README::

The OPIR Electron simulation was designed to create simulated data that mimicked real data, and create and test filters on that data. This was achieved in 2 parts. First, there is a file called OPIR\_Electron\_create\_data.py. This creates simulated data, stored in .csv files. Secondly, there are filter files that will read in the .csv files and apply a filter.

You will need some python libraries to run this code: gdal, osr, and numpy

**CREATE DATA:**

This is a self-contained file. It should need only the python libraries, and will output both .csv files and jpeg files of the data it creates.

Run the file, and it will ask for various inputs, such as x and y dimensions of the area, frequency of image capture (in Hertz), how long the image should be captures (in seconds), and finally a seed for the number generator. The last one is used for reproducibility of images.

The number of .csv and jpeg files produced is set to frequency\*number of captures.

Next, the program will ask if you will include background noise. If you choose to have noise, there is a choice between ‘coarse’ noise and ‘fine’ noise. **The ‘Coarse’ noise option is not working as intended, so use only the ‘Fine’ option@**

The code will output text file named ‘Data\_Header.txt’ containing these inputs for recording purposes. The Order is:

X\_dim

Y\_dim

Freq

Time

Seed

After that input, the program will run. The program will choose a random number of signal to appear, from 1 to 5. It will choose randomly for those signals to either move or stay still, starting location, signal strength, and if that strength will vary.

The code will then produce the .csv files and jpeg files.

**Filters:**

There are 3 filtering programs. These are named by the OPIR\_filter\_\*NAME\*.py paradigm. In each function, you must direct the function to the location of the .csv files you wish to use (line begins with ‘readRaster = ‘. Additionally, you must set the variable ‘number\_snapshots’, ‘x\_dim’, and ‘y\_dim’ variables to match the incoming data. (Number\_snapshots = freq\*time).

The function will read in the .csv files, apply the filter, and output a .jpeg of the filtered image.

**First Order Filter (FO):**

This function is based off the first order filter. It applies a filter based on the previous images (line 58). This filter needs a decay parameter, (set on line 38) that can be easily adjusted.

**Weighted Average Filter (WA):**

This filter applies a weight to each pixel based on a weighted average over the last ‘n\_samples’. The weight is calculated on line 58, the weights change by 0.5 ^n.