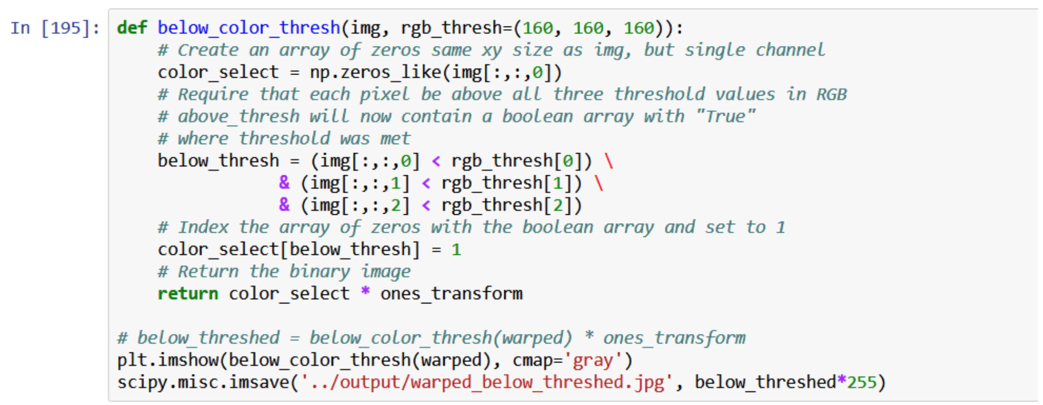
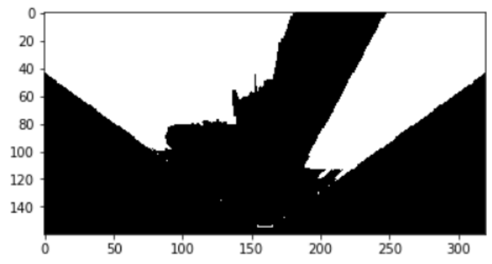
SEARCH AND SAMPLE RETURN PROJECT WRITEUP

NOTEBOOK ANALYSIS:

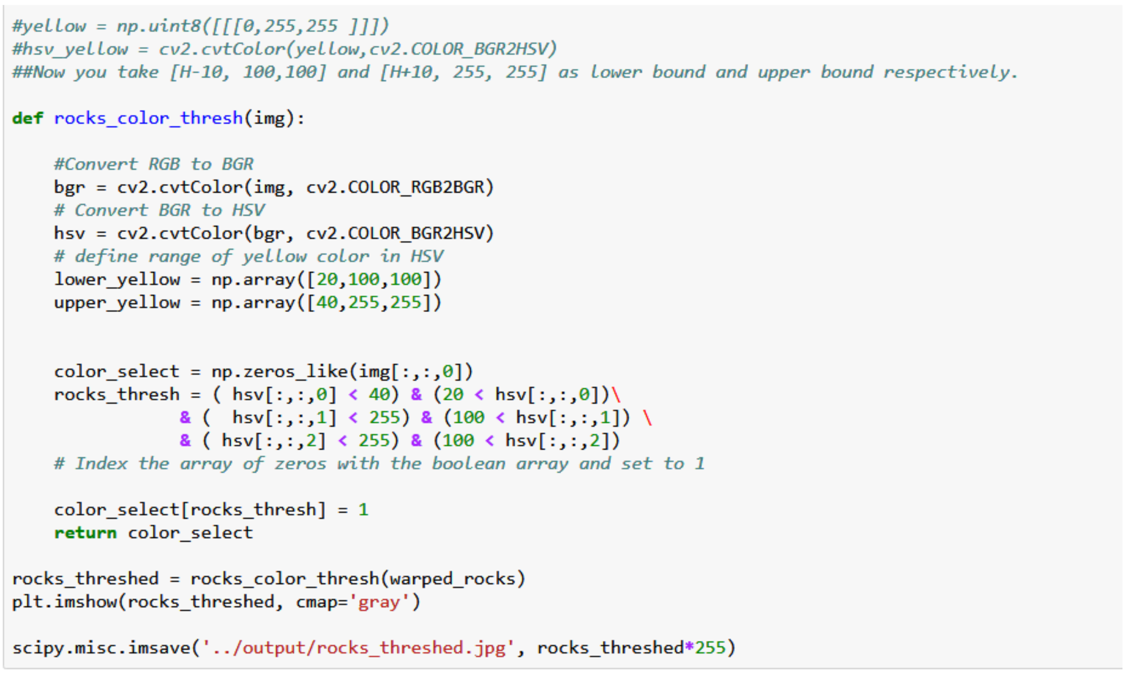
1) Obstacle and rock sample identification:



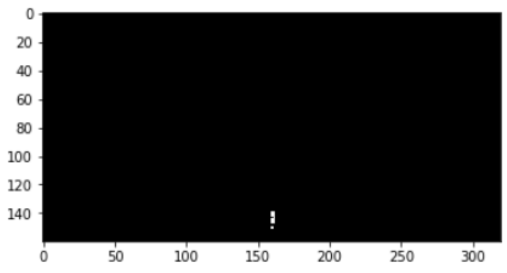
```
In [195]: def below_color_thresh(img, rgb_thresh=(160, 160, 160)):
              # Create an array of zeros same xy size as img, but single channel
              color_select = np.zeros_like(img[:,:,0])
              # Require that each pixel be above all three threshold values in RGB
              # above_thresh will now contain a boolean array with "True"
              # where threshold was met
              below_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[below_thresh] = 1
              # Return the binary image
              return color_select * ones_transform

          # below_threshed = below_color_thresh(warped) * ones_transform
          plt.imshow(below_color_thresh(warped), cmap='gray')
          scipy.misc.imsave('../output/warped_below_threshed.jpg', below_threshed*255)
```



