Project: Search and Sample Return

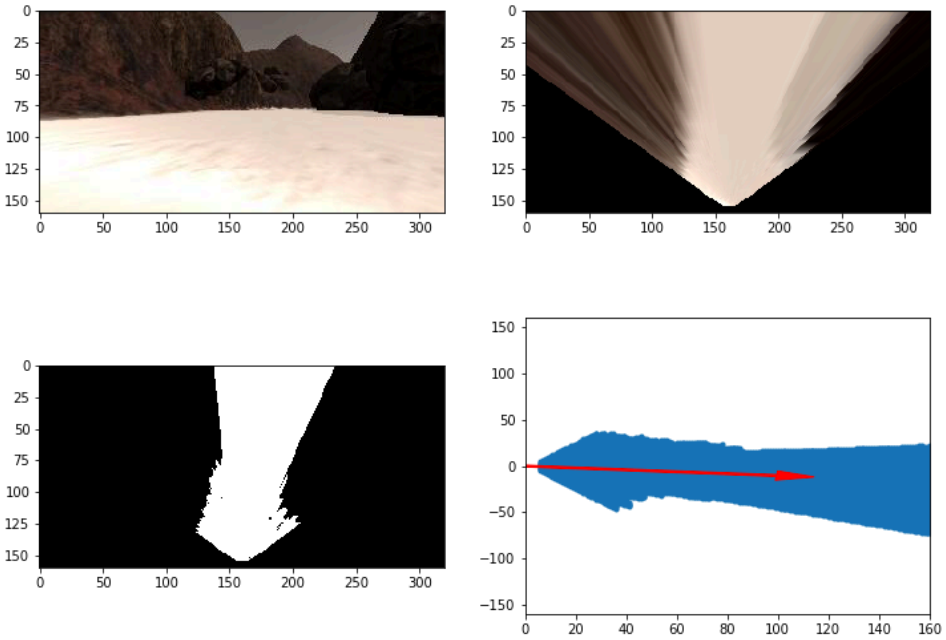
Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

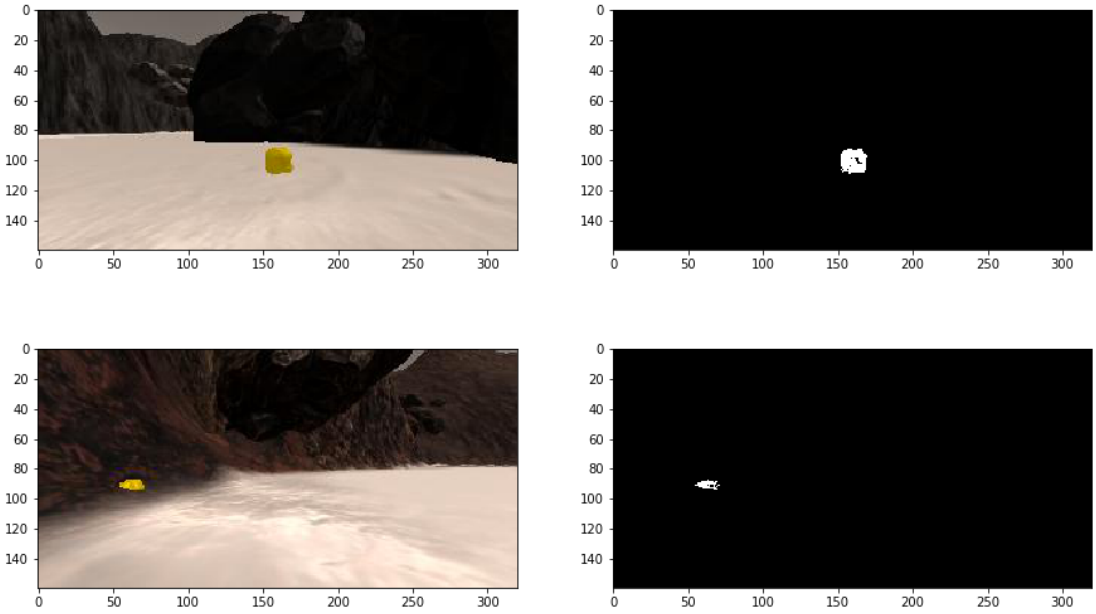
Notebook Analysis

1. Run the functions provided in the notebook on test images (first with the test data provided, next on data you have recorded). Add/modify functions to allow for color selection of obstacles and rock samples.

a) I modify the perspect_transform function to add mask return and modify the color_thresh that RGB > 180 to get threshed color select.



a) I add the find_rocks function to find rocks color than R and G > 110 and B < 50, and test the example rock picture and one other picture in test video.



1. Populate the process_image() function with the appropriate analysis steps to map pixels identifying navigable terrain, obstacles and rock samples into a worldmap. Run process_image() on your test data using the moviepy functions provided to create video output of your result.

1) Define source and destination points for perspective transform

2) Apply perspective transform

3) Apply color threshold to identify navigable terrain and use perspect_transform function return musk to calculate obstacles, then use find_rocks function to get rock samples

4) Convert thresholded image pixel values to rover-centric cords

5) Convert rover-centric pixel values to world cords

6) Update worldmap, if image has rock, update it in the worldmap with white color RGB=255, if obstacles worldmap has terrain color, delete the obstacles value to get clean terrain color in worldmap.

