Detection of Clouds in Visible Band Nighttime Imagery

Raymond Xiao (ECE), Carnegie Mellon University

Project Information:

I am working with Christopher and Helga who are researchers at GFZ and two CMU students named Peiqi Liu and Prabh Baweja as a research intern. Our goal is to try and identify cloudy areas at night from satellite imagery using computer vision techniques. This project involves a lot of programming and will be done in Python using libraries like OpenCV. This project is relevant because accurate information on cloud occurrence is of great importance for a wide range of remote-sensing applications and analyses. Numerous studies have been conducted on infrared bands and other multiple sensors carried by the satellites to detect clouds. This project aims to further assist astronomers in identifying light pollution in various locations from photographs.

Internship Role:

My main goal in this project was to attempt to automate the detection of clouds in night time satellite imagery. Cloudy areas that the algorithm identifies would then be covered with a cloud mask so that researchers would be able to differentiate between cloudy and clear areas in an image. For this project, there were two main datasets used by me and my colleagues; the Luojia-1 dataset (Luojia 1 is a CubeSat (6U) sized earth observation satellite built by Wuhan University) and the Cities at Night dataset (images taken from the ISS). The Cities at Night dataset is relatively large with over 60 thousand images. Due to time constraints, only a small portion of that was used to test the algorithms.

Figure 1: Sample image of regions in Germany from the Luojia-1 dataset.

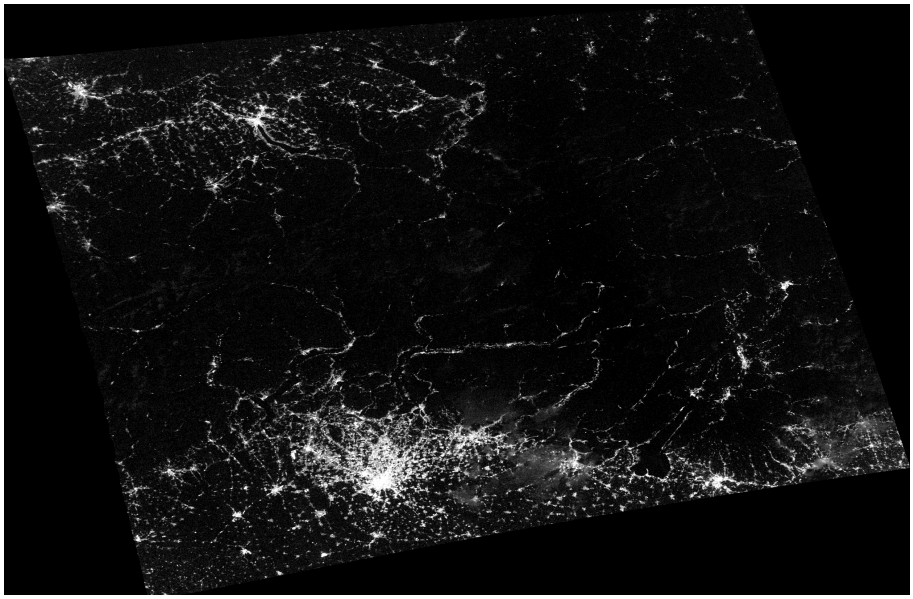
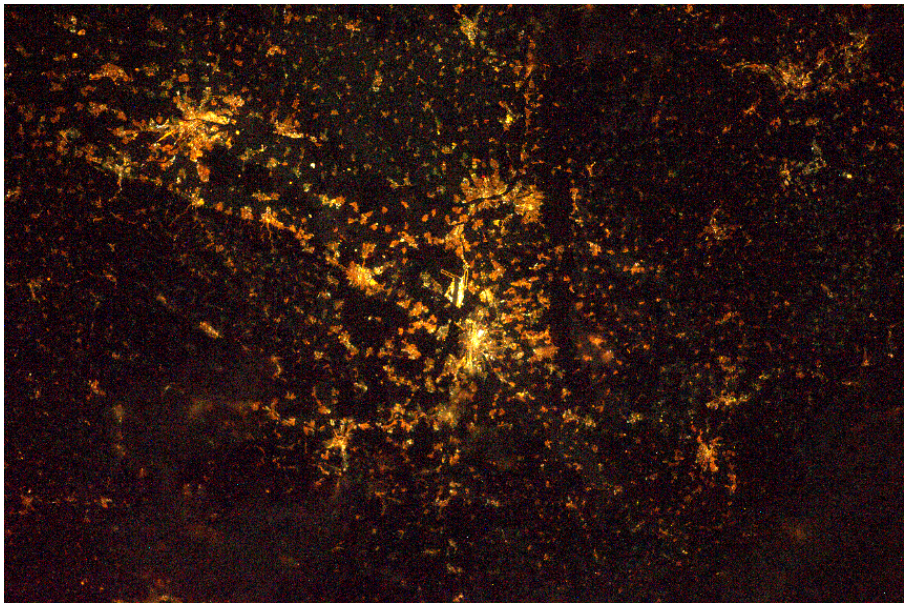
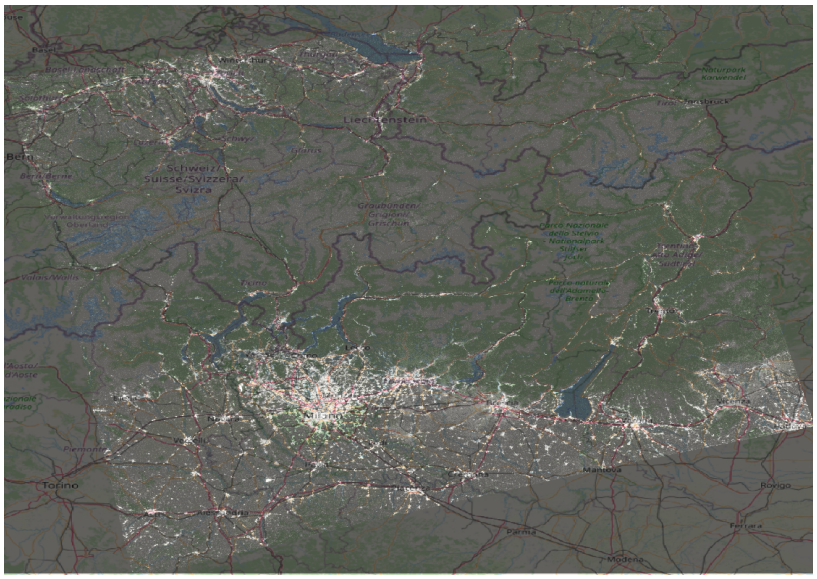


