Pixel Binning

Carter Rhea

January 18, 2019

1 Pixel Binning

The following write-up will be very succinct and only contain a brief discussion of the algorithm in general, the data storage scheme, and the parallelization. The program **Pixel Binning** was created as an alternative to a *Weighted Voronoi Tesselation* algorithm; instead of binning the pixels before making a temperature map, we create a bin for each pixel in which the bin contains not only the pixels but also enough of its neighbors to achieve the desired signal to noise.

1.1 Primary Algorithm

The algorithm in this case is incredibly simple and naive; all of the complications arise from programming considerations such as parallelization and data storage. The algorithm is as follows:

1. Read in pixel information
2. Step through each pixel
   (a) Calculate nearest neighbors
   (b) If target Signal-to-Noise is reached save results to file; Else repeat

And that's really it!

1.2 Data Storage

Instead of keeping all of the data in memory during the process of stepping through each pixel, we will write out pixel data to a file and then clear the local memory. This way, even if we have to step through 1 million pixels, we only ever have the information for a single pixel (assuming serial execution) in memory at a time and all other information will be written to disk.

1.3 Parallelization

Since we would classify this algorithm as *embarrassingly parallel*, we used the python `multiprocessing` module to parallelize the `for` loop used to step through the pixels. We also set up a lock system for writing to the shared file in order to avoid any race conditions.

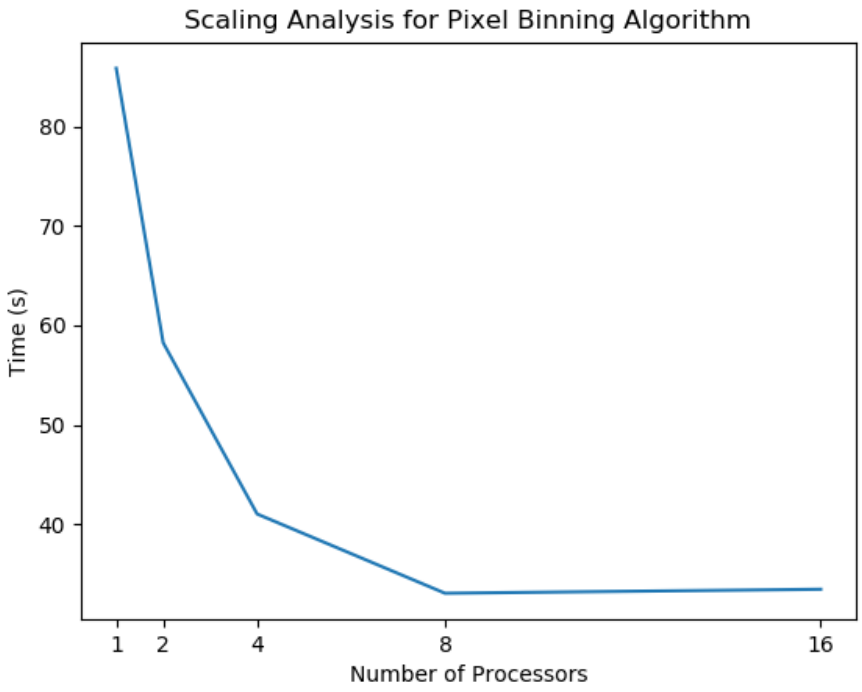

Figure 1: Parallel Timing Analysis run with *Intel Xeon(R) CPU E5-2620 v4 @ 2.10GHz* cores using *Linux Ubuntu 16.04.2 LTS 64-bit* with 62.8GB of RAM. The image used as an example had 1681 images and each pixel was required to have a Signal-to-Noise level of 10.