Home Work 2

Mobile Robotics 2015 - ShanghaiTech University

1 GIT (5%)

Take a look at http://gitolite.com/gitolite/glssh.html and use `ssh-keygen` to generate a ssh key pair **if you don't already have one**. Send me your public key `id_rsa.pub` via E-Mail! Do this till Friday, Nov 27 10:00 **am**!

2 ROS Tutorials (10%)

Work through the tutorials on http://wiki.ros.org/ROS/Tutorials/ till number 13. You do not need to do number 12 (Python).

3 Preparation

Make a new workspace, e.g. hw2_ws:
```
mkdir -p  /hw2_ws/src
cd  /hw2_ws/src
```
Get the files for the homework:
```
wget http://robotics.shanghaitech.edu.cn/sites/default/files/files/hw2_files.tgz
```
Unpack the files:
```
tar xfvz hw2_files.tgz
```
Make the workspace:
```
cd ..
catkin_make
```

Do not forget to always source "setup.bash" in the "devel" folder of your workspace for every new shell that you open!

```
source devel/setup.bash
```

If everything went well up to here (i.e. no errors during compilation) you are ready to proceed with the actual homework. If there was a problem, try your best to solve it - maybe ask you fellow students. If nothing helps you can contact Prof. Schwertfeger for help (come by the office or write an email)!

4 Calculate Global Pose Estimate (40%)

Your task is to calculate the global pose estimate of a robot in 2D. Input are 2D odometry estimates published on topic "/odometry2D". The odometry as well as the pose are of message type "mobile_robotics_hw_2/Pose2DStamped", that can be found in the downloaded code. Looking at Pose2DStamped.msg you will see that it consists of a "Header" and a "geometry_msgs/Pose2D". You can find out more about the messages by executing:
```
rosmsg show mobile_robotics_hw_2/Pose2DStamped
```

Your task is to use 2D geometry to calculate an estimate of the global position of the robot based on the odometry input. Your global reference frame should have the name "global". The first odometry estimate received should be relative to this global frame.

Make sure to publish the pose with the correct frame_id and the timestmap of the received odometry message.

File "mobile_robotics_hw_2/src/odometry.cpp" has everything you need already prepared - you just need to do the calculation and fill the corresponding message! You can add this around line 19.
`gedit mobile_robotics_hw_2/src/odometry.cpp &`

For every line of code that you write, add one line of comments (starting with "//"), describing what the line of code does.

In the very first line, add a comment (starts with //) and the following information: Your name (Chinese and Pinyin), your student id and your email. Also include the course name and the number of the homework (number 2).

Compile by going into the root of your workspace and execute "catkin_make".

Test by running:
`rosrun mobile_robotics_hw_2 mobile_robotics_hw_2_node`

See the "readme.txt" on how to send a test odometry message and how to replay and record the bag files.

Create a bagfile named "poses.bag" from the content of the **whole** "odometry.bag". So you have to make sure that you start your node first, then start the rosbag record, and only then start the rosbag play!

4.1 Purely Optional

If you are bored and want to play around with more random odometry values, you can start the node "random_odom2d". This will publish random messages of type "mobile_robotics_hw_2/Pose2DStamped" on the topic "/odometry2D". You can take a look at "random_odom2d/src/random_odom2D.cpp" for more details.

5 Calculate and Publish Speed (20%)

Additionally to the pose, calculate the speed of the robot in units per second (a unit in ros is by default meter). Publish the speed of type "std_msgs/Float64" under the topic name "/speed". Add all the needed code to "odometry.cpp". Record a bagfile called "speed.bag".

6 Display Pose and Path in Rviz (20%)

Rviz is a ROS program to display robot data. It does not handle our 2D pose message, so a converter to 3D pose and path message is provided. Run: `rosrun pose2d_to_3d pose2D_to_3D_node`
This will publish a "geometry_msgs/PoseStamped" and a "nav_msgs/Path" on the topics "pose3D" and "path3D", respectively. It listens to the output of your program on "pose2D".

You can run your program live together with the "pose2D_to_3D_node" or you can play your recorded "poses.bag".

Start rviz with:
`rosrun rviz rviz`
Add a path display and a pose display and set the according topics in those displays. You also need to set the Fixed Frame in Global Options on the left to "global". Also try what happens when you set Fixed Frame to "pose2D_to_3D_node".

Show the complete path and the last pose of "odometry.bag" in rviz, using "global" as fixed frame, and make screen shot. Make sure to also show (parts of) the relevant terminals in that screen shot. Save the screen shot in "mobile_robotics_hw_2/" under the file name "rviz.png".

7 Submission (5%)

Your submission consists of the compressed folder "mobile_robotics_hw_2/": "hw2_name.tgz" (replace "name" with your name). See the readme.txt on how to create that file. The following files are important:

- Everything that was in the original "mobile_robotics_hw_2/" folder - The code must compile and be the code you used to create the .bag file!
- Especially "odometry.cpp", including your code and the comments.
- "poses.bag" created by you!
- "speed.bag" created by you!
- "rviz.png": the screen-shot from rviz.

Send the .tgz to soerensch@shanghaitech.edu.cn by Monday, December 4th, 10:00 pm. The subject should have the follow format: `[Mobile Robotics][Homework 2][Pinyin Name]`