Figure 2: An example image of Luxembourg area from Cities at Night dataset



Challenges:

It is harder to identify cloudy areas in the night sky as compared to day time. Previously, researchers have developed algorithms to detect clouds in the night sky with the help of infrared satellites but our datasets are not composed of infrared images. In the visible band imagery, it is crucial to identify the city lights and the rural areas. This provides a confirmation towards the algorithm and a sanity check. Below is an example of a region in Germany overlaid over the OpenStreet maps to confirm the cities and the other areas in the region.



Cloud Detection Algorithm for Luojia-1 Satellite Images

For Luojia-1 Image, we started out trying basic approaches like applying a median filter on chunks of the image because cloudy areas should have a higher median. This, unfortunately, did not work as expected.

Later, Chris introduced some other techniques we can try, including Unsharp mask, a high-pass filter usually used to improve the clarity of an image. When applied to the nighttime satellite

image from the Luojia-1 satellite, the scatter of streetlights in cloudy areas is greatly reduced while the cloud stays mostly intact. This makes it easier to separate streetlights from clouds. This has become the foundation of this algorithm.

The next step is to remove all pixels brighter than a threshold. Since many images have very different local properties (eg. urban vs rural, cloudy vs clear) and we get bad results from images with wide ranges of brightness.

We later introduced an improvement so that the threshold is computed and applied to individual chunks of the image. This step is able to eliminate the brighter part of streetlights. The image is now only left with clouds and dimmer parts of the streetlights.

The next step is average blurring, a low-pass filter, which removes streetlights with no nearby cloud because of their relatively bright and sparse pixels (very bright pixels are already removed in the last step).

