Horizon-based localization for Udacity Challenge no. 3

Roboauto team

1 Intro

This document describes the approach of the RoboAuto team to the image-base localization for the Udacity Challenge.

2 How to

2.1 Building the project

The requirements to successfully run this project are a *docker*, *git* and a few easy steps:

1. Clone our git repository at *https://github.com/Roboauto/Udacity3*
2. Cd into Udacity3 folder in the cloned repository
3. Build a docker image (this could take several minutes): *docker build -t udacity docker*
4. To start the docker image for the first time: *xhost + ; docker run –volume=/tmp/.X11-unix:/tmp/.X11-unix –volume=/full/path/to/Udacity3:/home/robo –name=udacity udacity*

   else just: *xhost + ; docker start udacity*
5. Cd into ros folder
6. Everything is hopefully running and a new terminal should pop up. To run everything properly cd in this new terminal into ros folder and type *source setup.bash*
7. Build the project: *catkin_make_release*
8. Try if everything is running by starting roscore: *roslaunch ./src/udacity_launch/launch/bag_play.launch*

It is possible to split screen of the newly opened terminal (right click) and to run a different module in each screen. You need to use the "source setup.bash" command in each newly opened screen.

Typical setup with the commands below in different screens is:

- roslaunch ./src/udacity_launch/launch/bag_play.launch
- rviz
- rosrun slam slam
- rosrun validator validator
- rostopic pub (load map / localize)
- rosbag play

A few tips:

- a setup.bash file is located in the Udacity3/ros folder
- to add visualization in the rviz click "add" under "Displays" and choose from "select by topic" - /detection /horizont /slam
- default topics can be changed in the src/utils/include/utils/Topics.h file
- validator needs ros clock to be published (rosbag's –clock switch)

2.2 Creation of a map

The map is created by the ros node named *slam*. The command for running this module is *"rosrun slam slam"*. The default mode of the module is map creation so the only thing needed to start creating the map is just to play a bag file. The profile of the map should be seen as a green line in rviz. The default GPS topic is set to *"/fix"*. To save the map the command below is used:

```
rostopic pub /command std_msgs/String "data: 's path/to/mapfile'"
```

At this point the map should be saved.

2.3 Configuration

Some of the used parameters are stored in a configuration file and can be changed without the need for recompiling the whole project. The configuration file is located in the slam module and it is named "config.json". The parameters are loaded and printed on the screen at the start of the slam module.

Particle filter is a probabilistic algorithm so for it to be possible to replicate the achieved results one needs to use the same seed we used for the final run. The seed value is saved in the config file and by changing it to zero one can set the seed to random.

2.4 Visualizations in the rviz

The visualizations for rviz can be enabled in the configuration file. The loaded map is represented by a green line, a huge blue "arrow" denotes the actual position, the pink poles show boundaries of our dynamic augmented mode and the blue "columns" under the map represent the weighted density of the particles around that position.

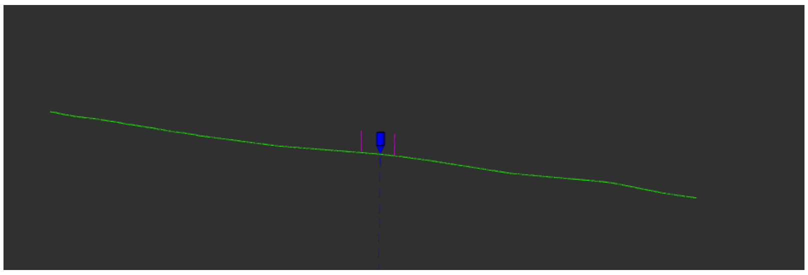

Figure 1: Visualization of a whole map.

