Assignment 2: Pose to SLAM

February 6, 2020

This assignment will be divided into two parts, the rst aim for you to get familiar with factor graphs and going to Poses to SLAM. Then you will have to implement this with the fetch robot in the playground environment. We have attached a presentation that you will use to answer this assignment. In the first slide of the presentation, put your name and GTID. Then, on each slide, answer the respective questions. Please don't create new slides, restrain yourselves to the space given but you can make the size of the text smaller if needed. As a deliverable, you will submit the presentation as a pdf to canvas.

Part I

Colab

1 Working with Colab

We want you to introduce you to GTSAM and work with factor graphs for pose SLAM. For this Part, we will give you a premade code which you will have to complete to make it work. As you complete the dierent stages of the code, we will ask you to reect on what is happening.

For this, we want to introduce you to Google Colab, which is an easy online tool to write and run python les. To do this, we recommend that you

1. Go to `https://colab.research.google.com/notebooks/intro.ipynb`
2. File -> Create a new Python 3 notebook
3. On the new page, in the first line install: $!pip install gtsam
4. Then press "insert code cell" (it has the sign + Code) and paste the rest of the code provided in the attached le of this homework.

NOTE: We leave the intallation of GTSAM in its own cell block, so we only need to do it once, an then we can run our code without installing it each time.

Now you can run each code cell to see the results.

2 Filling out Pose2SLAM

The provided code is an example that has been modied a bit for this homework. To create our graph, we will be working with sections 2a, 2b, and 2c (you might also need to change things in other sections of the code to avoid error in plotting as we increase or decrease the number of factors).

First, get familiar with the code and the plots that we are showing. The nal plot shows the robot in the dierent positions, where the red line is the x-axis and the green is the y-axis, while the circle being the uncertainty at the given pose.

- Exercise 1: Take a screenshot of the plot and put it on the PPT file. Give a brief explanation on why the circles are getting bigger.

From the second state of the robot at the pose (2,0,0) the robot gets back to this position at the seventh state. This is done in the code 2c, but the code needs to be xed to connect node 6 to the previous node 2 (and not create node 7)

- Exercise 2: Make the loop closure work. Take a screen and give a brief explanation on what happen with the covariance of each state.

Now, remove the prior from the factor graph

- Exercise 3: Comment what happens when you don't have a prior for the initial state in the graph and relate it to the gauge freedom.

Part II
Simulation

Now we will join the factor graphs with the simulated robot. We will give you the guidelines to create a le that subscribes to the odometry data from the robot and you will have to write the code that creates the nodes and close the loop when needed.

1. To start, you will need to create a package for assignment 2 as we did for assignment 1
2. You will create a le that will subscribe to the odometry data from the simulated robot (A simple subscriber example is attached to this assignment)

Create a subscriber

As mentioned in the previous assignment, ROS connects a node that is publishing some information/topics (in this case, the fetch robot is publishing its

odometry data) and other nodes that are subscribing to these topics to perform some computation.

In the following link

`http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29`

you will nd how a subscriber works. This subscriber has been modied in the attached le to already be subscribed to the odometry data. Mainly, you subscribe to the topic odometry and each time the robots publish its odometry data, your code will execute the function callback. You will have to add the poseSLAM code into this function to create the nodes (when the robot has move enough, create a new node and check for loop closure)

1. Exercise 4:Show the plot of the factor graph just like in part 1 of the assignment for three different moments in time, as you teleop the robot around. At least one of the plots needs to have a loop closure.

2. Exercise 5: Exercise 5: Give a brief explanation/diagram of what is the flow of the program. Include what are the relevant topics that are being published, who subscribes to them, and a high level explanation how you use this information to create your factor graph and find loop closures.

Helpfull information

- We want you to implement the factor graph with the simulated robot. We don't want you to create a controller for the robot, thus you can just keyboard teleoperate the robot around the worlds.

- You can use the fetch_gazebo simulation.launch and use that world to drive your robot around

- We are subscribing to odometry. Odometry message has the 3d position (msg.pose.pose), orientation as a quaternion (msg.pose.orientation), linear velocity (msg.twist.linear) and angular velocity (msg.twist.linear).