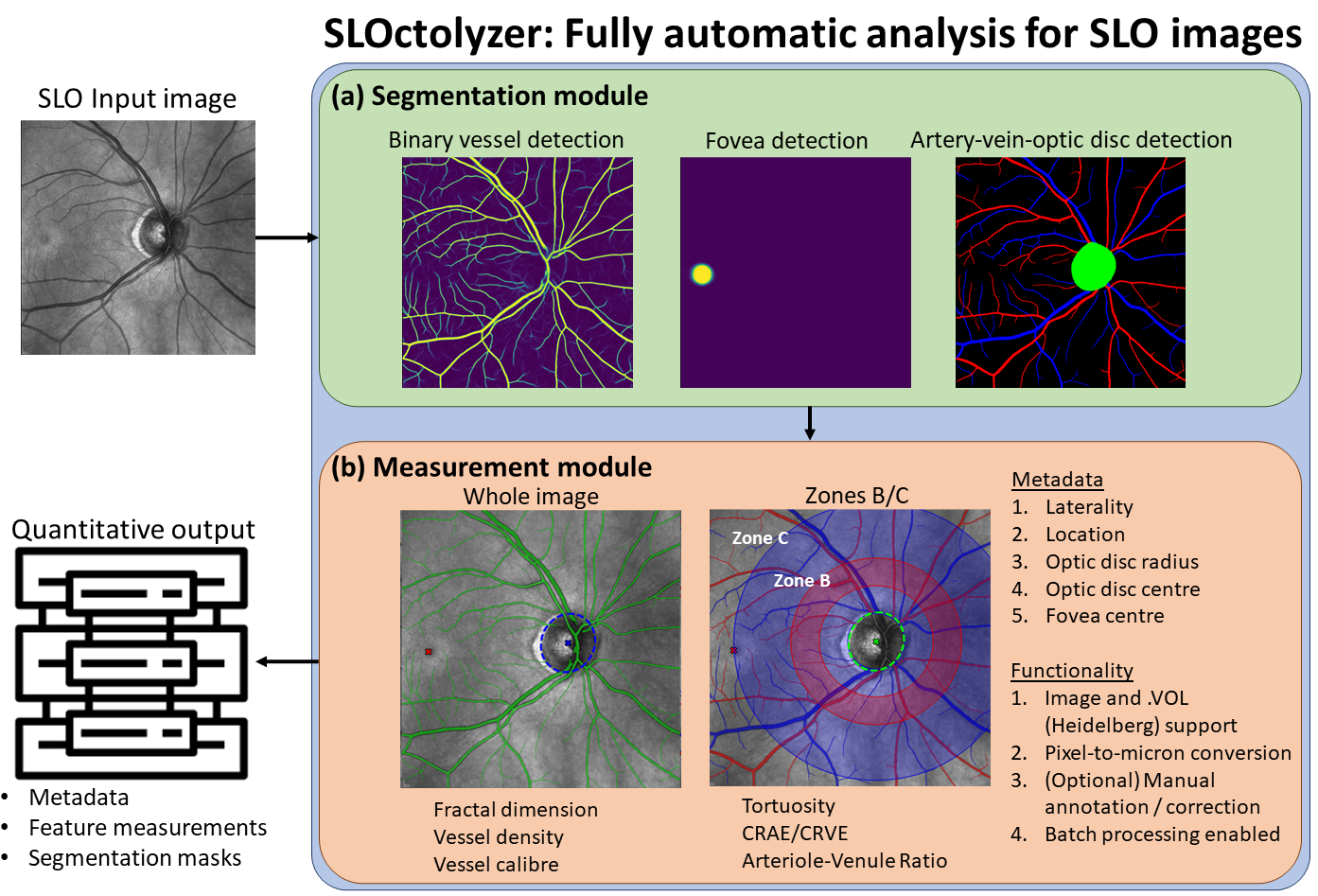
**Instructions on using SLOctolyzer: a software package for SLO image analysis**

Analysis toolkit for automatic segmentation and measurement of retinal vessels on IR-SLO images.