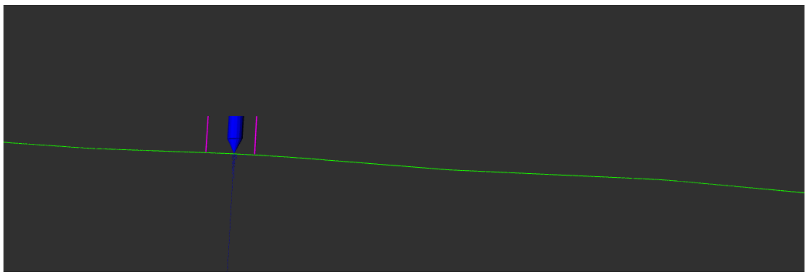

Figure 2: Closer look at density cluster, augmented mode boundaries and position marker.

2.5 Validating results

Localization is performed by the same module as map creation - slam. The command for running this module is *"rosrun slam slam"*. Further, it needs a map to localize on. The map loading is done via the following command:

```
rostopic pub /command std\_msgs/String "data: 'l path/to/mapfile'"
```

followed with the localization command (this command changes slam mode from map creation to localization mode).

```
rostopic pub /localize std_msgs/Empty
```

To easily validate the results, GPS coordinates are published on the "/slam/gps" topic. For the "real time" validation ros module validator was used (*"rosrun validator validator"*) which is capable of printing the actual RMS of each frame and the average RMS over the whole run. For this validation a BAG file with GPS coordinates is needed.

3 Overview

Based on the video quality and diverse light conditions horizon line was chosen as a core descriptor of each image.

Besides the horizon there were also some other descriptors. One of the most important is traffic light descriptor and vertical poles descriptor. With their usage we were able to make our localization even more precise. Other descriptors with lower influence are traffic sign detection and bridge detection. The traffic signs descriptor was not that effective because of a large percentage of false positives.

As localization method we used an augmented particle filter with several adjustments. The evaluation of particles was based on the descriptors mentioned above and will be described in detail in the following chapters.

4 Detections

4.1 Horizon

For obtaining the horizon line we tried two different methods. The first method was simple thresholding of an image's blue color channel. The second method was Otsu's dynamic thresholding. In different scenarios both of them proved to have advantages over the other. However, for validation dataset the Otsu's thresholding method was used. An image was then top-down scanned at each x-position and first pixel exceeding threshold was taken as the horizon value.

4.2 Inverse horizon

To further improve our horizon descriptor we also used a second level of a "horizon" line. In opposite to horizon line approach, bottom-up scanning of each x-position in combination with some morphological operations of the image. It is better seen at the image 3.

4.3 Traffic lights

A traffic light descriptor is useful addition when the horizon line around traffic light poles might heavily change (due the light conditions). The traffic lights score is configured in a way it only helps when lights are present in the evaluated position in the map and are also detected in the current frame.

Traffic lights were detected by combination of blob detection with color and shape filtering.

4.4 Vertical poles detection

Detection of the vertical poles is another descriptor used for finer position resolution. To prevent false positives only the poles that are of enough height are taken into account.

4.5 Sign detection

Detection of the signs was based on the HSV colorspace and shape filtering but did not seem to improve the results.

4.6 Bridge detection

Bridge detector locates the situations when the car is driving close to / under a bridge. Due to the fact that there is only one bridge in the validation dataset this descriptor also did not improve our results.

