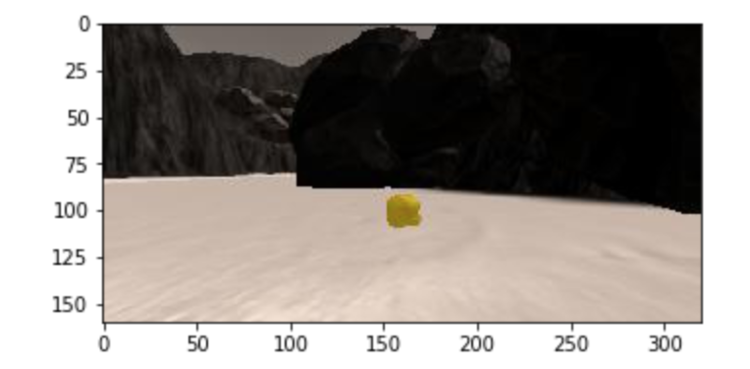
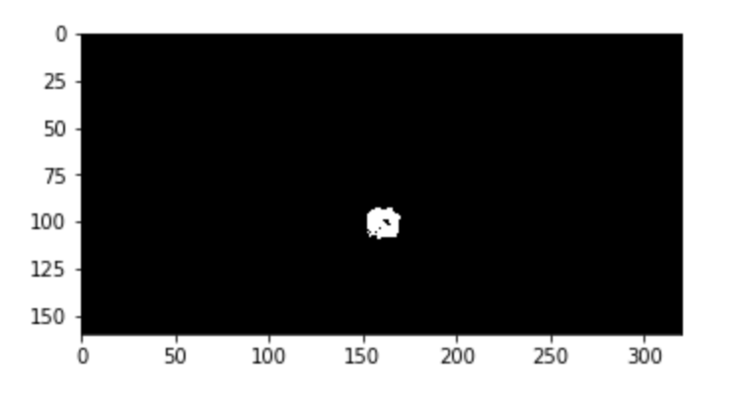
**SEARCH AND SAMPLE RETURN**

**By Patrick wellins**

Rock Identification For rock identification and mapping, I used a function called 'find\_rocks'. The 'find\_rocks' function takes an image and a tuple called 'levels' as inputs. The objective of this function is to take the input image that contains a rock and apply color manipulations in a manner that isolate the rock in the image. The meaning of 'isolate' in this context means the rock shows up as white in the image, and everything that is not a rock in the image shows up as black. The tuple 'levels' contains three integer values that determine which pixel values are True or False in the three color channels of the input image. These boolean values are stored in a variable named ‘rockpix’ at specific indices. A variable named 'color\_select' is initialized with zeros with the same width and height dimensions as the rock image, but only one layer deep. The indices that have the value 'True' in ‘rockpix’ set the corresponding indices in 'color\_select' to the number 1, all other values in 'color\_select' remain with the value 0. The 'color\_select' variable is the output of the 'find\_rocks' function, this output image either contains an image of a white rock with a black background if a rock is present in the image, or a black background if there is no

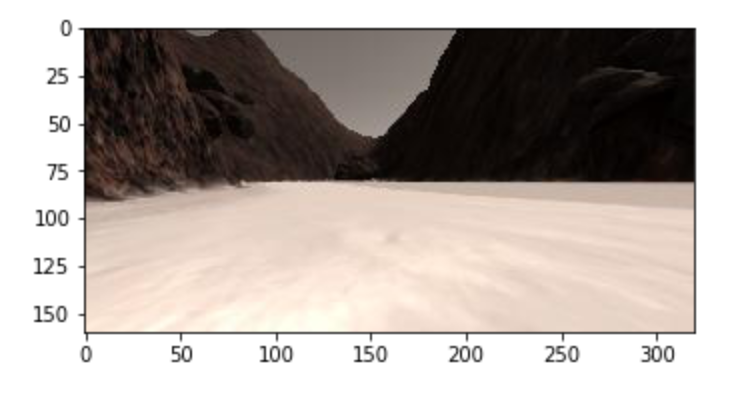
rock present in the image.

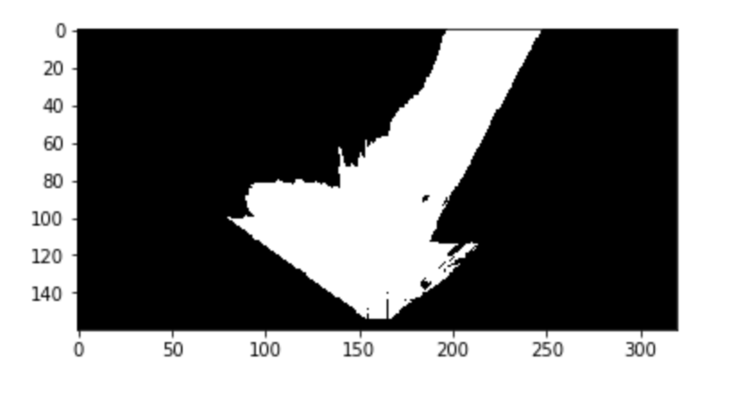




**OBSTACLE IDENTIFICATION**

Obstacles are designated as terrain that is not navigable in the simulation environment. The images that are recorded while the rover is driving are converted to a 'warped' image. The warped image is a top down view of the environment that is in front of the rover. The warped image goes through a process where the pixel values are manipulated in a manner that produces a black and white version of the warped image. The white pixels in this new image represent the navigable terrain in the map, the black pixels represent the 'obstacle' terrain. To accurately map the obstacle terrain, all of the pixel values in the black and white warped image have the 1 subtracted. This creates an image that is represented by 0's and - 1's, the absolute value is taken to convert the - 1's to 1's. This process of identifying obstacles in the environment is the reverse of identifying navigable terrain. Identifying obstacles in the environment is a requirement to accurately map of the environment. It is also worth noting that the image has a ‘mask’ applied to is to cut off parts of the image that are outside the field of view of the rover. This has the effect of placing black triangles in the lower left and right corners of the image.



