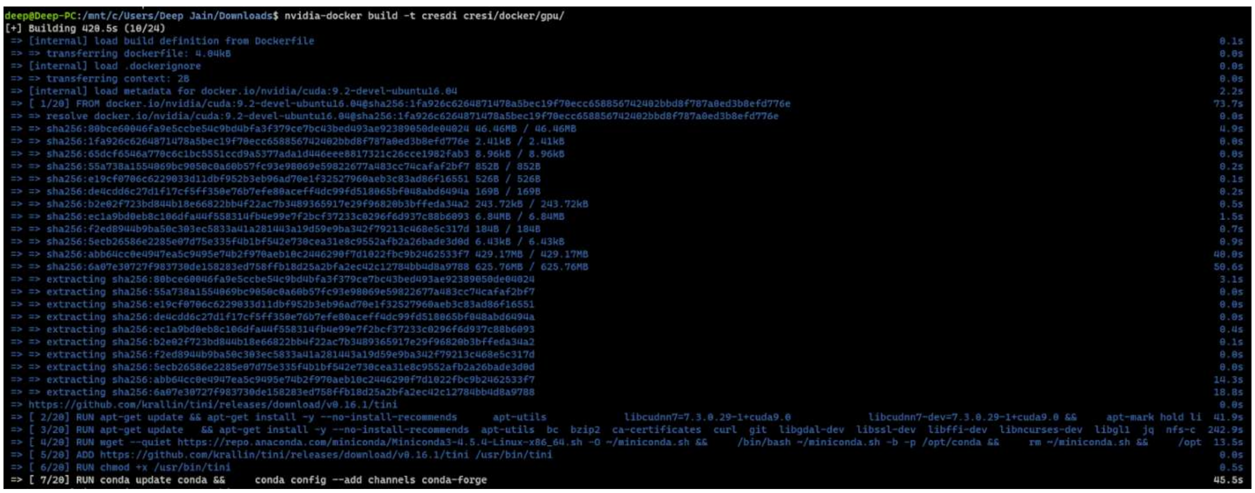
<u>CS 482: CV with CRESI</u>

Deep Jain

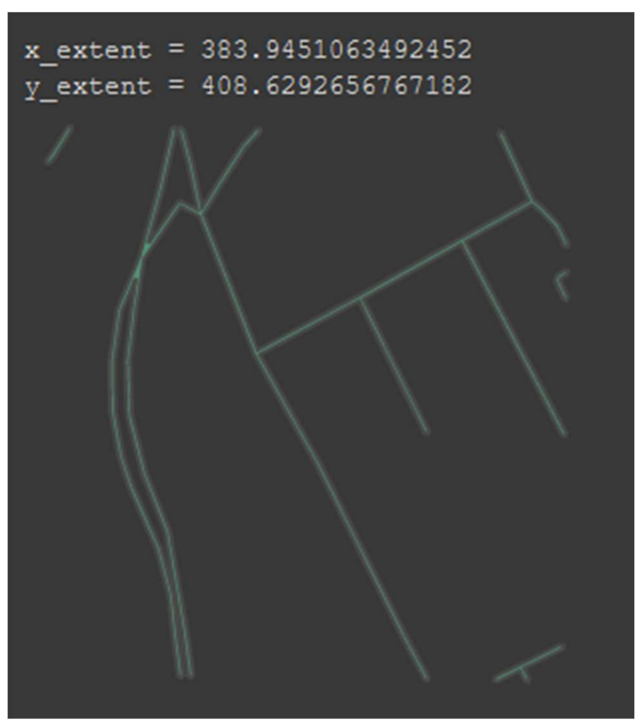
Part A: Difficulties and Success in Replicating Results

This project was both interesting and challenging. In my opinion, one of the most difficult part of this project was to set up an environment for CRESI to run correctly. Initially I thought about running it in WSL (Windows Subsystem for Linux). My reasoning was that Docker for Windows had functionality to run within WSL and I had a GTX 1050TI GPU. After setting up Docker so it can be used in WSL, I quickly came across a wide range of issues. One of my issues was that Docker and WSL were not talking to each other correctly. This was my initial output:



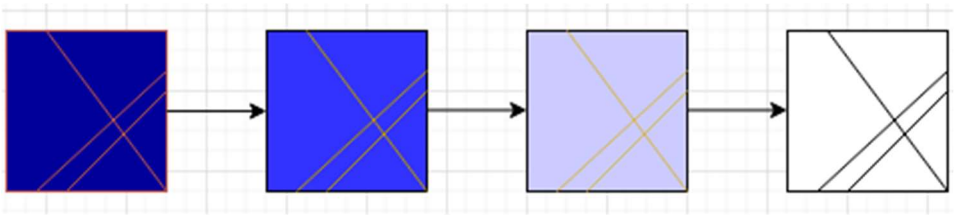
The Docker was building the CRESI files as expected, but when the environment check came, the environment was not being set up correctly. I would be getting errors explaining that the correct environment was not detected. After spending a day tinkering around with Docker's UI and WSL's settings, I decided to use Google Collab instead.

One of the biggest red flags I saw was the Google Collab did not have support for Nvidia Dockers as Ubuntu Linux had natively. As a result, I had to find a quick work around. I found out that one can build the necessary resources using a shell script so important dependency programs like GDAL can run with no errors. After getting the dependency set up, I effortlessly able to run the python files to prep the data and train CRESI to detect the roadways. I was able to get the following output of the busy intersections of Mumbai India:



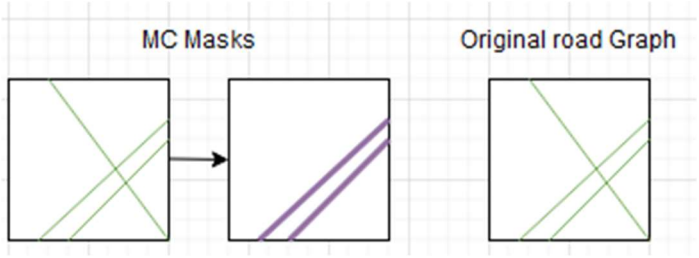
Part B: The Concepts Behind Cresi

The way CRESI works is by creating training masks. What this means is that certain algorithms are used to remove and isolate noise in images. RGB images are first created in order to properly have CRESI analyze the masks. We start then with a raw masked version of the image (initial output) and then you work your way up to just the skeleton structure. The algorithm is initially starting with the raw mask, and it advances to just the skeletal structure. The algorithm is then able to decide the specific roads the are available in the data. The final output of the graph is then placed into vectors. We use a Gaussian Kernel of a predetermined length and any other noise such as buildings, fauna, etc. are removed. The loss function, $L=\alpha cLCE+(1-\alpha c)Ldice,$ utilizes Cross Entropy in order to value the road connections more. After the data preparation is complete, we get the following output:



As explained before, the functions go through the removal of noise from the masks and the expected output just the skeletal structure of the roads that can be seen on the right.

CRESI also has a feature where its able to estimate the speed that is given by the speed masks. The way this works is by first looking at the segments of a midpoints. An 8x8 pixel patch is then decided on to be used to estimate the speeds. The speed of at each point of the skeletal mask is determined by first eliminating the low chance values. Then you take the average of the remaining values in the Multi-Class Masks. For each the speed is represented by the variable R, the time is represented with the variable t and the length is represented by the variable l. The average is calculated until you get the proper value of speed (R). The equation can be defined as $\triangle t=\triangle l/r$. The estimation of speed can be shown in the following diagram:



In the Multi Class Mask, the roads are color coded so speeds can be defined. The color purple is used to identify speed. In this case, 45 MPH.