user places slits and holes within the frame and can configure the slit lengths and widths as needed. These objects can be placed by clicking on the image or by measuring the center of an object in the image. This configuration can be saved in a “.mdp” (MOIRCS Design Program) file, which is a simple ASCII CSV-formatted file containing the types, positions and attributes of the slits and holes in pixel units. An MDP file can be loaded to initialize the program if the user needs to make subsequent adjustments.

The spectral dispersion of the slits and holes on the detectors can be visualized with an option, allowing one to see if there are overlaps or other issues that could cause contamination of the spectra. The user adds as many holes and slits as necessary and adjusts their locations and parameters till they are satisfied that the design is reasonable. Finally, the user concludes the process by a final save of the the MDP file and also exporting an output SBR file (with the “.sbr” extension) that contains the physical coordinates and measurements used by the mask cutting machine.

3. Issues with the IDL Version

The IDL version of the mask builder program has several issues that were problematic for our use case:

- IDL itself has licensing restrictions, which makes it difficult to configure which computers can run the software. The license server is inside the observatory firewall, so the user either has to be logged into the IDL host, or run a special “virtual machine” that includes an IDL run-time with the application code and access to the license. However, due to export control policies, a registration and verification process is necessary on the vendor’s website(SOFTWARE 2025) to download the virtual machine software. Between the license fee, licensing restrictions and the virtual machine registration issues the whole process is onerous compared to open-source solutions.

- The virtual machine running the IDL program was notoriously unstable and would crash very easily without any clear idea of the cause. Needless to say, this did not impart a sense of confidence in the software.
- Fewer and fewer staff are knowledgeable in IDL application development and the original author of the software has largely moved on to other responsibilities.

4. Porting from IDL to Python

Given the issues described above, we were interested in porting the application from IDL to Python. Python has good support for astronomical applications with good base frameworks like *astropy* or *skyfield*, and a number of GUI frameworks to choose from.

We ascertained that the Python version of the application would need to (1) Load and display a FITS file, and adjust contrast or set cut levels on the image for good visibility of the overlays; (2) Draw overlays on the image to show the field of view, detector positions, slit and hole positions with labels, spectral dispersion, and (3) provide a UI with controls for loading and saving MDP files, adjusting parameters for slit position, width, etc. and exporting the mask cutting SBR file.

Subaru Telescope uses the *Ginga* package(Jeschke et al. 2012), an open-source toolkit for building astronomical viewers in Python, in a number of different applications. This viewer already provides many of the basic features needed for this program including good support for viewing FITS files and drawing graphical overlays. During the summer of 2025, our intern Minkyong Kim took on the task of turning this IDL program into a Ginga plugin within an approximate 7 week time span. It is worth noting that Kim had no knowledge of astronomy or spectroscopy before starting the project, but did have a good understanding of Python programming.

The basic procedure for guiding Minkyong toward the goal was as follows. First, we explained in general terms what the IDL program should be doing and provided a copy of the IDL code. Second, We gave her a few different example sets of FITS files with input (mask design) and output (mask cutting) files. Third, we gave her time to experiment with the IDL program and understand the basic UI and how that worked. After this initial introduction, we introduced Ginga, showing the features that covered much of the needed functionality and explaining how the plugin architecture works (and showing a few examples of plugins). We guided her to get a skeleton of a new plugin working with a very simple UI and then answered questions as needed as she iterated toward a solution.

After the seven week internship, the MOIRCS Mask Builder plugin was fully functional. Our staff continued to polish and do mild refactoring on the program for an additional few weeks to make the operation a bit more intuitive and maintainable, as well as adding some additional features.

5. Using the MOIRCS Mask Builder Plugin

Using the MOIRCS Mask Builder is fairly straightforward coming from the old workflow. After starting the plugin, all files can be loaded from the plugin UI, shown in the right side of Figure 1. The “File” menu at the top of the UI allows loading and saving of various kinds of files, while the panels below that allow the user to select which

overlays are shown, add and edit holes and slits, select a grism, or adjust the default parameters for the grism.

An object (slit or hole) can be added by clicking precisely within the image, outlining a box around an object to find the center of it, or typing directly in the position in either pixels or celestial coordinates.

6. Advantages of the MOIRCS Mask Builder Plugin

The new plugin offers a number of desirable features over the older IDL program:

- Ginga is open-source and licensed under the generous and unrestrictive BSD license that the rest of the Python ecosystem is based on, so there are no restrictions on where it can be installed and used.
- The instability of the old program is eliminated; Ginga is an extremely stable program that has been in continuous nightly use at the observatory since 2012.
- The plugin allows performant zooming and panning, and allows us to provide a position angle feature so the user can type in a position angle to see the spectral dispersion with various rotations vs. the focal plane.
- The plugin correctly handles overplotting on different pixel scales, so the user should no longer have to provide an image pixel-scaled to an exact value (although we still need to verify that the mask cutting file is generated correctly with alternate scales).

7. Getting and Installing the MOIRCS Mask Builder Plugin

To use the MOIRCS Mask Builder Plugin you need to first install Ginga and then one additional package. We recommend using a virtual python environment such as *virtualenv*, *conda* or similar to install and run this plugin. An example installation is shown below assuming a conda-type system:

```
$ conda env create -n moircs_mdp python=3.13
$ conda activate moircs_mdp
$ pip install ginga[qt6]
$ pip install git+https://github.com/naojsoft/naojutils
```

After installing this way you can run the MOIRCS MBP by simply typing “ginga” on the command line inside your conda environment, and then opening the plugin from the “Plugins => Subaru => Planning => MOIRCS Mask Builder”.

References

- Ichikawa, T., Suzuki, R., Tokoku, C., Uchimoto, Y., Konishi, M., Yoshikawa, T., Yamada, T., Tanaka, I., Omata, K., & Nishimura, T. 2006, Proceedings of SPIE - The International Society for Optical Engineering, 6269
- Jeschke, E., Inagaki, T., & Kackley, R. 2012, in Software and Cyberinfrastructure for Astronomy II, edited by N. Radziwill, & G. Chiozzi (SPIE). Vol. 8451
- SOFTWARE, N. G. 2025, The IDL Virtual Machine, <https://www.nv5geospatialsoftware.com/Support/Maintenance-Detail/the-idl-virtual-machine>. Accessed on October 22, 2025