SLOctolyzer is a fully automatic pipeline which is capable of fully characterising the vessels, fovea and optic disc in confocal, near infra-red scanning laser ophthalmoscopy (IR-SLO) images. The pipeline utilises fully automatic deep learning methods for segmenting these landmarks, including classification of vessels in arteries and veins.

SLOctolyzer is also capable of extracting clinically-relevant features of interest of the segmented retinal vessels. The code used to measure these features is heavily based on the code produced by [Automorph](https://tvst.arvojournals.org/article.aspx?articleid=2783477), whose codebase can be found [here](https://github.com/rmaphoh/AutoMorph).

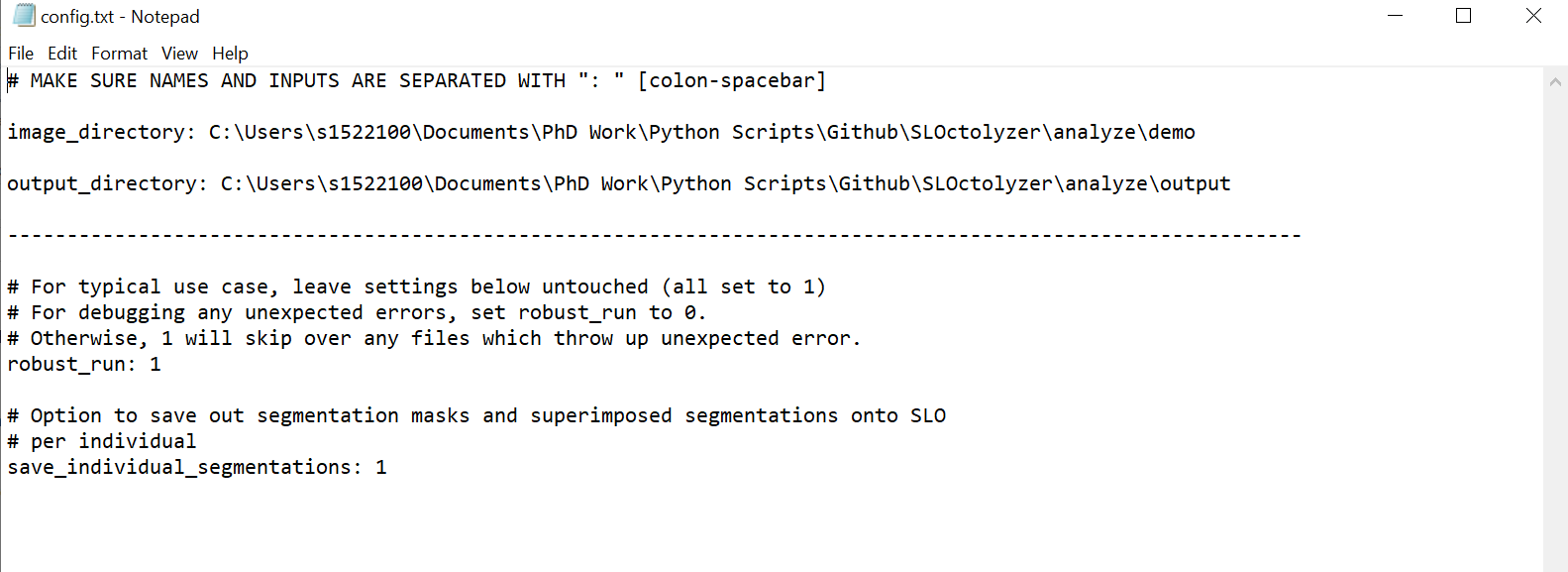
****See below for a visual description of SLOctolyzer's analysis pipeline.