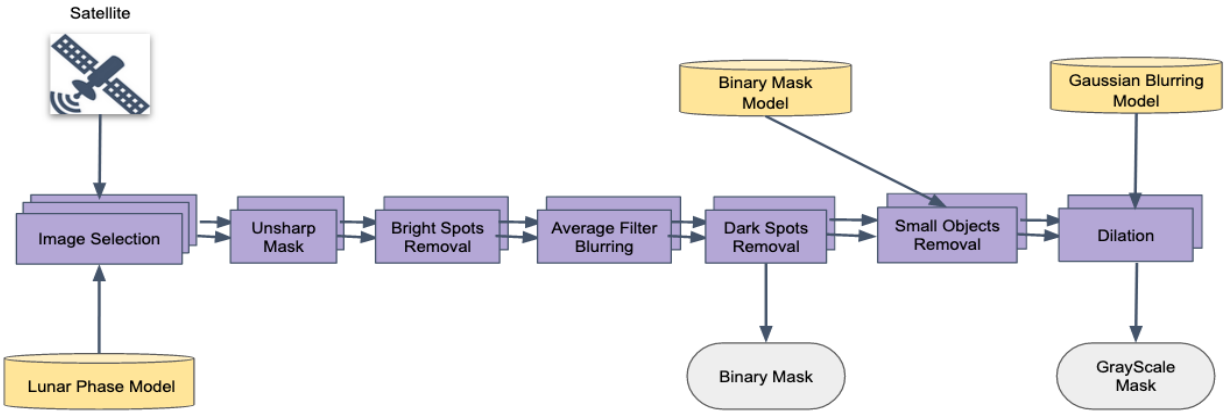
Right now the cloud mask is starting to take shape, but some noise reduction is still needed. The next step is to remove all pixels darker than a threshold. Similarly, the threshold is computed and applied to individual chunks of the image.

There are some remaining spotty bright noises that cannot be removed by the previous steps. The next step is to generate statistics of all connected components in the mask and remove the ones with an area smaller than a threshold (eg. the mean size of all connected components).

The mask now predicts where there are certainly clouds, but there are most likely clouds near the predicted areas. To be more conservative with the prediction, the next step is to run a grey dilation to extend the predicted area with the cloud. If gradient prediction is needed, another Gaussian blur can be applied to smooth out the dilated mask.

The process is not yet automated given our limited time to work on the algorithm. It does not work well for images with a large amount of moonlight so images with lunar phase close to a full moon need to be excluded manually. Also, the thresholds used for each step is manually set, with some variability according to the mean of the image.

Here is the system overview of the algorithm:



Cloud Detection Algorithm for Cities at Night Satellite Images

The algorithm for the Cities at Night dataset is similar to the previous algorithm both in terms of procedure and general idea. The images downloaded from the dataset are all in .NEF format which is a raw camera format. They must first be processed into readable data and then

transformed into a NumPy array. Since many of the satellite images are shots of the same general area with a translation, a list of similar images can be stitched together into one composite image. This composite image is then brightened to increase the contrast between artificial lighting and dark areas. Since we are defining a binary cloud mask (either there is a cloud in a particular area or there isn't), we want to convert our image to a grayscale one so that subsequent processing on it is quicker and more efficient. The image is then blurred with a Gaussian filter (which is a low pass filter) to reduce noise such as small isolated lighting and helps separate clouds from street lights. A high pass filter is then used to sharpen the image and bring into focus the remaining bright streetlights since they should not have clouds covering them.

