CAMO-S GUI Documentation – May to August 2021

**Purpose**: To build a graphical user interface (GUI) that can perform spectral analysis of meteors and meteor fragments, using data collected by the spectral and direct cameras from Eigenfield and Tavistock sites. This includes image analysis and modification, calculation of model spectra, construction of libraries, and converting scripts to be Linux-compatible, among other tasks.

**Required imports for GUI functionality:**

* Western Meteor Physics Library (WMPL)
  + Github link: <https://github.com/wmpg/WesternMeteorPyLib>
* PyQt5
  + <https://pypi.org/project/PyQt5/>
* PyQtGraph (version 0.12.1, April 2016)
  + <https://pypi.org/project/pyqtgraph/0.12.1/>
* Imageio (version 2.9.0)
  + <https://pypi.org/project/imageio/>
* PyQt5 designer
  + <https://pypi.org/project/PyQt5Designer/>

**Files for GUI functionality:**

* Camo-S.ui file, open and edit with designer (working .ui file is available on github)
* Spectral .vid file - can be saved in any folder
* Direct .vid file - can be saved in any folder
* Spectral flat file (the one we’re currently using is available on github) - should be saved in the *same folder* as CAMO-S.py

**Required folders for spectral library/python driver functionality** (to construct model spectrum):

* Spectral Library (available on github)
  + Should have all necessary support files, it is advised to leave files and folders where they are. Moving files may disrupt the code, as file positions are sometimes identified relative to each other.

**CAMO-S.py overview (sequential order):**

See script for full doc string associated with each function.

1. Imports and support functions

* Standard imports from python and pyqt5
* Import functions from the wmpl library
* Cython init is there to supporting loading and applying flast later on
* Support functions and classes
  + Adjust levels: called later to adjust the levels of the image seen by the user (not the levels used to analyze the measured spectrum)
  + loadImage: load an image using imageio
  + binImage: will take a given image and return a binned image array
  + FlatStruct (class): structure containing flat field, used to load and apply the flat later on
  + loadFlat: reads in file and path name
  + applyFlat: takes output from loadFlat, applies flat to actual image