**0) INSTALLATION**

1. Download/clone the SLOctolyzer directory which stores the main codebase.
2. You will need a local installation of python to run SLOctolyzer. We recommend a lightweight package management system such as Miniconda. Follow the instructions [here](https://docs.anaconda.com/free/miniconda/miniconda-install/) to download Miniconda for your desired operating system.
3. After downloading Miniconda, navigate and open the Anaconda Prompt, and individually copy and run each line found in “SLOctolyzer/instructions/install.txt” to create your own environment “slo-analysis” in Miniconda and download necessary packages.
   1. **(Optional)**: if you have a GPU running locally to use SLOctolyzer, line 3 in install.txt should be “pip3 install torch torchvision --index-url https://download.pytorch.org/whl/cu121”

**1) GETTING STARTED FROM THE TERMINAL**

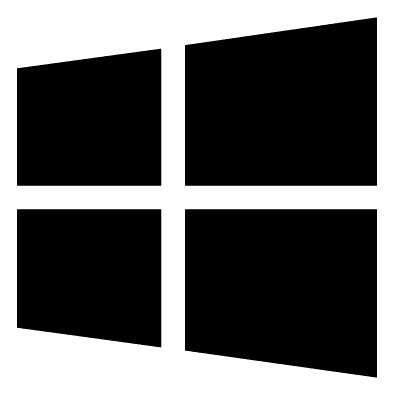
1. The configuration file config.txt is used in conjunction with running the software from the terminal (see below) and contains user-specified parameters. See the table below for each parameter and their definition.