For identifying the obstacles, the logic of color_thresh has been reversed (> changed to <). However, it also considers the area which is not in field of view of the bot also as obstacles (white). So another function ones_transform = perspect_transform(ones, source, destination) has been multiplied with the final output of the modified color_thresh function to take the field of view issues into consideration.



```
#yellow = np.uint8([[[0,255,255 ]]])
#hsv_yellow = cv2.cvtColor(yellow,cv2.COLOR_BGR2HSV)
##Now you take [H-10, 100,100] and [H+10, 255, 255] as lower bound and upper bound respectively.

def rocks_color_thresh(img):

    #Convert RGB to BGR
    bgr = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    # Convert BGR to HSV
    hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
    # define range of yellow color in HSV
    lower_yellow = np.array([20,100,100])
    upper_yellow = np.array([40,255,255])


    color_select = np.zeros_like(img[:,:,0])
    rocks_thresh = ( hsv[:,:,0] < 40) & (20 < hsv[:,:,0])\
                & (  hsv[:,:,1] < 255) & (100 < hsv[:,:,1]) \
                & ( hsv[:,:,2] < 255) & (100 < hsv[:,:,2])
    # Index the array of zeros with the boolean array and set to 1

    color_select[rocks_thresh] = 1
    return color_select

rocks_threshed = rocks_color_thresh(warped_rocks)
plt.imshow(rocks_threshed, cmap='gray')

scipy.misc.imsave('../output/rocks_threshed.jpg', rocks_threshed*255)
```



For identification of rock samples, cv2 documentation for color classification has been studied and it was found that the hsv values for yellow lie in the range (20,100,100) and (40,255,255) as mentioned above.

2) `process_image():`

Firstly, the source and destination points have been defined, first one by approximation from the image and second, as midpoint of bottom edge, with some offset. Threshold predefined functions (as defined above) have been applied on the perspective transform. Next, thresholded image pixel values have been converted to rover-centric coordinates and then, to world coordinates. As it is being performed for a single image, the color channels in the worldmap have simply been set to Red, Green, Blue for Obstacles, Rocks and Navigable Terrain respectively. Then, in order to ensure the colors don’t mix, a variable has been defined which gets set to True whenever Blue channel value > 0, and this truth value is used to set obstacle values to zero at places where there are obstacles, thus eliminating the possibility of overlap of the colors. Finally, the mosaic image has been defined for displaying the output image (Nothing much modified here).

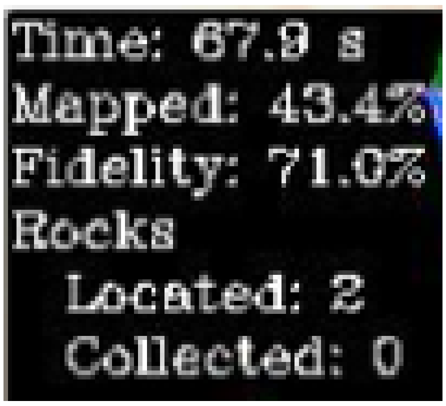
AUTONOMOUS NAVIGATION AND MAPPING:

Before the `perception_step()` function, the color threshold function from the notebook has been added and a variable 'mask' has been added in the perspect_transform function for the obstacles. In the `perception_step()` function, source & destination points, perspective transform, color threshold functions have been defined as have been in the notebook. For the vision image to be displayed on the left side of the screen, the obtained truth values of the three functions of Obstacles, Rocks and Navigable Terrain color threshold are being multiplied with 255 to assign them the corresponding RGB value. As explained earlier in the notebook, map image pixel values are being converted to rover-centric coordinates and then, to world coordinates. Now, instead of simply setting color channels in the worldmap to Red, Green, Blue, they are progressively added a particular value as series of images are being fed in, and thus, the map slowly keeps forming. The navigable terrain has been given more weight for clear visibility in the map. Rocks function has been separately defined because of the problem of height of the rocks, i.e. they are not flat surfaces and this might cause an error in mapping them. Therefore, minimum rock distance function has been used to account for that error.

Then, Navigation angles are being obtained from to_polar_coordinates function to be used in the `decision_step().` Decision step function has been left as it is. The decision tree has been made through a series of conditional statements, for example for setting the throttle value to throttle setting when its velocity is less than maximum given that there is enough navigable terrain, initially an if statement has been defined for ensuring there is enough navigable terrain pixels inside which another if statement has been defined to ensure the rover velocity is below maximum velocity.

The FPS is 14 and resolution is 800 * 600 with graphics quality 'Good'.

Satisfactory results were obtained with the decision tree as it is. I have run it for 67 seconds. The rover has mapped 43.4% of the environment with 71% fidelity. It has located 2 rock samples. The results can be improved by designing better decision tree with more specifications, re-evaluating the source points during perspective transform for better accuracy, adjusting the color threshold values for navigable terrain and obstacles & using a better criterion for accounting for the rock's height.



```
Time: 67.9 s
Mapped: 43.4%
Fidelity: 71.0%
Rocks
  Located: 2
  Collected: 0
```