ESE 680 Autonomous Racing

Lab 5: Scan_matching - PLICP method - Localization

Author: Baihong Zeng   Date: 10/15/2019

(b) A short video of the simulation demonstrating the odometry estimation.

Answer:

A short video showing the pose is uploaded to (missing link).

(c) A PDF writeup describing your approach and the roadblocks you encountered in the development.

Answer:

I mostly follow the instructions given in the lab instruction pdf to finish the updatetransform() function. One minor problem I encountered is that the output theta may be larger than 2pi, so I need to rectify the range of theta. Also, greatest_real_root() function may produce incorrect roots when the coefficient for the fourth order variable is zero causing "lambda" to be 0 at all time. I calculated the root by using functions mentioned in https://github.com/opencv/opencv/blob/master/modules/calib3d/src/polynom_solver.cpp to correct this issue.

getCorrespondence() function is based on the pseudo-code in Censi's paper. Everything works perfect except a few bugs. I was trapped in the while loop and got nan value in the first place. Trapping in the while loop is because the initial value of "last_dist_up" and "last_dist_down" are the same "DBL_MAX" which causes some weird problems. I fixed this problem by initializing them to be some large values like 100,000 instead of "DBL_MAX".

Nan value usually occurs when accessing the out of range index of a vector. I fixed this problem by adding an if statement to limit its value between boundaries.

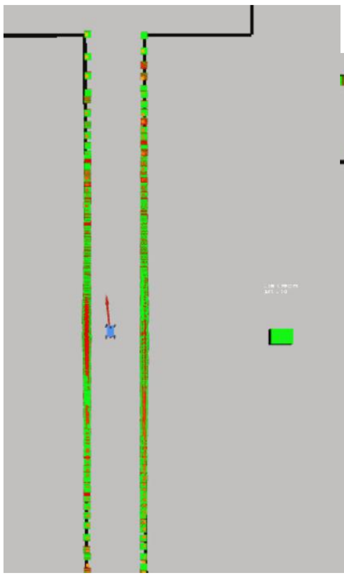
One other problem I found is that the scan_matching/pose is marked on the laser frame instead of the global map frame. However, from the experiment and my understanding of the code, it should be marked on the map frame since the odom is marked on the global map frame as well. Thus, I changed this parameter in the publishPos().

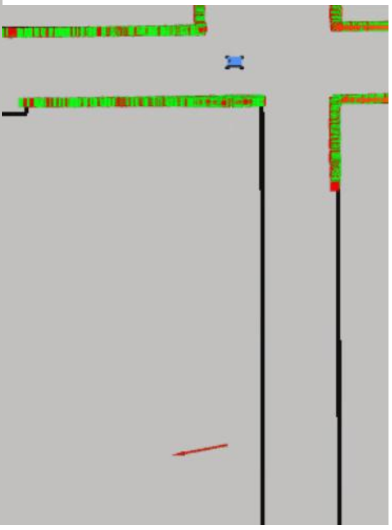
```
msg.header.frame_id = "map";
```

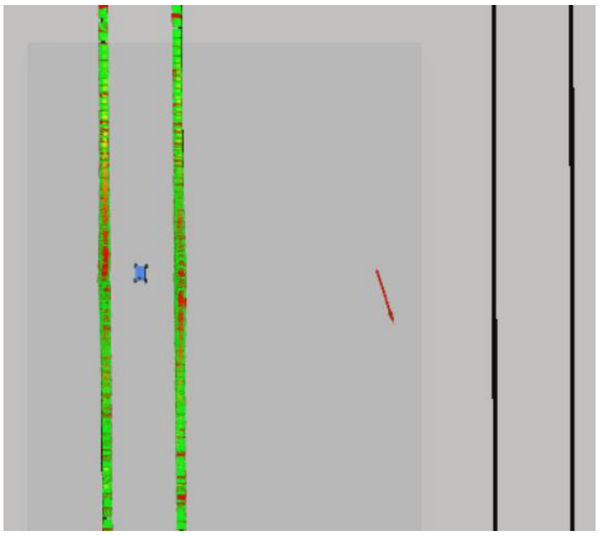
In this way, if this algorithm works perfectly and localizes itself well, the red pose arrow should have the same displacement and orientation as the car. However, in the simulation, we can tell that there are still limitations of this algorithm which will be illustrated in the next section.

