Draft version November 2, 2018
Typeset using LaTeX **manuscript** style in AASTeX62

# 1. INTRODUCTION

This manual is written for the reorganization-pyraf branch of the pipeline. For the original manual, see the master branch. This branch requires a separate environment than the other branches. The pipeline needs to use `pyraf` (which only supports `python2.7`) at this stage of development. The environment file `SAMOS_specs.txt` provides the correct environment for this branch of the code.

The main content of the directories the user should have in his or her working directory for the current version of the pipeline is shown below:

- `SAMOS-working-dir/`:
    - `cleaner.sh`
    - `Docs/`
    - Readme.md
    - `LDSS3/`
    - `helper_files/`
    - `linelists/`
    - `py_mods/`
- `SAMOS-working-dir/py_mods/`:
    - `CreateFuelStructure.py`
    - `CreateInput.py`
    - `CreateSlitStructure.py`
    - `FlatNorm.py`
    - `InitializeSAMOS.py`
    - `NormDivFlats.py`

    * `OutlineSlits.py`
    * `Overscan.py`
    * `OverscanAndTrim.py`
    * `SAMOSHelpers.py`
    * `SlitCutout.py`
    * `SlitID.py`
    * `__init__.py`

- `SAMOS-working-dir/helper_files`:
    * LMask1.SMF
    * LMask2.SMF
    * LMask1_ycoords_c1.txt
    * LMask2_ycoords_c1.txt
    * LMask2_ycoords_red_c1.txt

This document reviews the purpose and describes how to use these programs to reduce the test data from LDSS3. Eventually, it will be used to reduce multi-object spectroscopy data taken with the SOAR Adaptive-Module Optical Spectrograph (SAMOS). For information on the instrument, see (Robberto et al. 2016). In order to avoid reinventing the wheel, this pipeline is heavily based on the Flame DRP written by Belli, Contursi, and Davies (2017). Flame is written in IDL, while the SAMOS pipeline will ultimately be written in Python3, but this version uses Python2.

## 2. INITIALIZATION

The pipeline is made to run in an ipython session. Upon starting a new session, import the initialization script with '`from InitializeSAMOS import initialize_SAMOS`'. This module is responsible for reading the headers from the sample data in **LDSS3/2017-11-30/**, and organizing the information into data structures for use by the rest of the code.

The initialization script takes in two arguments, the name of the directory with the test data (date on which data was retrieved) and the mask for which the user wants to use. The initialization then returns the main data structure of the pipeline, defined by the variable name '*fuel*'. The call for this step should look like:

```
In [1]: from InitializeSAMOS import initialize_SAMOS
In [2]: fuel = initialize_SAMOS(datadir,mask)
```

After running and reading over the terminal output, you should have a new directory named **LMask1** and a new file named **LMask1.db**. This step also now keeps track of the field image. The FITS headers for the test LDSS3 data do not contain the slit positions, so having the image of the actual slit mask is necessary at this time. The slit mask information is contained in the files LMask1.SMF and LMask2.SMF. The explanation for these files can be found on the Carnegie Observatories website, here.

## 3. `OVERSCANANDTRIM`

Next, we have to determine the overscan and trim the edges of the fits data. This code reads the database file and creates arrays which separate the data into flats, comparison lamps, and science data.

The main part of `OverscanAndTrim` is the calling of the module `Overscan.py`. This module parses the fits data array into regions of bias and target data by reading the fits headers BIASSEC and DATASEC respectively. There is a function in `Overscan.py` called `FieldTrim(...)` that trims the field mask to the correct size. Then, I looked in DS9 for the top edges of each of the slits and made a text file of the positions which will be used in the step `OutlineSlits`. The function responsible for trimming and bias subtraction of the other FITS files is `Overscan(...)`.

The bias is due to a combination of the camera noise from the readout process and electric "pre-charge" on a CCD chip by the electronics. The BIASSEC header gives the region of overscan, which contains information for this bias. An array of this overscan region is made by grabbing the rows and columns from the data array (`d = f[0].data.astype(''f'')` → `overscan = d[biassec rows,`

`biassec columns]`). The function then takes the median (or mean) of the overscan regions along the rows (axis=1), and subtracts these values from each data value in their respective data rows. The main procedure is shown in the figure below.

$$\begin{pmatrix} cropped & and \\ bias & subtraced \\ data & matrix \end{pmatrix} = \begin{pmatrix} d11 & d12 & d13 & ... & d1m \\ d21 & d22 & ... & ... & d2m \\ ... & ... & ... & ... & ... \\ dn1 & ... & ... & ... & dnm \end{pmatrix} - \begin{pmatrix} [b1] \\ [b2] \\ [b3] \\ [...] \\ [bn] \end{pmatrix}$$

With the new data matrix, the function makes a new FITS file for the files originally entered as input. The output FITS are placed in their respective LMask directories and the 'ccd' part of their original file names are replaced with LMask(1,2).

To run this step, type

```
> ./OverscanAndtrim LMask(1,2).db
```

## 4. NORMDIVFLATS

Once we have our cropped and bias subtracted fits files, we need to locate and combine the flat frames into one master flat, which will be divided from the science frames. First we make a stack of the flat field frames and scale them by their median. then the stack is median combined and normalized, which gives the pixel-to-pixel variation in detector sensitivity. The function outputs a fits file named LMask(1,2)master_flat. Finally, the science frames are divided by the master flat and written out to new frames, which are placed in a directory named **flat_fielded**.

To call this routine, use

```
> ./NormDivFlats LMask(1,2).db
```

The function also creates thumbnail images of the output data frames.