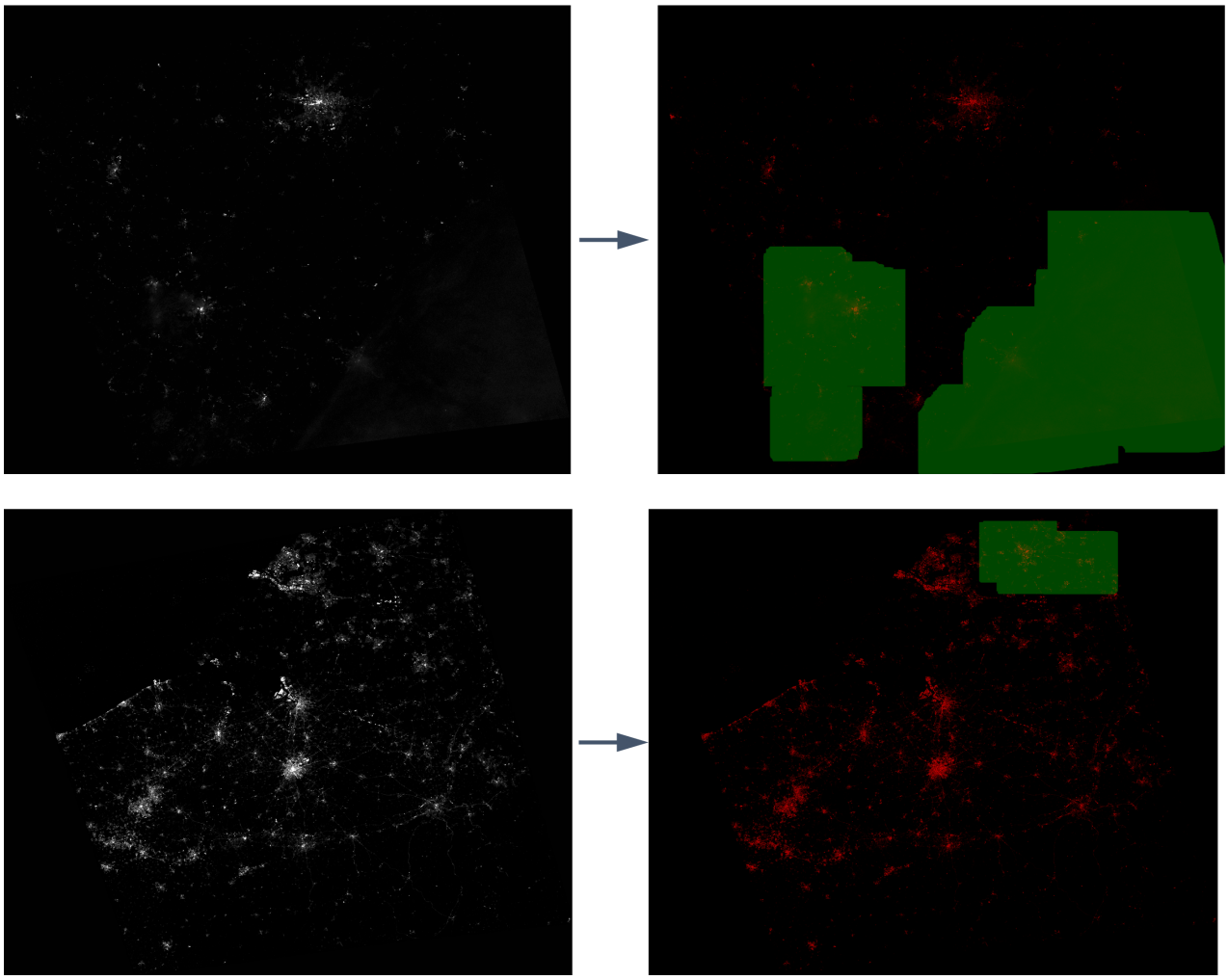
As stated before, since there are a wide variety of image landscapes and brightnesses to account for, using hardcoded thresholds results in poor performance. Thus, an adaptive threshold is used that changes for each image based on it's brightness and dimensions.

An initial cloud mask is formed based on the thresholded image. It is then opened (dilation followed by erosion) to remove small isolated chunks of clouds that are most likely unwanted noise. Finally, a flood fill algorithm is run to fill in any potentially remaining holes within the mask so that the masks are smooth and do not have any holes/sharp edges. This algorithm does not work particularly well on images that are extremely dark or extremely bright. Future steps would be to increase its robustness through further testing and tweaking of parameters.

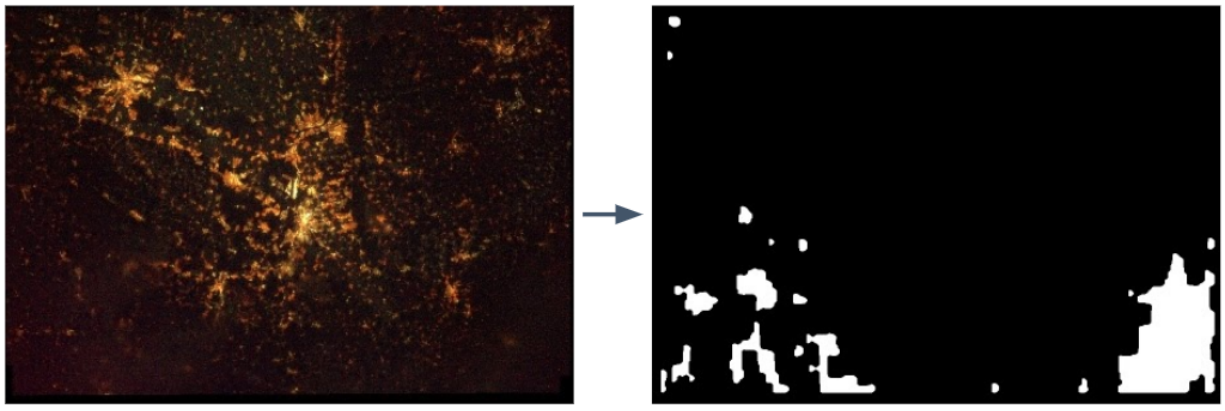
Results:

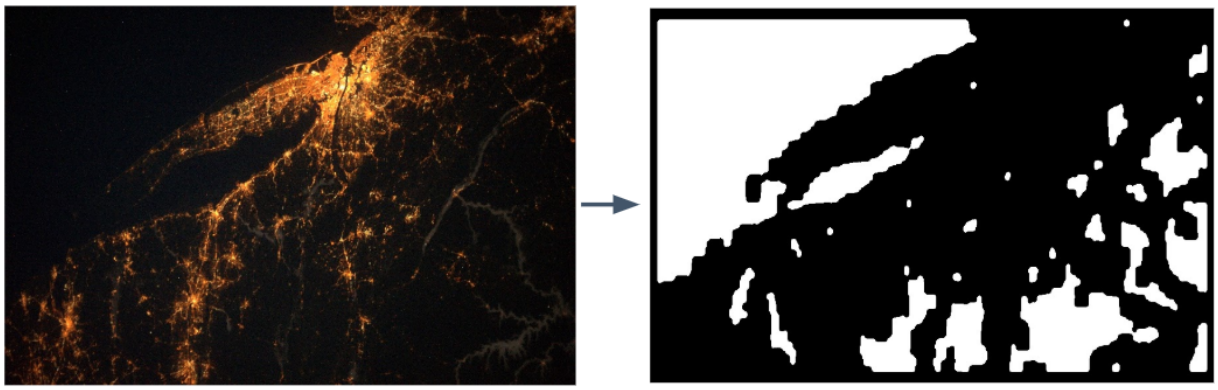
Both algorithms are generally able to correctly predict most clouds in non extreme images. The two following pairs of images are a successful case and a failed case respectively for the Luojia-

1 dataset . The image on the left is the original image and green areas are the predicted cloud masks.