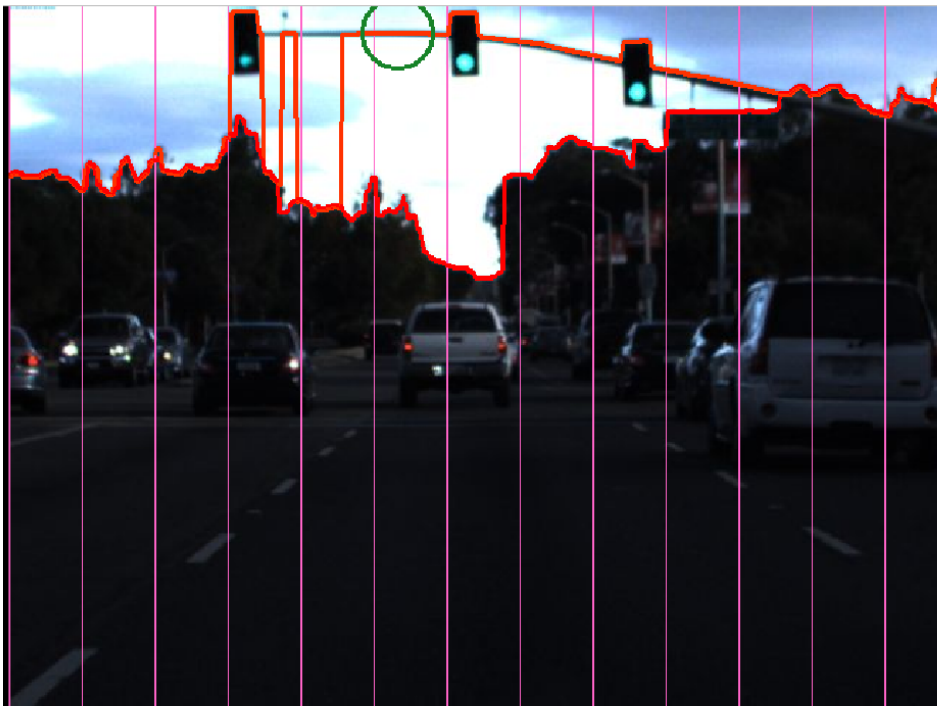

Figure 3: Two different red lines represent a horizon and an inverse horizon. It can also be seen that the horizon line is not copying the traffic light pole correctly.

5 Particle filter

As a localization method we chose augmented particle filter. The particle filter was further adjusted with a dynamic augmented mode (see chap. 5.3).

5.1 Motion estimation

An estimation of a distance change is needed for a proper particle movement step of the particle filter. For this purpose we utilized a feed forward neural network in a combination with the dense optical flow algorithm. Calculated values from two consecutive images are then fed as an input into the neural network. The distance between those two frames is then obtained as an output of the neural network.

5.2 Horizon comparison

Due to the fact that horizon is a core descriptor used for the evaluation of a position and it was necessary to tweak the comparison to be robust enough but also to be able to distinguish between roughly similar horizons. The difference between the horizons serves as the base score for the resampling step of the particle filter.

The first step in a horizon comparison is undoubtedly finding correct overlap of two horizon lines. A horizontal shift is not calculated directly but several possible shifts are generated and the one with the best score (lowest difference) is then taken. The possible horizontal shifts are generated based on the distance between the biggest changes (differences) in the horizon line. For example:

```
H1  = [4,5,4,1,2,6,7]  H2  = [4,1,1,2,6,5,7]
H1d = [1,-1,-3,1,4,1]  H2d = [-3,0,1,3,-1,2]
```

The biggest changes (differences) in the horizon line H1 are at 3rd and 5th position. For the second horizon line those are at the 1st and 4th position. For H1 the possible horizontal shifts are by 2 or by 1 to the left.

There was also an effort to utilize the Fourier transformation to find the best horizontal shift but the "correctness" seemed to be too "steep" for our purpose. Similar horizons where shifted just fine but little more different horizons got low scores and that did not seem to work well with the particle filter.

By having a horizontal shift it is easier to determine the vertical shift. The idea is to compensate a systematical vertical shift between the horizons. This is done by simply averaging the difference between values in the same position in both horizons. The values at the very top of the image (if the horizon line is ”above” the picture) are excluded.