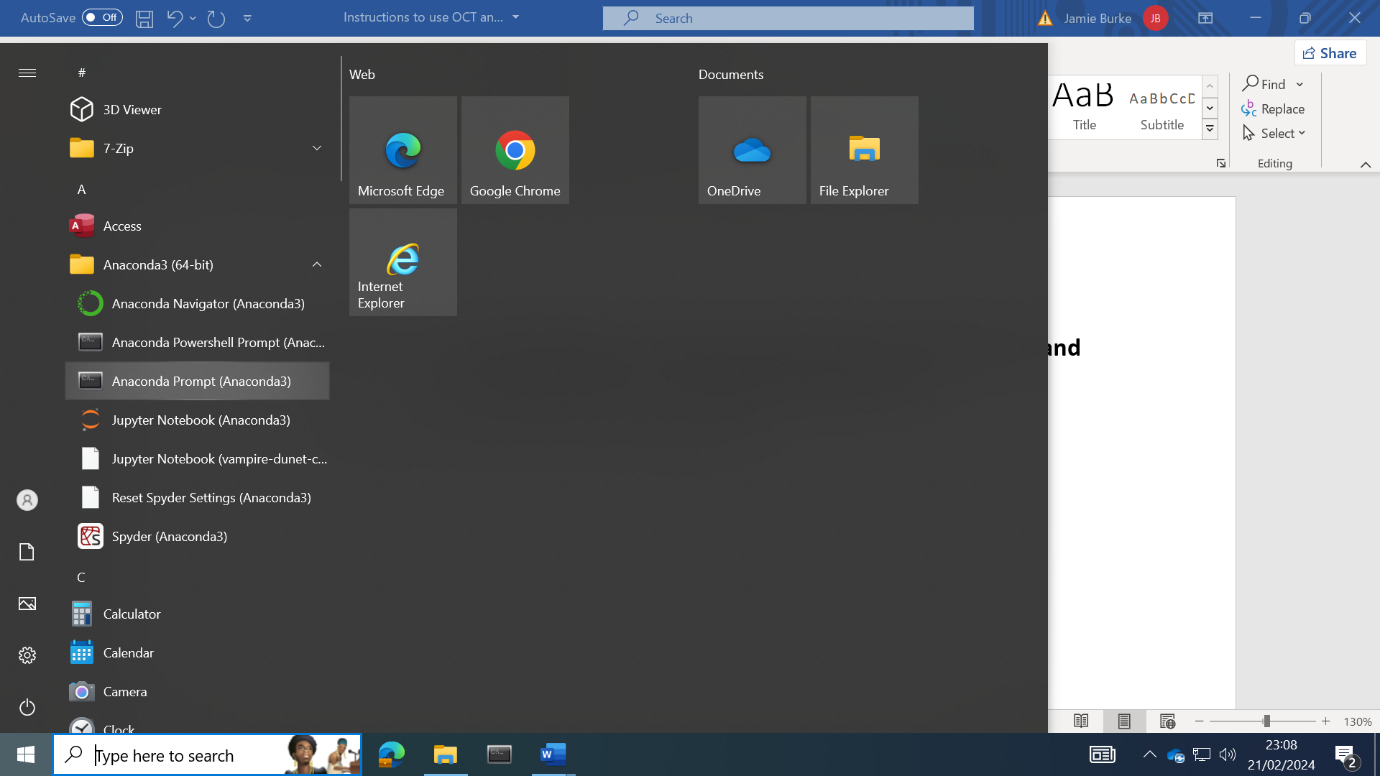


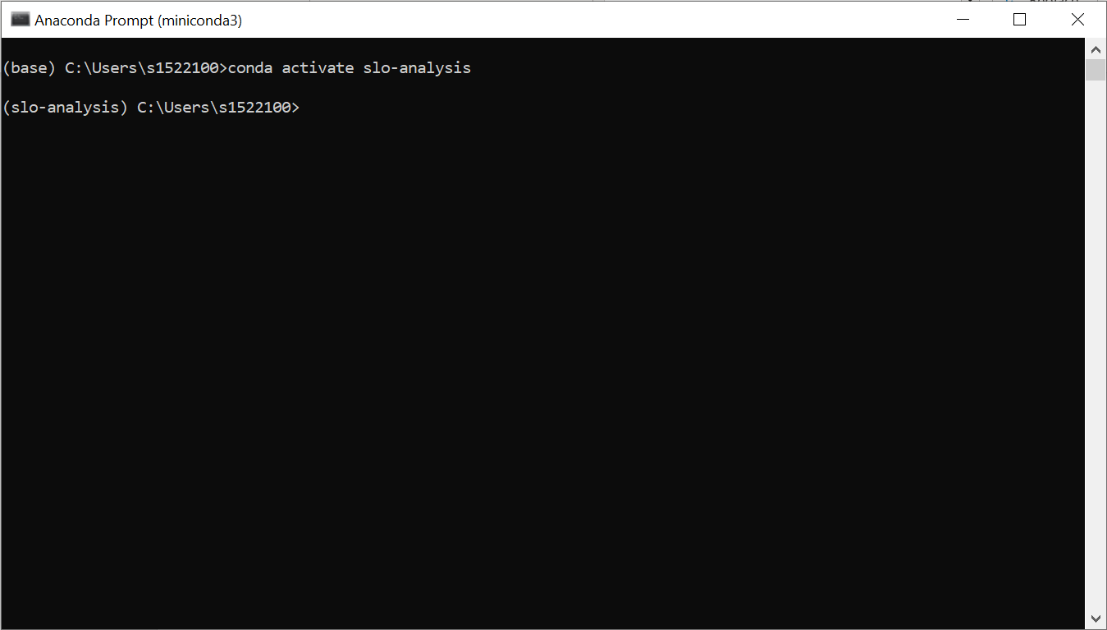
|  |  |  |
| --- | --- | --- |
| Parameter | Description | Expected value |
| image\_directory | Where SLOctolyzer will look for image files. | Any valid directory path. **Note: there should be an “: “ [colon-spacebar] before the path is written.** |
| save\_directory | Where all results will be saved. | Any valid directory path. **Note: there should be an “: “ [colon-spacebar] before the path is written.** |
| robust\_run | This will ensure the software will run without fail, skipping over any files which failed to be analysed unexpectedly. Set this to 0 to debug any problems. | 0 (No) or 1 (Yes) |
| save\_individual\_segmentations | Flag to save out the SLO image, segmentation masks, a log file and images showing the regions of interest measured.  Might not be preferred if processing large batches | 0 (No) or 1 (Yes) |