2. Primary Class - contains GUI functionality

* GUI Modifications (initialize graphics view items, buttons, etc.)
  + All *direct* *video modifications* will come first, in the order of: image view, image level histogram, mouse cursor
    - Direct file image view: “window” in which the user will see the video frames
    - Direct image histogram: histogram to the right of the image where the user can manually adjust image levels
    - Initialize direct mouse: change the cursor to a crosshair in the direct image view widget, and allow the user to get the image coordinates when they click the mouse
  + All *spectral video modifications* will come next, in the order of: image view, image level histogram, mouse cursor and region of interest, background sets, markers, and spectral flat
    - Spectral file image view: same as direct file image view, for the spectral video frames
    - Spectral image histogram: same as direct image histogram. Note that if the user adjusts the levels in the GUI, it will not affect analysis of the measured spectrum
    - Initialize spectral mouse and region of interest: similar to direct mouse, make the cursor a crosshair in the spectral image view. *Also* initialize the spectral region of interest, which will be used later to analyze a certain part of the image and measure the spectrum
    - Initialize background sets: frame numbers for set 1 and set 2 (both of which should *not* have the spectrum visible) to be summed and averaged to provide an image background. This background can be viewed by the user, and will be subtracted from the spectrum to provide a clearer measurement.
    - Spectral markers
      * Affine marker: will plot the affine transform from the direct image view so the user can confirm which spectra they want to measure, provided more than one is visible on the spectral image view (i.e. if fragments are discernable)
      * Profected affine marker: used by the GUI to project the affine marker onto the spectrum, as the affine marker is never exactly on the desired spectrum. This marker is not seen by the user, but used by the plotting function to get the scale of the plot.
    - Load spectral flat: allows user to input file and path name, followed by a function call to load the spectral flat to every spectral video frame.
* Button triggers
  + Direct file control buttons (see code for button names/associated functions)
  + Spectral file control buttons (see code for button names/associated functions)
  + Plotting buttons (see code for button names/associated functions)
* Functions to support image upload and analysis
  + Direct file control functions
    - uploadDirectVid: reads video into GUI, displays 0th frame
    - updateDirectFrames: re-sets the frame being shown, updates associated time, date, and frame number. Adjusts image levels using adjustLevels function (as seen in section 1).
    - nextDirectFrame: increase the frame number by 1
    - lastDirectFrame: decrease the frame number by 1
    - affineTransform: use affine transform file, scale plate, and plateScaleMap to run the affine transform *from* direct *to* spectral images. At present, the affine transform file is only loaded in the script: ensure the file is in the *same folder as the CAMO-S.py file*.
  + Spectral file control functions
    - uploadSpectralVid: same as uploadDirectVid, but for the spectral file
    - updateSpectralFrames: same as updateDirectFrames, but instead of using image levels it applies the spectral flat.
    - nextSpectralFrame: same as nextDirectFrame
    - lastSpectralFrame: same as lastDirectFrame
    - spectralROI: introduces a square region of interest, which can be resized/rotated as needed to fit the spectrum.
    - checkSpectralBackground:
      * Uses frame sets 1 and 2, initialized in section 1, to define a frame range, build and fill an array, find the median of the array, and create a background image. Includes flat in background calculations.
    - showSpectralBackground: shows background in a pop-up window so the user can confirm it is satisfactory. Not a necessary step.
    - checkSpectralRegion: get the array region in the spectral ROI box to define image data
    - clearSpectralROI: removes ROI box on spectral image view and clears all associated data. Resets spectral roi.
    - clearAffine: removes the affine marker associated with the affineTransform function in the direct file control functions. Re-initializes the marker so the affineTransform function can be performed again later, with new data.
    - removeSpectralFlat: Very similar to updateSpectralFrame button, but uses the adjustLevels function instead of applying the image flat. Will only remove the flat from the current frame. Background and spectrum will still be calculated with the flat applied.
  + Joint file control functions
    - nextFrame: moves both direct *and* spectral frames forward by 1 frame.
    - lastFrame: moves both direct *and* spectral frames back by 1 frame.
    - nextTimeFrame: moves to next frames that are as close in time as possible.
    - lastTimeFrame: moves to last frames that are as close in time as possible.
  + Plotting functions
    - profectAffine: projects the affine marker from its mapped location on the spectral image to a point on the spectrum
    - plotMeasuredSpec: calculates the background and the region of interest, even if the user hasn’t manually checked them. Uses scaling parameters given (2.85 nm/pixel) and projected point on spectrum to plot the spectrum in as intensity vs wavelength (nm).
    - clearSpec: clears all measured spectra shown on the graph.

**CAMO-S.py supplemental information:**

1. User sequence

Some steps must be taken before others, some are optional. A general sequence the user might follow is:

* Upload direct/spectral videos
* Upload flat
* Find frames with meteor, meteor fragments, spectrum
* Select and adjust region of interest
* Optional: check spectral background and region of interest
* Click on meteor in direct camera, run affine transform
* Plot measured spectrum

2. Current issues to resolve:

* Because the mouse click event is now used to attain the click coordinates, the image cannot be moved by clicking and dragging with the mouse. This is a PyQt5 graphics view issue, I didn’t get to looking at how to reassign click and drag. Denis should be able to provide more guidance on this.
* The timelock function for keeping frames as close together in time as possible doesn’t work for the first 99 milliseconds of each second; the function depends on the length of the string, but if there are 99 milliseconds or less the length of the string changes.
* The function from pyqt5 used to get the ROI handles gets the position within the entire scene, not the position within the spectral frame. There is another function - self.roi.getHandlePositions() - that exists in [pyqtgraph version  0.12.2](https://pyqtgraph.readthedocs.io/en/latest/graphicsItems/roi.html). I constructed the GUI with pyqtgraph version 0.12.1. This function will give the location of the handles within the image itself, which is much more preferable than what we currently have. Unfortunately I had some problems with downloading 0.12.2, but hopefully you have more luck and this will be a quick  fix.

3. Still to be completed:

* Calculation of the model spectrum using Pete Gural’s C code (in the form of the python library that runs off this code). Functions in the library are covered in another document.
  + You can connect buttons to these functions using ‘self.Button\_name.clicked.connect(functionName)’
  + Some buttons may require a ‘lock’ function, where once it is clicked it is locked until it is clicked again
  + It may be helpful to look at/run through the python driver to get  a feel for the C code functionality. All functions in the python library are also fully documented.
* Combining measured spectra from multiple frames to reduce noise in the measured spectra
* Adding reference lines for notable elements (Na, Fe, Ca)
* Calculation of extinction and trajectory from text/pickle files (Denis can provide more guidance on this)