Project: Search and Sample Return

By: S. Aravindh Annamalai

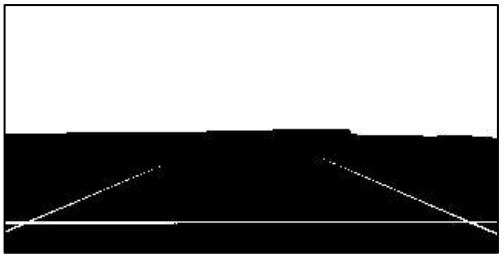
Writeup / README

Goals of This Project:

- Download the simulator and take in data in the training mode.
- Run the Jupyter notebook file and make the necessary changes such as for obstacle and rock identification. Populate the **process_image** function with the various functions.
- Use moviepy to process the images and create a video from the test dataset.
- Fill in the **perception_step** function to process the images and create the worldmap
- Fill in the **decision_step** to make the rover navigate and map the terrain autonomously.

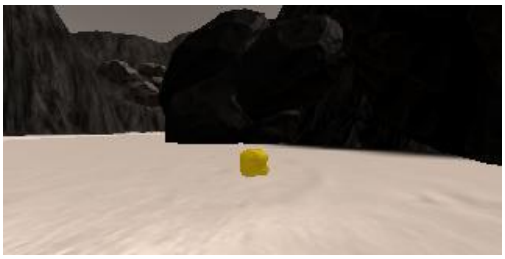
1. Notebook Analysis

-**Thresholding For Obstacles**: I created a function **obstacle_thresh** similar to the **color_thresh** function.It checks for pixels that lie below the threshold which is 160 in this case.



```
def obstacle_thresh(img, rgb_thresh=(160, 160, 160)):
    # Create an array of zeros same xy size as img, but single channel
    color_select = np.zeros_like(img[:,:,0])
    # Require that each pixel be above all three threshold values in RGB
    # above_thresh will now contain a boolean array with "True"
    # where threshold was met
    above_thresh = (img[:,:,0] < rgb_thresh[0]) \
                & (img[:,:,1] < rgb_thresh[1]) \
                & (img[:,:,2] < rgb_thresh[2])
    # Index the array of zeros with the boolean array and set to 1
    color_select[above_thresh] = 1
    # Return the binary image
    return color_select
```

-**Thresholding For Rocks**: I created a thresholding function for rocks by checking if the pixels values lie within the appropriate value for gold color.[Here the threshold values were rgb_thresh=(75,250,50,50)]



```
def rock_thresh(img, rgb_thresh):
    color_select = np.zeros_like(img[:,:,0])
    above_thresh = (img[:,:,0] > rgb_thresh[0]) & (img[:,:,1] < rgb_thresh[1]) \
                & (img[:,:,1] > rgb_thresh[2]) \
                & (img[:,:,2] < rgb_thresh[3])
    color_select[above_thresh] = 1
    return color_select
```

-Process Image Function:

In the process_image function I processed the images by calling the previously added functions.

- First I read the image in the current instance of the data class and stored it in image.I also defined the source and destinations.

```
image=mpimg.imread(data.images[data.count])
source = np.float32([[14, 140], [301 ,140],[200, 96], [118, 96]])
destination = np.float32([[image.shape[1]/2 - dst_size, image.shape[0] - bottom_offset],
              [image.shape[1]/2 + dst_size, image.shape[0] - bottom_offset],
              [image.shape[1]/2 + dst_size, image.shape[0] - 2*dst_size - bottom_offset],
              [image.shape[1]/2 - dst_size, image.shape[0] - 2*dst_size - bottom_offset],
              ])
```

- Then using the **perspective_transform** function I warped the image and then performed thresholding for navigable terrain,rocks and obstacles using the 3 different thresholding functions.

```
# 2) Apply perspective transform
warped=perspect_transform(image,source,destination)
# 3) Apply color threshold to identify navigable terrain/obstacles/rock samples
threshed=color_thresh(warped,(160,160,160))
obstacle_threshed=obstacle_thresh(warped)
rock_threshed=rock_thresh(warped)
```

- Then I converted the x and y coordinate of the thresholded images into rover coordinates using rover_coords function and to world coordinates using pix_to_world function.

```
# 5) Convert rover-centric pixel values to world coords
xpos=data.xpos[data.count]
ypos=data.ypos[data.count]
yaw=data.yaw[data.count]
xpix_world,ypix_world=pix_to_world(xpixel,ypixel,xpos,ypos,yaw,200,10)

xpix_world_obstacle,ypix_world_obstacle=pix_to_world(xpixel_obstacle,ypixel_obstacle,xpos,ypos,yaw,200,10)

xpix_world_rock,ypix_world_rock=pix_to_world(xpixel_rock,ypixel_rock,xpos,ypos,yaw,200,10)

# 6) Update worldmap (to be displayed on right side of screen)
data.worldmap[ypix_world,xpix_world,2]=255

data.worldmap[ypix_world_obstacle,xpix_world_obstacle,2]=255

data.worldmap[ypix_world_rock,xpix_world_rock,2]=255
```

- The world coordinates were then added to the worldmap.I did not make any changes to this part of the code.

2. Autonomous Navigation and Mapping

Perception_step

- I modified the perception_step function by running the various functions used for porcessing the image.I also added the function necessary for obstacle and rock detection.
- I introduced a condition where the coordinates of the terrain, rocks and obstacles are added only if the roll and pitch values of the rover are within a certain value

```
if (Rover.roll < 5) & (Rover.pitch <5) :
    # 7) Update Rover worldmap (to be displayed on right side of screen)
    Rover.worldmap[terrain_world_y,terrain_world_x,2]+=1
    Rover.worldmap[rock_world_y,rock_world_x,1]=255
    Rover.worldmap[obstacle_world_y,obstacle_world_x,0]-=1
```

Decision_step

- At first I changed all the conditions and made several functions such as start rover,stop rover , and turn rover so that I could use these as building blocks.Unfortunately it did not work properly.
- So I went back to the original code and made only a few necessary changes.I introduced a condition where in the first 10 seconds the rover would turn by taking the mean direction into account and adding an offset.By this time the rover would have reached the wall so after 10s, the rover turns by taking the maximum angle and subtacts an offset thereby following the left side wall.

```
if (Rover.total_time<10):
    Rover.steer = np.clip((np.mean(Rover.nav_angles * 180/np.pi) + 10), -15, 15)
Rover.steer = np.clip((np.max(Rover.nav_angles * 180/np.pi) - 30), -15, 15)
```

- I also added a time delay so that when the rover mode is forward but the rover is not moving or its stuck then the rover mode is changed to stop and the time delay helps it to exit from continuous turning.

```
if (Rover.vel<0.1):
    Rover.mode='stop'
    Rover.loop_num=1
    Rover.steer=-15
    time.sleep(0.5)
```

Results:

The rover seems to have some difficulty in starting.But once it has started following a wall it is able to move very quickly and accurately.The fidility is around 60%. I rover sometimes gets stuck when it goes into a cornor surrounded by rocks.

If I were to pursue this project further I think I could improve the fidility as well as introduce a function to pickup the rocks and return it to the center.

NOTE: Resolution : 1280 * 720 ; Graphics Quality: Fantastic ; FPS:25