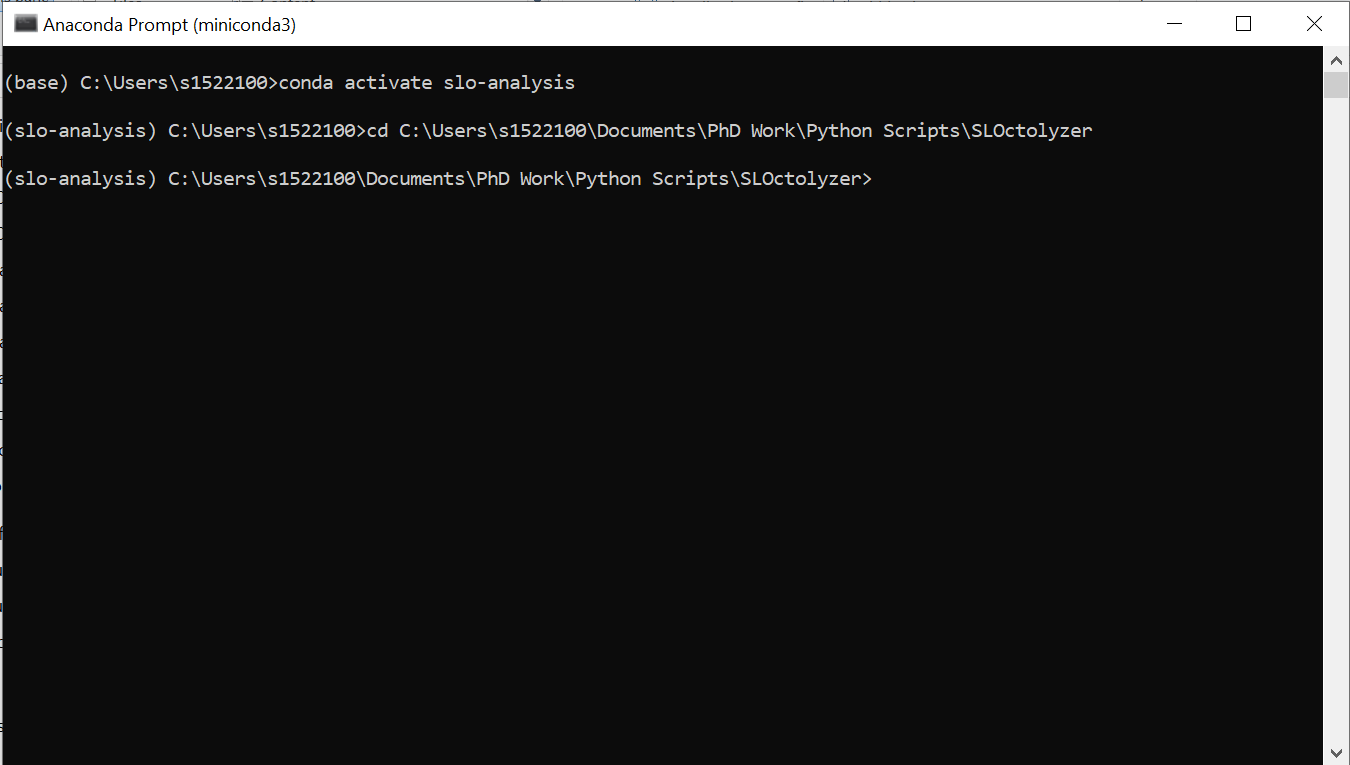
1. In SLOctolyzer\analyze\demo, there are example images you can use to test out the software and its outputs.
2. **(Optional**) If you wish to convert pixel space into physical space, fill in the information for each file placed into image\_directory in an excel document “fname\_resolution\_location\_eye.xlsx”. This template file can be found in SLOctolyzer\analyze\demo.
   1. For location, you can specify if it’s a , i.e. if its “Macula”- or “Optic disc”-centred.
   2. You can also specify the Eye type, i.e. if its “Right” or “Left”
   3. **Note:** this is not compulsory, and the images will be analysed without specifying this file. However, all measurements will be in pixels.
   4. **(To use)** copy the template .xlsx file stored in SLOctolyzer\analyze\demo and save it in your specified image\_directory and edit the rows accordingly to correspond to your own image data. There is an additional excel sheet in the file (“key”) describing what should be inputted for each column in the template file.

