**Rover Project Report:**

The main idea of the project is to find the path of the rover in an unknown environment by collecting the images at each point, analyzing them and send back to the rover to find the navigable path.

The analyzing part is done by performing color threshold to every image. Every image consists of number of pixels of which every pixel contains RGB values. The image at every point is composed of different RGB values. In this example, the dark color is treated as obstacle and the light color is treated as navigable terrain.

As per the RGB color values, we can differentiate the navigable terrain with the obstacle area as follows.

**Notebook analysis:**

For navigable terrain, the RGB values should be greater than (160, 160, 160) respectively. And anything less than those values is treated as obstacles. For the rocks, the RGB values are taken from color picker approximately. The modification is made to the color thresh function in the notebook to identify obstacle and rock samples. The resulting image is as follows:



The image coordinates are converted to rover coordinates using rover\_coords () function and they are converted to polar coordinates in rover space using to\_polar\_coords () function. From rover space pixels, they are converted to world coordinates using pix\_to\_world () function. The different figures are shown below:



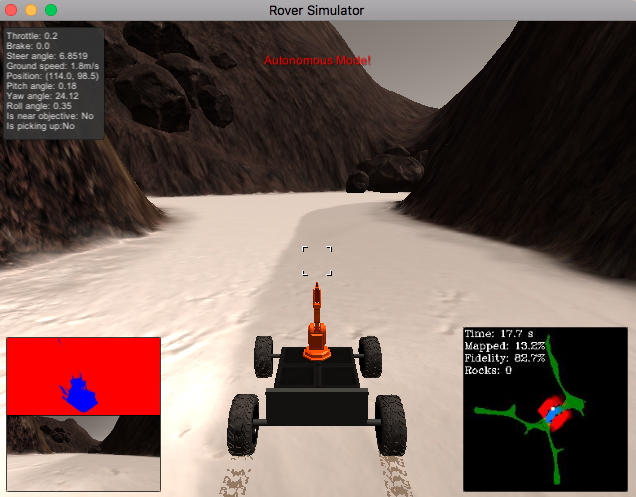
The process\_image () function is updated so that the images coming as input to rover are converted to rover centric and then to world coordinates and the map is published as the output video by using moviepy function. The video is shown below.



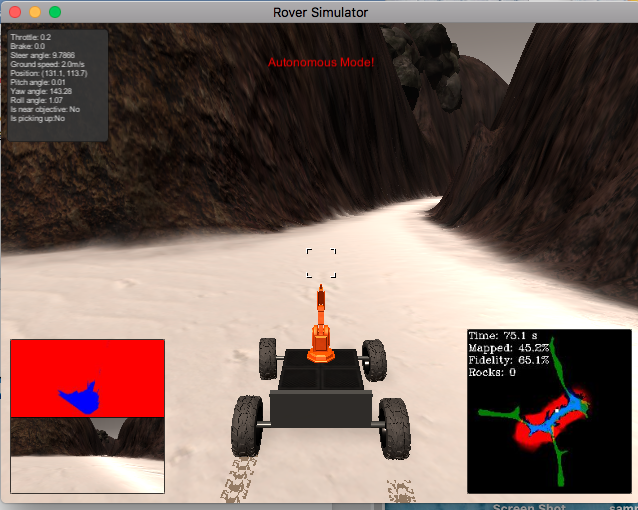
Autonomous Navigation and Mapping:

The perception.py is modified same as the process\_image () function. The main processes include calculating the perspective transform to the rover images and identifying the navigable and obstacle pixels simultaneously. The improvements are made to the rock pixels to display it in the white color on the map. All these are included in the perception function where it is called from the drive\_rover.py.

We launch the project with drive\_rover.py where the remaining steps perception.py and decision.py are launched from drive\_rover.py. Slight or no modifications are made to the decision.py in this project as the rover can be able to map autonomously over 40% with more than 60 % of the fidelity.



The above figure shows the initial state of Rover when launched in the autonomous mode:



The state of rover when it is 40% map, above 60% fidelity and with the identification of one rock on the map is shown below:

There are some improvements that can be made to the project to prevent some issues that rover faces while going through autonomous navigation.

1. To prevent the rover not to move in the same path or in loops continuously. To do this my idea is to locate the pixels that rover has already gone through and feed them with the new pixels which rover has not gone.
2. To prevent the rover to get stuck on the small rocks. My idea to prevent this is to steer continuously to right or left until the rover velocity is less than the max velocity and rover is in forward mode.

**The project is simulated in the rover with the following configuration:**

Device: Mac OSX Sierra 10.12.5

Processor: 2.5G Hz intel core i5

*In the simulator*: Screen resolution: 640 \* 480, Graphics Quality: Good