SLAM Homework 1 - ICP

Runze Yuan (yuanrz@shanghaitech.edu.cn)

October 6, 2021

1. Algorithm Pipeline

The goal of this homework is to compute the trajectory of the car. In order to do that, we compute the relative pose between each two adjacent frames using ICP. The pipeline of the algorithm is as follow.

1. Every time we read two lines of data into memory, which reduces the memory cost.
2. The resolution of each frame is $2 * \pi/360$. Transform the distance into points using

$$x = \cos(resolution * i) * distance$$
$$y = \sin(resolution * i) * distance$$

   in which i is the index of the data and *distance* is the value of the data.

3. Using ICP to calculate the relative pose from curFrame to lastFrame. ${}^{last}_{cur}T$
   (a) For every point in curFrame, using brute force searching, search the nearest neighbour in lastFrame and create a correspondence set C = $\{P_{cur}, P_{last}\}$. In this step, we apply rejection in which the correspondence whose distance is larger than 0.5 will be rejected. In this scenariobrute force searching is enough, because the size of data is small.
   (b) Calculate the center of P_{cur} and P_{last}. Then create decentralized point sets for P_{cur} and P_{last} using its center.
   (c) For every point in decentralized set, create the covariance matrix and sum together.

$$\Sigma += p_{last} * p_{cur}^T$$

   (d) Apply SVD decomposition on Σ and the result is

$$R = U * V^T$$
$$t = p_{lastCenter} - R * p_{curCenter}$$

   Here we need to consider R may not be in right hand coordinate system. So if the determinate of R is less than zero, make $R := -R$
   (e) Determine the convergence. The convergence criteria is that $theta < 10e-5, norm(t) < 10e-3$. If the algorithm is not converge or reach to the max iteration, go to (a) using the transformed point cloud.
4. Multiply the transformation T between each adjacent frames using chaining rule. We can get the trajectory of the robot.

2. Usage

1. Dependency:

(a) ROS (version:noetic) (for visualization)

(b) Eigen (for computation)

2. Compile: Create a directory named **icp_ws**, put the src directory into it. Then run **catkin_make -DCMAKE_BUILD_TYPE=Release**

3. Run: **source devel/setup.bash** and **rosrun icp icp_node path_to_data.txt**

   For visualization, open rviz and add a path visualization , change the name of the topic to **"path"**

3. Evaluation

In our case, there is no need to do outlier filtering because there is no noise in data.txt.

The performance is shown below

Table 1: performance table

	time cost	rotation error(degree)	translation norm error
data	1.32813	169.596	2.41848
data_noisy	2.47748	-104.532	1.93543

The path visualization is shown as below

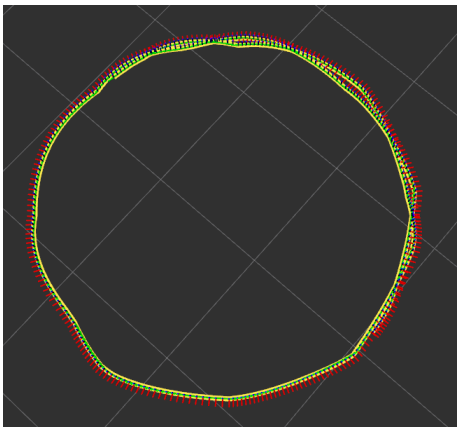

Figure 1: data

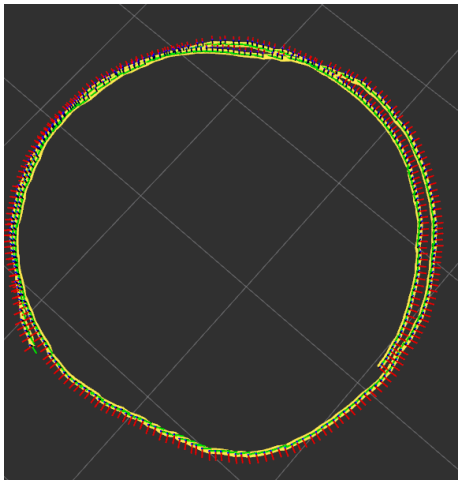

Figure 2: data_noisy

Because the accumulated icp error, there is a large drift in the path.

In this scenario, the drift can be easily reduced to zero if we estimate four line and use the four lines to do icp to solve the pose because there is no noise in the data. But we did not implement it in this homework.

We spend more time on data containing noise because the noise make the convergence harder. The error of noisy data seems smaller than non-noisy data, the reason can be that the noise makes the searching correct accidently.