Notebook Analysis

The notebook has been very useful companion to the lessons. It was possible to familiarize with the necessary steps involved in the perception phase from simple example images and processing of the information it contains for getting inside which will help as next decision taking.

The simulator has been used in training mode to record enough variety of camera images. An overall sample of 1042 images has been recorded. The images have been analyzed by adopting/improving the various basic functions of transformation and filtering in order to identify navigable terrain, obstacles and samples to be collected. Finally a movie has been generated.

Several functions have been defined in the notebook but realizing that they are generic, I have been creating 2 separate files nd_functions.py containing all of the functions finally used and nd_classes.py containing the definition of all the classes involved. This promotes reduction of the notebook code and increase reuse of the code by properly encapsulation into functions. The final image has been customized to reflect this work.

Autonomous Navigation and Mapping

The first step has been to run the already available scaffolding code by adding the different parts as from several hints and templates I could see in the Slack and chat with other students. This was useful to understand the mechanism of communication between the Simulator used in autonomous mode and my python code/process.

The simulator provides the data dictionary and functions lie the **perception_setp()**, **rover_update()**, and **decision_step()** are been called in order to process the data, get insight and finally make decision to driven the next steps of the rover.

The perception step requires reuse of the previous work done in the notebook. The same can be told for the **output_image()**, this can actually be customized based on personal flavors. The most interesting part is the decision step. Here the main strategy is to get all the pixels which make a navigable terrain. For each of them get the distance to the rover and the relative angle. The velocity vector is then oriented in line with the average angle of navigable terrain pixel which keeps the rover more or less in the middle of the navigable map.

The Sample collection is done by setting a bit upon detecting the presence of a sample in the map. In this case the mean angle of all the possible angles which makes the sample pixels have been used to drive the rover towards the sample. The distance to the sample is been used to decide by when the Breaks might be activated so to stop the rover in-front and enough closed to the sample so that it can be picked up. For this purpose the decision step has been modified to allow for this.

How to improve

At the moment of this write-up, I am driven from the deadline and submit the project. In an ideal case where more time is available and also for having even more fun (I have a full time job and a family with 3 kids) I would like to optimize the following:

Collection of sample.

The routine at the moment is far from being optimal; sometimes the rover would go over the sample by not realizing that it is close enough so to pick it up. Here I would consider a better control of the speed/breaking process.

Speed of exploration of the map.

Here it is required a better routine on the actuation… maybe a set of functions which take care of acceleration/deceleration in alignment with the distance to the targets (obstacles/samples/etc…)

Improved exploration algorithm

What I want is to avoid revisiting areas which has already been visited. Following the same side of the wall like in a labyrinth might be a good starting point. Another option is keeping track of the previous decisions made.

Improved accuracy

The thresholding of the various objects could be improved by better studying the various maps in the dataset. This would allow better recognition of obstacles, shadows and navigable ground.

Configuration:

Simulator: 1920x1200 (Fantastic), FPS: 28