(d) Writeup should include plots of measurements of performance and analysis of results

Overall, the accuracy for measuring orientation is acceptable considering the speed of car is not slow while the displacement measurement has a large gap between the real value and the measured value.

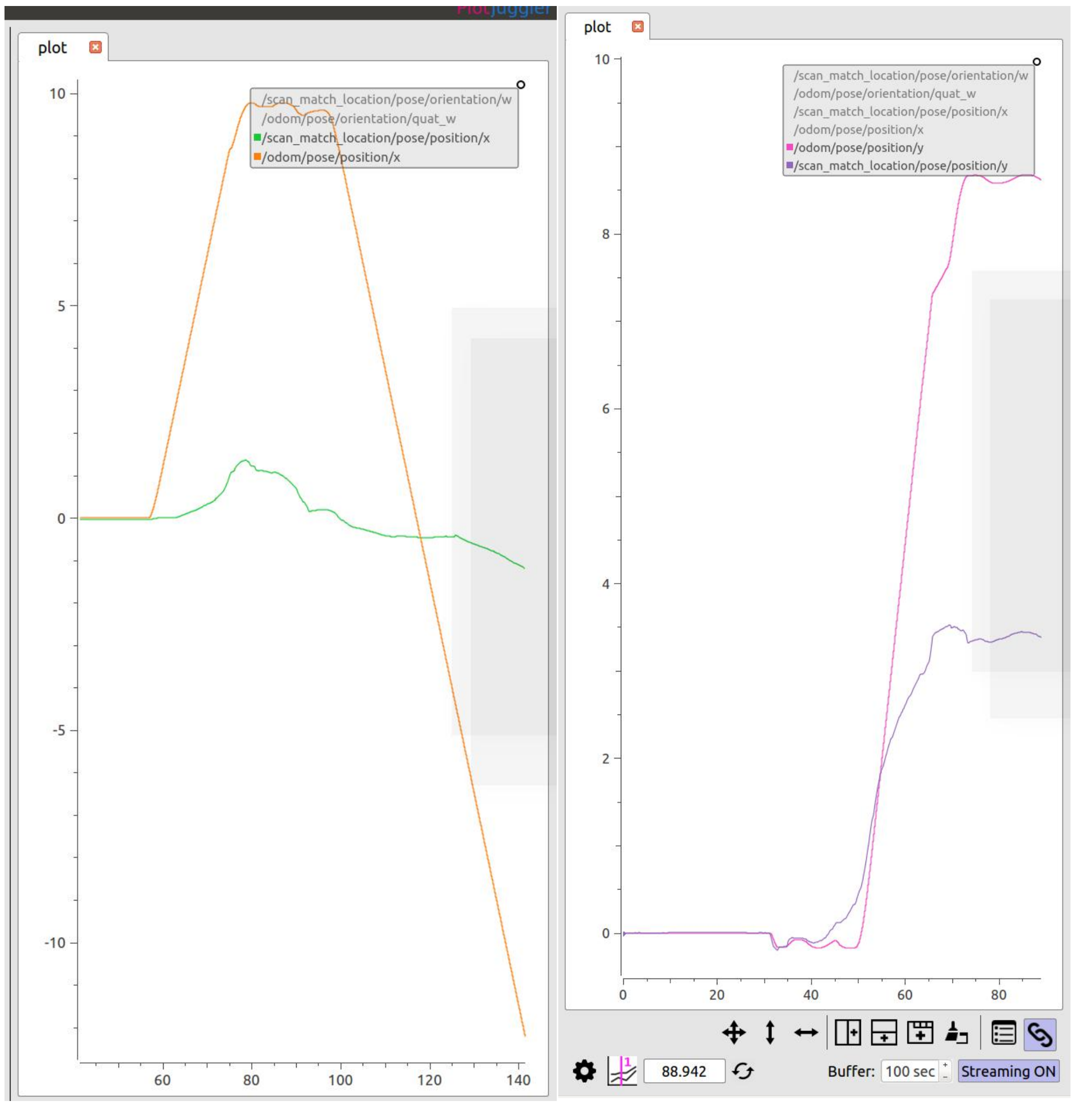




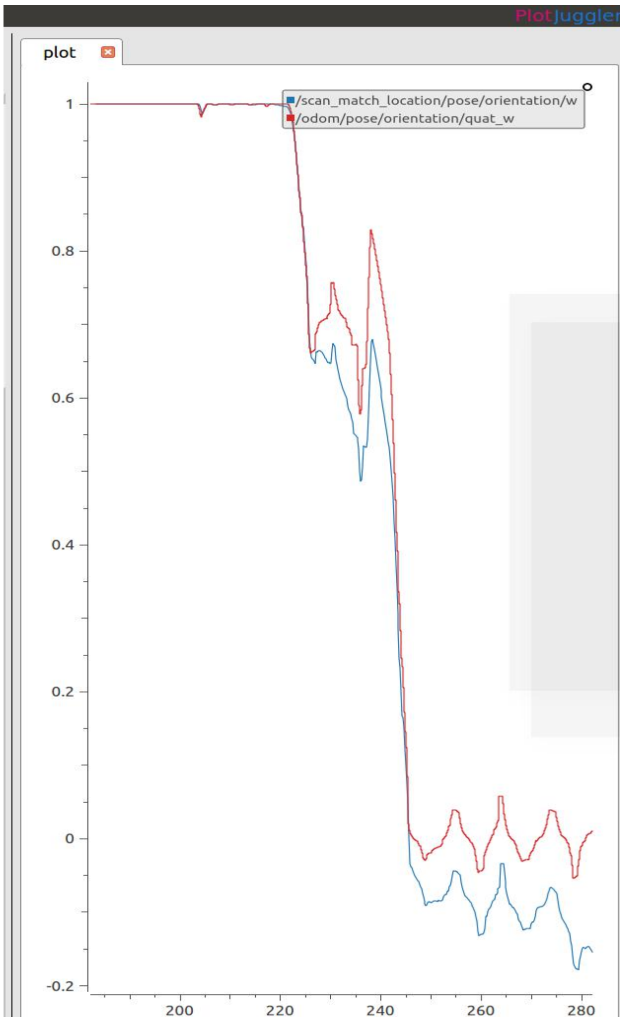


The three screenshot above show the visualized pose and marker of the scan_matching topic. The green dots overlap with the laser scan reading well meaning that the getCorrespondence() function generates correct correspondence. However, the red arrow has a similar orientation with the car but the displacement is way off. This is because when the car running in the straight line, the wall is too smooth and the algorithm can’t catch much feature from its reading to calculate its displacement. Thus, the algorithm assumes the car is not moving on the straight hall way.

The following two graphs shows the comparison of the reading between odom and scan_matching respect to displacement x and y on the left, and right, respectively. In the left graph, the reading is around 0 since the car has not moved yet. The car starts going forward from 60s and the discrepancy gets larger since the algorithm does not think the car is moving but it actually is. The measured displacement matches the real one when the car turns the corner since the corner provides many features for the algorithm to localize.



The following figure shows the orientation reading. It is obvious that the algorithm provides a roughly accurate reading and they both have the same trend. Some static error is still observed after 240s. Overall, the discrepancy is small and acceptable.



The following six parameters remain 0 during all time.
scan_match_location/pose/position/z; odom/pose/position/z;
scan_match_location/orientation/x; /odom/orientation/quat_x;
Scan_match_location/orientation/z; /odom/orientation/quat_z;

There are two variables "MAX_ITER" and "number_iter" to be changed to vary the accuracy of the correspondence and the speed of the program. The increment of "MAX_ITER" decreases the speed heavily so I only set it to be 2. While the increment of "number_iter" does not slow down the program that much and it can effectively rectify the measurement error, I set it to be 3.

(e) Any other helper function files that you use.

Answer:

I used functions from https://github.com/opencv/opencv/blob/master/modules/calib3d/src/polynom_solver.cpp to calculate the real root for quartic function.