7) Make a mosaic image, below is some example code

Autonomous Navigation and Mapping

1. Fill in the perception_step() (at the bottom of the perception.py script) and decision_step() (in decision.py) functions in the autonomous mapping scripts and an explanation is provided in the writeup of how and why these functions were modified as they were.

In perception_step, I modified the same as process_image(), the different part is that I test how to find and pick up rocks. So if it find rock , the Rover. nav_angles get the rocksmap and run close to the rock.

In decision_stop():

a) In mode 'forward', I add Rover. stopspeed_num value to count and prevent car in forward mode but can't go forward. And add Rover. loopstep value to prevent car drive in loop turning around. Than modified the steer value strategy and change mode strategy.

b) In mode 'stop', I add Rover.stopsteer_num to count the condition that car turn left but it doesn't work, and turn right instead.

c) Add mode 'rock', if find rocks , it will close to the rocks and pick up.

2. Launching in autonomous mode your rover can navigate and map autonomously. Explain your results and how you might improve them in your writeup.

Note: running the simulator with different choices of resolution and graphics quality may produce different results, particularly on different machines! Make a note of your simulator settings (resolution and graphics quality set on launch) and frames per second (FPS output to terminal by `drive_rover.py`) in your writeup when you submit the project so your reviewer can reproduce your results.

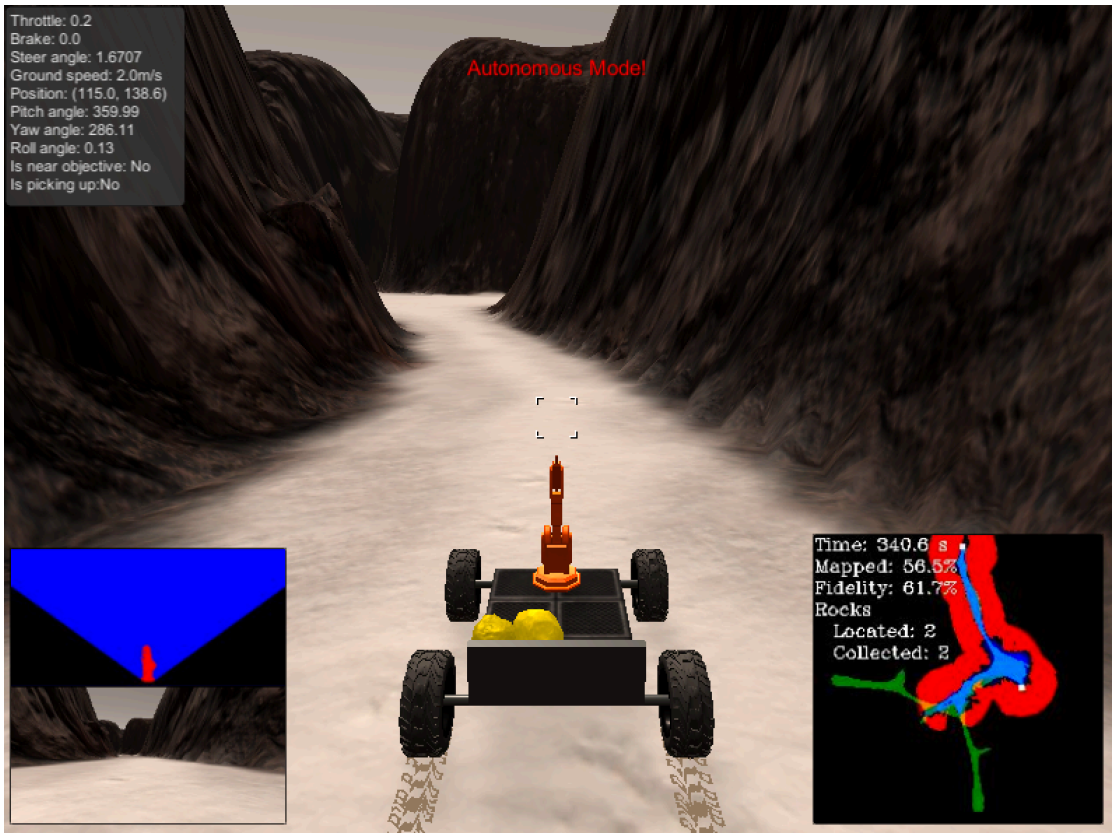
Current FPS: 29, Screen resolution: 1024x768; Graphics quality: Good;

Mapped 40%~60%, Fidelity 60%~70%

I think the most important thing is let the car continue driving. So I add many strategies to prevent car stop or turn round that I write in the last part. I run the drive_rover.py many times and it can always continue driving and pick up stocks.

I try to speed up the car to 2.5m/s, but I found it hard to control directions and easy to hit the wall, and make fidelity lower. But when I update some parameters like stop_forward and go_forward, the car drive more efficient and reduce the frequency to hit the wall.

From the result I might improve them in many other way in the future. (a) When in bifurcation, find the right way to prevent in the duplicate road. (b) Find a strategy to speed up and brake, so it can drive more efficient.




Throttle: 0.2
Brake: 0.0
Steer angle: -15.0000
Ground speed: 0.7m/s
Position: (144.7, 110.9)
Pitch angle: 0.33
Yaw angle: 300.34
Roll angle: 359.57
Is near objective: No
Is picking up:No
Autonomous Mode!
Time: 197.7 s
Mapped: 46.8%
Fidelity: 70.4%
Rocks
Located: 2
Collected: 2