Introduction

This is the explanation of what I did in order to complete this project.

Simulator settings:

- Screen resolution: 1024 x 768 (Windowed)
- Graphics quality: Good
- FPS: between 13 and 20

I wrote my thoughts for each of the main functions:

perception.py

First of all I populated the necessary functions with the code I tested in the notebook, I just made minor changes due to the variables name. I wrote *obstacle_color_thresh* in order to select the obstacles, and I found the threshold values using a histogram.

For the rock detection I wrote the function *locate_rocks* which does the identification using color threshold for each RGB channel independently and then filling the holes of the binary image (executing *binary_fill_holes* from scipy.ndimage) and finally doing a AND of the three binary images. All this operations are done before the execution of *perspect_transform* in order have a rather circular object to detect. There is room for improvement here using color thresholding with another color space.

I defined the function *to_cart_cords* to transform coords from polar to cartesian, because I need it for testing other functionality.

In *perception_step* function, I added a *binary_fill_holes* call over ground binary image in order to improve the ground mapping fidelity. It is used with default parameters, so it only fills small holes, I didn't try filling bigger holes and I'm not sure how much it helps to improve fidelity. What definitely improves mapping fidelity is the use of thresholding for pitch and roll.

I added the binary image of rocks detection without *perspect_transform* to the vision image in order to have a better estimate of what the algorithm is doing.

I also estimate the distance to an obstacle in front of the rover inside a FOV of 2º, this to improve navigation and obstacle avoiding.

I tried to establish a "memory" in order to check if the rover had already mapped a place, but it didn't work very well.

decision.py

In "forward" mode I added the condition of "in front distance greater than 2m" for moving, to avoid hitting object in front of the rover. I also keep the steering angle when stopping.

I tried to implement the rock picking function, but it didn't work at first approach and I didn't try another algorithm.

drive_rover.py

I added other parameters to the rover state as they were needed.

supporting_functions.py

I changed the separator character from ";" to "," in the info stream.
I added more info on screen during developing, this was quit useful.

About Notebook

In the Notebook, I followed the instructions in order to complete the functions. In *process_image()* I changed the warped image for an image from the rock detection function *locate_rocks()* to see what was going on and if the algorithm was working fine. I added 100 for each detected position in order to see it clearly on the map. I also added an image of the current's rover position just for testing purposes. This time I added 50 for each position.