Homework 4

Robotics 2020 - ShanghaiTech University

1 Introduction

In this HW you will implement a particle filter for localization. You will use a gazebo world without obstacles, just consisting of many AprilTags placed in the sky. Your robot has an up-looking camera and can observe the tags - with a certain noise. AprilTags can appear more than once in the world. You will be provided with a map of the poses of all tags. All this code will be provided by us.

The framework for HW 4 is provided here: https://star-center.shanghaitech.edu.cn/gitlab/schwerti/robotics2020_hw4_framework

Ultimately your task is to autonomously drive the simulated robot to a certain position (within one meter radius). That position is given in the coordinate system of the AprilTag map. So you will need to localize within that map. Since tags appear multiple times you will need to drive around to finally estimate your position. The steps of your implementation should be the following:

1. Read this document and "readme.md" first
2. Parse the map provided ("src/homework/src/tags_pose_map.txt") to your own map data
3. Employ some strategy to (autonomously) drive the robot around (e.g. random walk). The initial pose of robot is initialized randomly for your tests - we will put a specific pose when grading your HW.
4. Detect AprilTags as you drive around (provided by us - we add artificial noise - do not mess with that! The noise parameters will be provided to you.)
5. Implement a particle filter to localize yourself (no other algorithm for localization is allowed). You can use the noisy odometry.
6. Visualize your particles in rviz (code snipped is provided)
7. Detect when your localization estimate as converged to one solution
8. Then drive to the goal coordinates and stop
9. Send an empty ROS message to the topic "/finished"

2 Details

You implement a ROS package named "homework". The node is also called "hw4", compiled from "hw4.cpp". Provide all files needed for compilation. The node listens to messages of type "tf2_msgs::TFMessage" on the topic "/tags" for the AprilTag detections. They are provided in the frame "robot". You also listen to messages of type "tf2_msgs::TFMessage" on topic "/odom". Finally, you will get the goal point on the topic "/goal" of message type "tf2_msgs::TFMessage". Do not use any other input message.

You output messages of type "twist_mux" on topic "cmd_vel" to control the robot. Once you reached the goal point publish the empty message of type "std_msgs/Empty" to the topic "/finished".

2.1 Map Format

The map of AprilTags is provided in "homework/src/tags_pose_map.txt". The "txt" file contains the tag number about each AprilTag and the pose about them. The position precision is 0.1m.

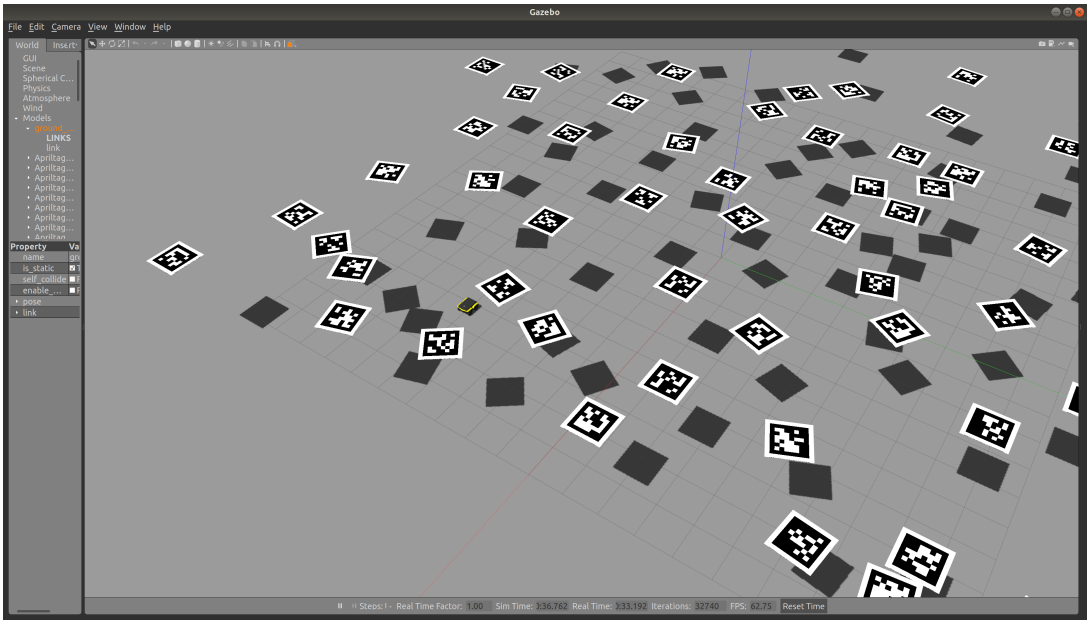

Figure 1: The simulation environment.