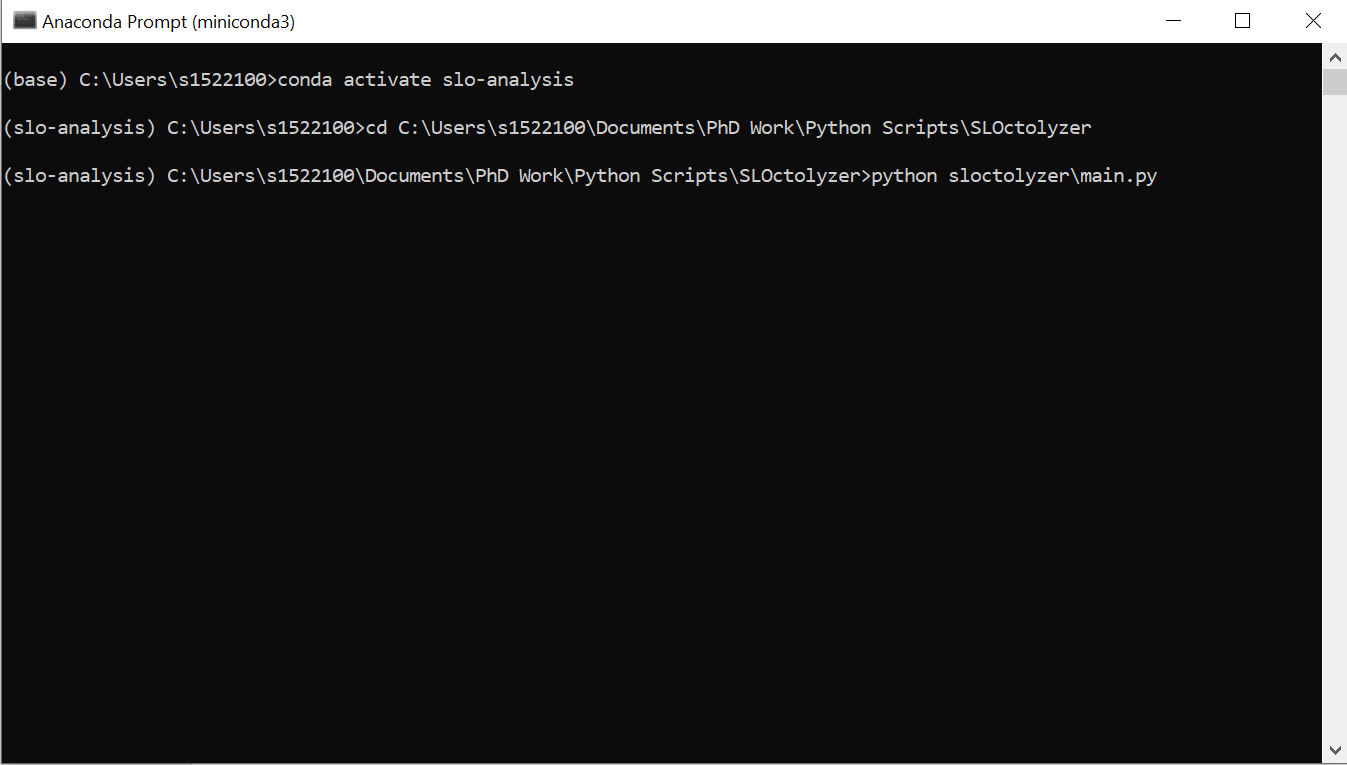
**2) RUNNING SLOctolyzer FROM THE TERMINAL**

1. Launch the Anaconda prompt application (see below)
   1. Select the Windows icon.
   2. Click the drop-down list of the Anaconda3 (64-bit) folder.
   3. Select the Anaconda Prompt (Anaconda3) application.