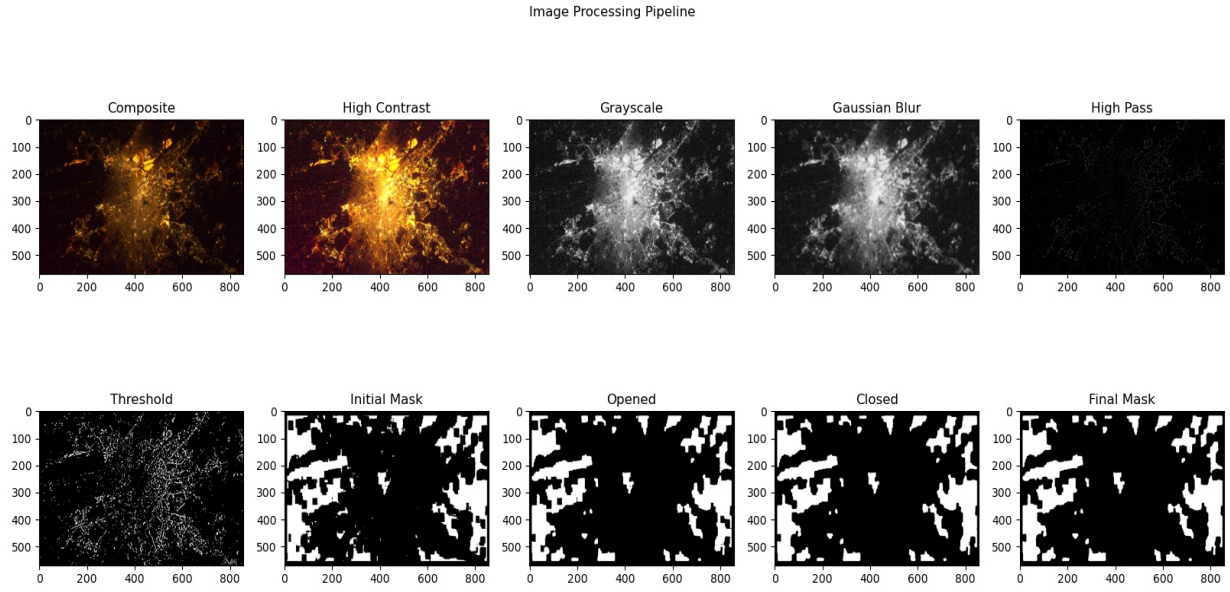


Here is a success case and a failure case for the Cities at Night cloud detection algorithm.





Lastly, a figure of the full image processing pipeline with intermediate images shown for every step.



Future Work:

Develop a computer vision deep learning algorithm to automatically use the image processing filters and segment clouds from the night sky. This technique will be evaluated on a large dataset to see the accuracy of this deep learning algorithm. We might be collaborating to develop a

robust algorithm for cloud detection and also try for a publication on the algorithm if we get satisfactory results.

Experience Description:

I came into this project not sure of what I would be able to accomplish and whether I would be proud of my work or not. As mentioned previously, I worked with Tony and Prabh as well as Christoper and Helga from GFZ. I enjoyed collaborating with these people from different backgrounds and I think that this diversity improved my project quality substantially. I especially enjoyed working with Tony and Prabh because they were tackling a very similar problem (just on different datasets) and so I was able to see what processing techniques they were using and incorporate/expand on those. Overall, I think our work environment was very friendly and collaborative which helped everyone.

Knowledge Gained:

The main takeaways that I have from this project are computer vision related skills and image processing techniques in addition to learning how to approach real world problems without an easy solution. More specifically, I learned a lot about the Python modules Numpy and OpenCV which were the main tools used in this project. In addition, I wrote a script to automate the download of satellite images from the dataset which exposed me to web scraping. On a less technical note, I also strengthened my teamwork skills through repeated collaboration and made connections with other researchers.

Timeline:

Date	**Time Spent**	**Tasks**
July 1	3 hrs	Initial meeting w/ Christopher Looked over links he sent
July 2	2 hrs	Researched about satellite images & clouds at night
July 5	4.5 hrs	Downloaded some test images & tried out some filters Most of time was spent reading documentation
July 6	3 hrs	Weekly meeting & visiting + reading sources that were sent
July 7-12	17 hrs	Learning more about raw images and how to process them How kernel sizes affect filters Morphological transformations Started fleshing out processing pipeline
July 13	0.5 hrs	Weekly meeting - discussed what everyone was doing and future plans/steps
July 14 - 15	6.5 hrs	Meeting w/ Tony and Prabh Testing out processing and making lots of small changes to try and optimize performance
July 17 - 19	5.5 hrs	More testing and tweaking of algorithm Expanded dataset to include ~10 images so far

		Writing script to automatically download images from Cities at Night website
July 20	4.5 hrs	Weekly meeting & beginning to modularize code with less hard coded constants
July 22	1.5 hr	Meeting w/ Tony and Prabh to discuss what's techniques are/aren't useful Working on helper functions to make code more clean
July 23 - 26	9 hrs	Finalized image processing pipeline Added function to plot results at every step so that whole pipeline can be visualized Testing on a lot more data and generalize to wide variety of images
July 27	4.5 hrs	Last meeting w/ Christopher since he is going on vacation Downloaded ~50 images total and beginning to test each image individually
July 28 - 31	8.5 hrs	Changed some functions to vary outputs based on image brightness rather than being hard coded so that it will hopefully have better performance Still doing lots of tweaking to constants to try and improve overall detection

Aug 3	3 hrs	Meeting w/ Helga to discuss what has been happening Wrote script for data augmentation (producing similar images from one) to be able to generate large data
Aug 5 - 6	5.5 hrs	Last stages of testing and refining Revised download script to be able to download every single image from Cities at Night .csv file (~60,000 images)
Aug 7	2.5 hrs	Meeting w/ Helga and small presentations about our code Working on Powerpoint presentation to present to class on Friday
Aug 8 - 9	4.5 hrs	Writing final research report