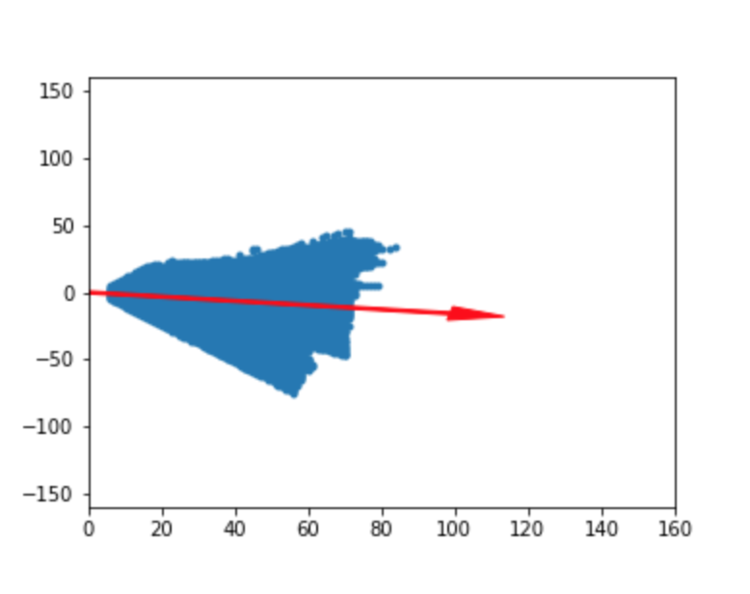


**PROCESS IMAGE FUNCTION**

The process image function contains functions and variables that create the map of the environment. To create the map of the environment, the original Rover camera images have to be processed to the point where they represent information about navigable terrain, rock samples, and obstacles in the same coordinate system as the ‘worldmap'. The first step is to take the original image and convert it to a top down view, so the original image looks like a 2-D image, this image is called ‘warped’. A color threshold is applied to the warped image, this color threshold makes navigable terrain is show up as white. This image is then converted into Rover coordinates, and the Rover coordinates are then converted to world coordinates. The conversion from one coordinate system to another is done by rotating and translating pixels. In addition to processing the images from the Rover to world coordinates. The pixels that represent navigable terrain are plotted plotted blue in the world, obstacles are plotted red, and rocks are plotted white on the world map.

**PERCEPTION AND DECISION STEP**

Perception The 'perception\_step' function takes as input an instance of a Rover object. The Rover object has a current image which is processed to create a map of navigable terrain, obstacle terrain, and rocks in the environment. This is achieved by using many of the functions and variables that are present in the 'process\_image' function. The 'perspect\_transform' function converts the rover camera image into a top down ‘warped’ image, this image is processed with the 'color\_threshold' function. Pixel values are manipulated to isolate navigable terrain, obstacles and rocks. The images that contain navigable terrain, obstacles, and rocks are then converted to rover coordinates which are then converted to world coordinates. Translation to world coordinates is necessary for creating a map of the environment. The pixels that represent navigable terrain increase the blue channel of the Rover.worldmap by 10, and the obstacle pixels increase the Rover.worldmap red channel pixels by 1. If there is a rock in the environment, the Rover.worlmap green channel is set to 255 where there are rocks. To increase the fidelity of the map, I included an if statement that requires the roll of the rover to be in a specific range for the mapping to take place. In addition to mapping the environment, information from the image is used in the decision function for controlling the rovers actions. The 'Rover.nav\_angles' variable is used in the decision step to control the steering angle of the rover. The amount of navigable space in front of the rover is additional information that is gained from the perception step. The navigable space in front of the rover determines the course of action for the rover.



**DECISION**

The 'decision\_step' function takes an instance of a Rover object as input and returns a Rover object as output. The 'decision\_step' function updates the rovers throttle, steering angle, and brake. These variables are dependent on the terrain that surrounds the rover. The condition for the rover to move forward depends on there being sufficient navigable terrain in front of the rover. The steering angle that the rover takes when the rover has sufficient navigable terrain in front of it depends on the 'Rover.nav\_angles' variable. The steering angle is shifted to the direction of the navigable terrain in front of the rover, the steering angle is clipped by +/- 15 degrees. When there is insufficient terrain in front of the rover, the rover stops and turns until there is sufficient navigable terrain on front of the rover. When there is sufficient terrain in front of the rover after stopping and turning, the rover begins to move forward again, and the steering resumes in the fashion that was previously described. The change in the variables, throttle, steer and brake is the actuation of the rover. The fundamental processes of robotics includes perception, decision making, and actuation. The perception and decision functions control the fundamental processes of the rover. I added an additional conditional statement to the 'decision\_step' function, the additional condition controls the 'Rover.send\_pickup' variable. This conditional statement controls the action of picking up rock samples.

**IMPROVEMENTS**

There are a couple of problems that I can see when the rover is in autonomous mode, the two problems include picking up rocks, and the rover getting stuck on obstacles. To improve the rover’s capability of picking up rocks, the decision step must take into consideration the angle of the rock in the environment relative to the rover. An improvement can be made if when a rock is present in the map, the steering angle depends on the position of the rock and not the navigable terrain. On occasion the rover will get ‘stuck’ with the throttle set to the top setting while the speed is zero. To counter this problem there can be a conditional statement in the decision step that detects this problem and puts the rover in reverse until there is sufficient navigable terrain in from of the rover to move forward.