## 5. OUTLINESLITS

Since the headers for the test data do not contain slit information, I typed up a text file of slit positions called **LMask(1,2)_ycoords_c1.txt**. This file contains the top edge pixel for each of the slits in the field image. The module `Slit_id.py` contains the function `get_edges(...)`, which is responsible for tracing the slits, using the master flat output from `NormDivFlats` as a template. `get_edges` takes the master flat and the slit positions text file as input. The function steps along cutouts of the FITS data frame based on the y-axis positions given in the text. For each step, the median of the cutout along the x-axis is taken, and then the derivative along the y-axis is computed to locate the transition from the chip to the slid edge. This steps along the slit until it reaches the end, storing the x and y positions. The arrays of slit positions are written to a region file called **reg_txt.reg** within the corresponding mask directory, and can be loaded into DS9. An example of this for LMask2 is shown in figure 1. The slits outlined here are representative of the slits found in the image of the mask field, so not all of the slits in a science image will be included in this step.

To run this step, use

```
> ./OutlineSlits LMask(1,2).db
```

## 6. TL;DR

This is the most up to date version of the documentation for the SAMOS pipeline thus far. Currently, there is no main script to run all the processes at once, so the user will have to run them step by step. To summarize, the user input for data reduction up to and including slit identification is,

```
> ./WhichFITSFiles 2017-11-30 LMask*
> ./OverscanAndTrim LMask*.db
> ./NormDivFlats LMask*.db
> ./OutlineSlits LMask*.db
```
(1)

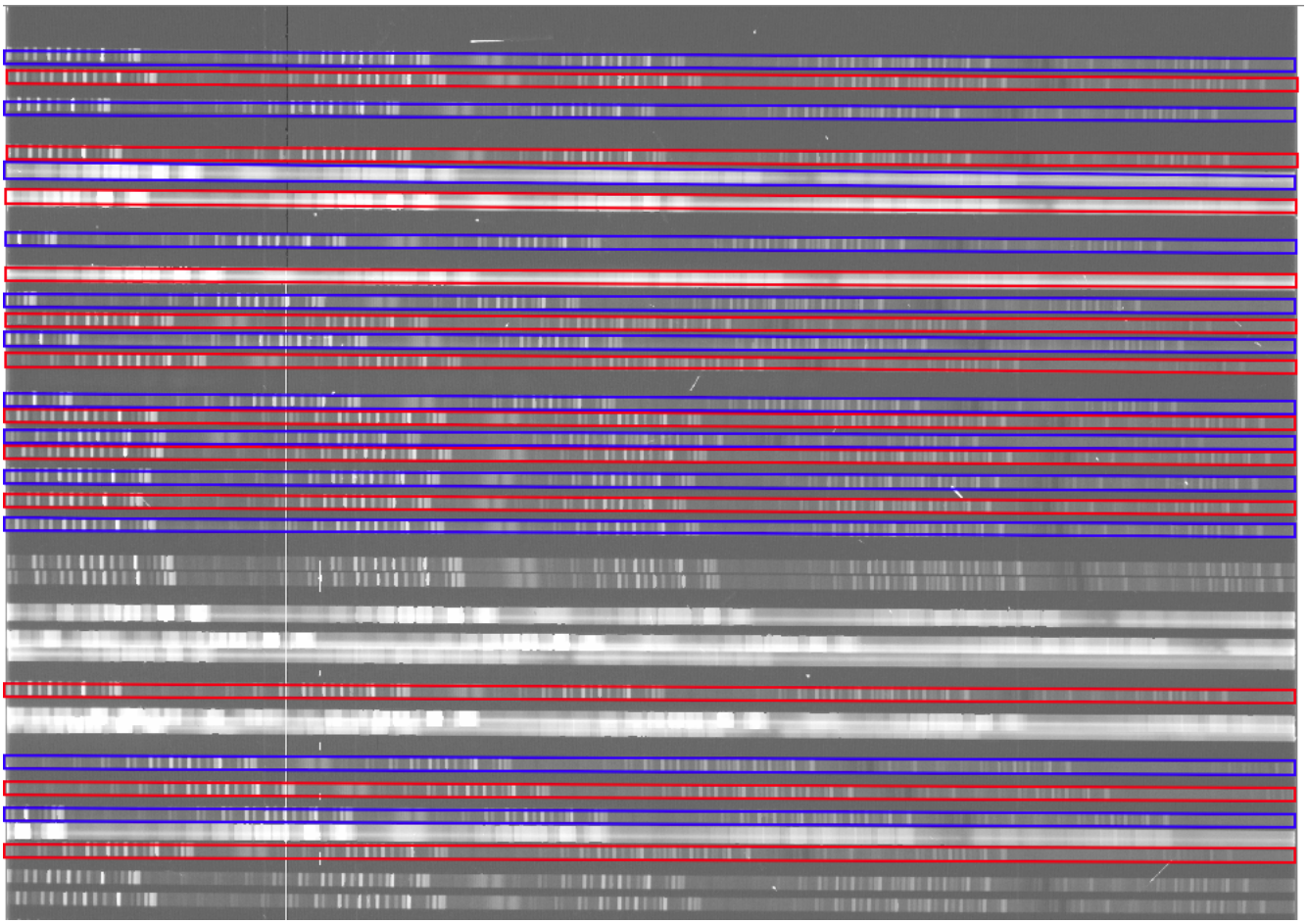

**Figure 1. helper_files/slitsLMask2.reg** overlayed onto science image in DS9.