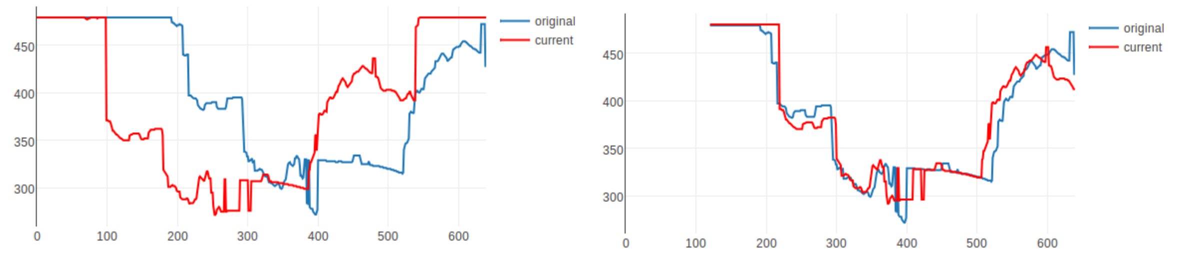

Figure 4: The example overlap of the horizons located near each other. The original horizon line (blue) is saved in the map and the current horizon line (red) is taken from the current camera frame.

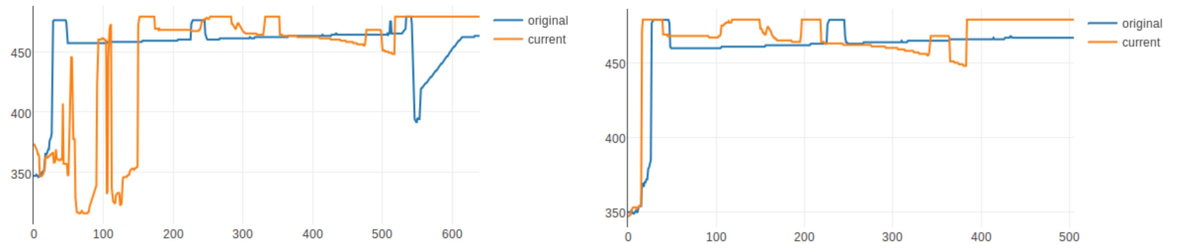

Figure 5: The example overlap of the deceiving horizons from different locations. In this picture it is possible to see the frequent deceiving situation with the contour of the traffic light pole.

At this point the horizons are overlapped and the next step is their comparison. The total difference of the horizons consists of two components. The first one is a pure average distance between the horizons which serves as rough estimation of the horizons similarity. The second component aims at the finer distinction of the horizons. It calculates the percentage of the horizons overlap that is closer than some smaller threshold (and is not ”above” the image). The total difference of two horizons is then a sum of those two components.

5.3 Dynamic augmented mode

Another very important aspect of our solution is a dynamic augmented mode for the particle filter. A standard augmented particle filter adds some random particles across the whole map. This prevents the particle filter from converging into a local extreme without the possibility to later find the correct position.

While this helps it also may, in some cases, pull the correct position somewhere else because of the temporary better score of the new position. To prevent those jumps and to accommodate the history of the positions we introduced the dynamic augmented mode.

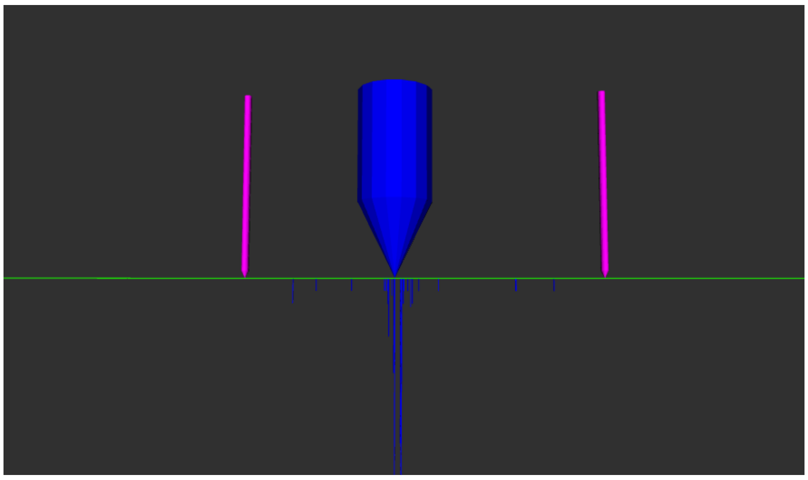

Figure 6: Close look at the visualizations of the dynamic augmented mode borders and position. All the new particle are generated only between the pink poles.