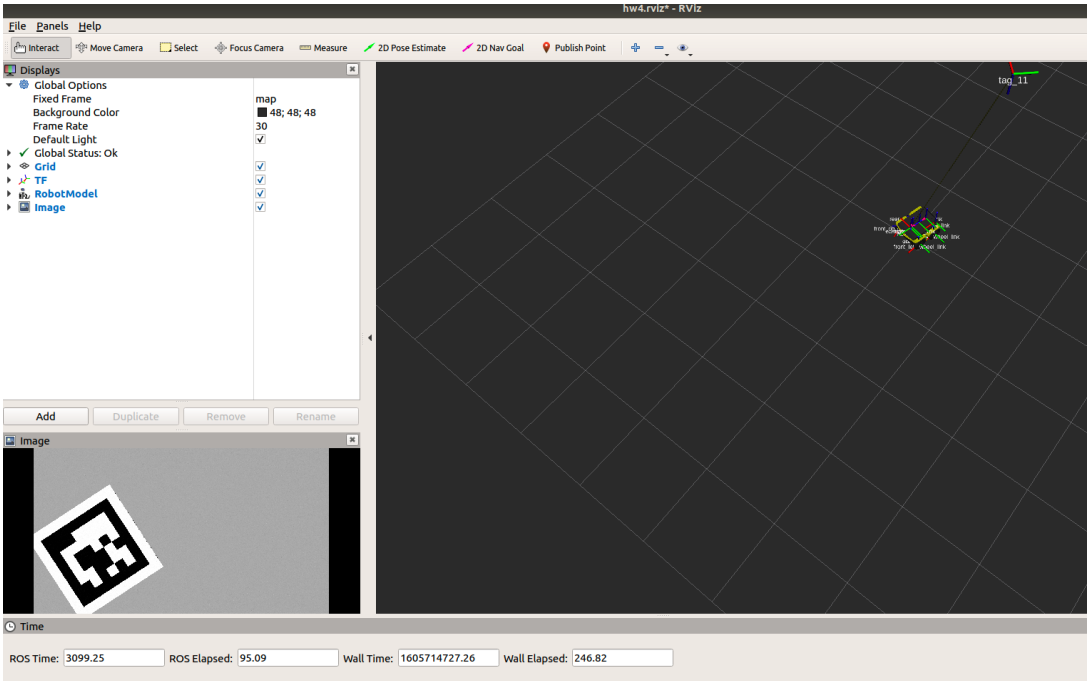

Figure 2: The rviz interface.

2.2 How to use the Simulator

In gazebo, you can get our simulation world about this homework, which is shown in Fig. 1. You can also find you jackal robot in the simulation environment, and the pose of the robot will be changed simultaneously when you control it.

2.3 RViz

You can find the transfrom information about some joints (not all joints because there may be some conflicts about the same tags) and the image captured by camera on the robot, which is shown in Fig. 2.

2.4 Noise

Those are the noise parameters you should assume in your program ("src/preprocessing/preprocessing.py"):
AprilTag pose noise: tags_t_noise (0.1 default about translation), tags_r_noise (0.05 default about rotation)
Odometry noise: odom_noise (0.1 default about translation), odom_r_noise (0.05 default about rotation)

2.5 Implement the particle filter (70%)

This task is worth 70% of the HW 4 points.

2.6 Provide Screen Shots (20%)

Additionally from providing your particle filter implementation in hw4.cpp, we require you to provide two screenshots from the RViz particle filter visualization. In that visualization you need to scale the size of the arrows according to the weight you calculate for each particle (bigger is better) - make it look nice...

Provide one screen shot from the beginning of the run, where we can nicely see several clusters. Provide another screen shot from the end of the run, when the pose estimate has converged. Put both screen shots in a "report.pdf".

2.7 Academic Honesty

This is a homework - it is not group work! Everybody has to do it by themselves.

3 Submission (10%)

Your submission consists of just the folder called "homework" ina your gitlab repo (yes, just "homework" - there is now "hw4" or the number 4 - we know which one it is") which contains "src/hw4.cpp", "CMakeLists.txt" and "package.xml". Apart from them, you should also submit a simple report called "report.pdf" in the folder "homework". Failure to comply to these instructions loose the 10% for the submission.