1. Activate the python environment which stores all python packages necessary to run the SLO analysis software, slo-analysis:
   1. Type “conda activate slo-analysis”
   2.  Press enter
2. Navigate to where SLOctolyzer has been saved, i.e.
   1. Type “cd path\to\SLOctolyzer”.
   2. Press Enter.



1. Once you have put all your images into the path specified in image\_directory, and (optionally) specified their scale/location/eye, you can run the SLOctolyzer:
   1. Type “python sloctolyzer\main.py"
   2. Press Enter.

We have provided two example SLO images to demonstrate using the software. Then try out editing the segmentation masks using ITK-Snap to recompute metrics (see 5, below).

**3) OUTPUT FILES FROM RUNNING SLOCTOLYZER AT THE TERMINAL**

* In the output\_directory there will be a log file analysis\_log.txt storing information printed out to user for each image analysed.
  + **Note:** This can be useful to inspect why an image failed to be analysed, or if the fovea was not detected, etc.
* In the output\_directory there will be a .csv file analysis\_output.csv storing metadata and feature measurements for each image file row-wise.
  + **Note:** This is the final output file which can be loaded in for data analysis.
  + **Note:** If there is an empty row with only a filename, this image failed to be analysed.
* By default, an additional directory output\_directory/segmentations is saved out containing images of the segmentation masks superimposed onto the SLO image file.
  + **Note:** This is helpful for quickly checking the outputs of SLOctolyzer’s segmentation module.
* By default, in output\_directory folders will be created for each image file (using the image’s filename for reference). Stored in each folder are
  + Feature measurements and metadata saved as output\_directory/filename/{filename}\_output.xlsx
  + A .txt file saved as output\_directory/filename/{filename}\_log.txt

This can be helpful if an unexpected error crashes the analysis run halfway through a large batch, as the software will be able to automatically identify previously analysed images based on the existence of output\_directory/filename/{filename}\_output.xlsx

**Delete the {filename}\_output.xlsx file if you wish to re-analyse the image associated with it.**

* In config.txt, by default the option save\_individual\_segmentations is set to 1 will save out the segmentation masks, original SLO image and images of the segmentations superimposed onto the SLO.
  + **Note:** You may want to set this to 0 if processing large batches of images, to save on memory consumption. However, the segmentation masks will not be accessible if set to 0, so if memory is not a problem/if analysing a smaller batch, keep this as 1.
  + Moreover, saving these segmentation masks out allows you the opportunity to manually annotate and fix any errors made by the segmentation models (see 5, below).

**4) RUNNING SLOCTOLYZER USING AN IDE**

* If running interactively from an IDE (VSCode, Jupyter notebooks/lab), use the notebook “usage.ipynb” to get started on using the analyse.py script for your own image file individually.
  + Here, there are parameter inputs to flag to save out the results into save\_path once flagging save\_results=True.
  + There is no need to specify the individual segmentation models as parameter inputs, as they will be automatically detected if left as None.
  + If you know the eye (right/left), location (macula- or optic disc-centred) and/or scale (in microns-per-pixel) you can specify these in the parameter inputs.
    - If you do not know these, leave them as None and they will be inferred as best as possible.

**5) CORRECTING SEGMENTATION ERRORS**

* We do not have any automatic functionality within SLOctolyzer to correct any vessel segmentation errors.
  + We rely on the user to identify any visible problems with vessel classification, etc.
  + The AVR value (CRAE divided by CRVE) can be used to quickly identify any segmentation errors. In particular, if AVR > 1 then this is outputted as a warning to the user, as this should not be the case.
* We do provide functionality to correct retinal vessel and optic disc segmentation via **ITK-Snap**. There are instructions on using ITK-Snap for manual annotations in the directory SLOctolyzer/instructions which describe how to use ITK-Snap and correct the binary vessel mask, and also the artery-vein-optic disc segmentation masks.
* Once the segmentation masks are saved out in the same folder with the original segmentation mask whcih was corrected, the pipeline can be run again and SLOctolyzer should automatically identify this additional manual annotation and re-compute the features!