The dynamic augmented mode reduces or increases the space, where are the new particles were added based on the sureness of our position. The more certain is the position the smaller is the area of augmented mode around that position. The sureness of a position is based on the horizon score. There is also a mechanism that limits the increase of boundaries after longer time with a sure position. Thus it prevents the augmented mode from increasing the boundaries rapidly in the situations where the horizon is not detected properly and to stabilize the position over a longer period of time.

5.4 Position selection

To calculate the best position a weighted density of particle's surroundings is used. For each position with a particle the position score is calculated from a few surrounding meters. Each particle contributes to the score with a value of $\frac{1}{\delta}$, where δ is the distance to the calculated position (see eq. 1 and 2).

$$w_p(\delta) = \begin{cases} 0 & \delta \geq d_{max} \\ \frac{1}{\delta} & \delta < d_{max} \end{cases} \tag{1}$$

$$f_{pos}(\theta_{pos}) = \sum_{n=0}^{N} f_p(\|\theta_{pos} - \theta_n\|) \tag{2}$$

Where $w_p()$ is the distance-based weight function, δ is the distance between two positions, δ_{max} is the maximal distance, $f_{pos}()$ is the score function of the position θ_{pos}, N is the number of all the particles and θ_n is the position of the n_{th} particle.

6 Workflow

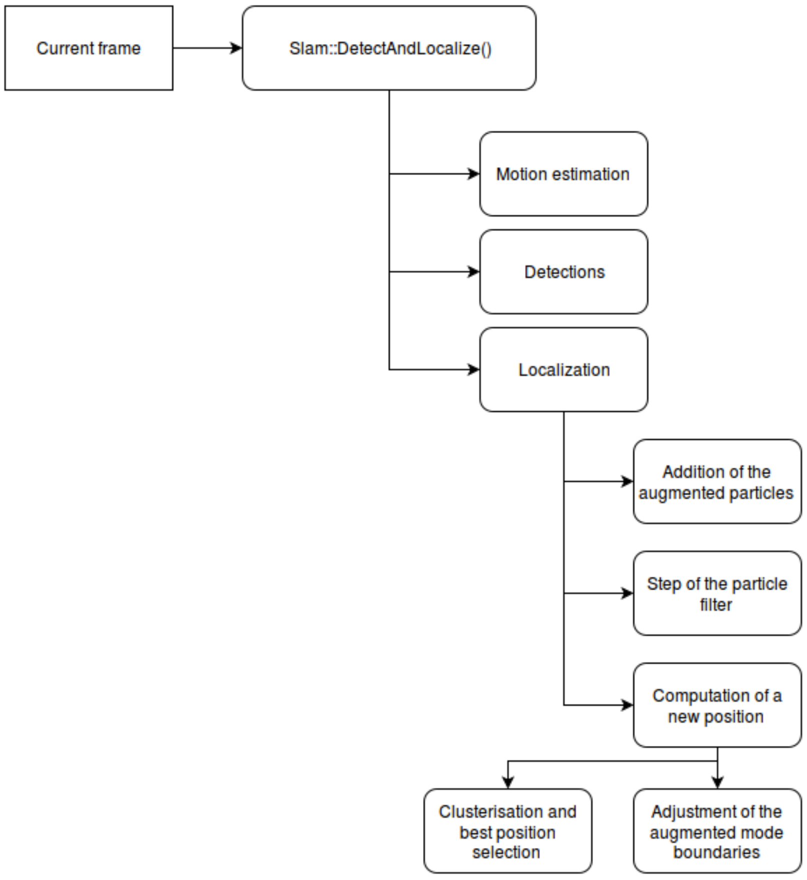
Current frame
Slam::DetectAndLocalize()
Motion estimation
Detections
Localization
Addition of the augmented particles
Step of the particle filter
Computation of a new position
Clusterisation and best position selection
Adjustment of the augmented mode boundaries

Figure 7: Diagram of a flow of actions in response to the new incoming image.