FIRST ROBOTICS PROJECT 2022

Name: Christian Confalonieri

Description of the files inside the archive

The archive contains 2 packages:

1. **project1**, containing:
    - **cfg/dynamic_rec.cfg:** cfg file for dynamic reconfiguration
    - **launch/project1.launch:** the launch file that starts everything
    - **msg/WheelSpeed.msg:** the custom message where the wheel speeds, computed using the linear and angular speeds present in cmd_vel, are published
    - **src/fw_omnidirectional_robot_odometry.cpp:** the c++ file where all actions are performed
    - **srv/SetPose.srv:** the service that resets the odometry to any given pose (x, y, θ)
    - **CMakeLists.txt**
    - **package.xml**
2. **project1_calibration**, containing:
    - **launch/project1_calibration.launch:** the launch file that starts all useful calculations for calibration
    - **src/calibration.cpp:** the c++ file where all parameters are estimated
    - **CMakeLists.txt**
    - **package.xml**

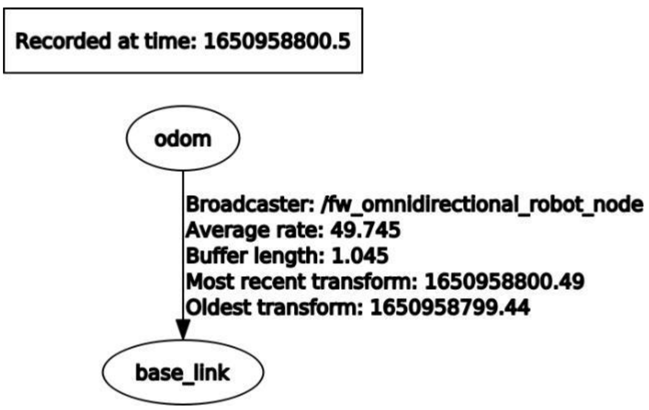
It also contains an .md file (with the related TF tree .png) and the related .pdf file where I transcribed the instructions.

ROS parameters

I used 3 parameters to reset the pose:

1. **initial_x:** value assigned to x (at startup it is set to 0)
2. **initial_y:** value assigned to y (at startup it is set to 0)
3. **initial_theta:** value assigned to θ (at startup it is set to 0)

Structure of the TF tree



Structure of WheelSpeed.msg

> Header header
> float64 rpm_fl
> float64 rpm_fr
> float64 rpm_rr
> float64 rpm_rl

Usage

I tested my project on ubuntu 18.04 with ROS Melodic

Prerequisites

```
mkdir -p ~/your_workspace/src
cd ~/your_workspace/src
```

put project1 and project1_calibration folders here.

Compile

```
cd ~/your_workspace
catkin_make
```

Running

```
roslaunch project1 project1.launch
```

or

```
roslaunch project1_calibration project1_calibration.launch
```

After running it you can start the desired bag.

You can run

```
rosrun rqt_reconfigure rqt_reconfigure
```

to dynamically reconfigure the integration method.

You can run

```
rosservice call set_pose "x: a
y: b
theta: c"
```

to reset the odometry to a specified pose (a,b,c).

List of Math Equations Used

Forward Kinematics Equations:

$$V_x = \frac{R}{4}(u_{fl} + u_{fr} + u_{rr} + u_{rl})$$

$$V_y = \frac{R}{4}(-u_{fl} + u_{fr} + u_{rr} - u_{rl})$$

$$\omega = \frac{R}{4(l+w)}(-u_{fl} + u_{fr} - u_{rr} + u_{rl})$$

Inverse Kinematic Equations:

$$u_{fl} = \frac{1}{R}(V_x - V_y - (l+w)\omega)$$

$$u_{fr} = \frac{1}{R}(V_x + V_y + (l+w)\omega)$$

$$u_{rr} = \frac{1}{R}(V_x + V_y - (l+w)\omega)$$

$$u_{rl} = \frac{1}{R}(V_x - V_y + (l+w)\omega)$$

NOTE: RPM are received in rad/min and position in ticks

$$u = \frac{RPM}{60 * T} = \frac{dp}{dt}\frac{2\pi}{N \cdot T}$$

Convert a point in the local reference frame to a point in the global reference frame (used in integration methods):

$$X_G = \cos(\theta)X_L - \sin(\theta)Y_L$$

$$Y_G = \sin(\theta)X_L + \cos(\theta)Y_L$$

Calibration

To estimate the CPR encoder I used the RPM velocities received via the sensor_msgs/JointState message.

$$N = \frac{dp}{dt}\frac{2\pi \cdot 60}{RPM}$$

To estimate the wheel radius, I inverted the Euler integration formula used to calculate the odometry in the main program, thus obtaining, from the pose received from the geometry_msgs/PoseStamped message, the linear and angular velocities.

$$R = \frac{4V_x \cdot 60 \cdot T}{RPM_{fl} + RPM_{fr} + RPM_{rr} + RPM_{rl}} = \frac{4V_y \cdot 60 \cdot T}{-RPM_{fl} + RPM_{fr} + RPM_{rr} - RPM_{rl}}$$

To estimate the wheel positions along the x- and y-axis, I used the previously estimated radius and velocities.

$$l + w = \frac{\frac{RPM_{fl} \cdot R}{60 \cdot T} - V_x + V_y}{-\omega} = \frac{\frac{RPM_{fr} \cdot R}{60 \cdot T} - V_x - V_y}{\omega} = \frac{\frac{RPM_{rr} \cdot R}{60 \cdot T} - V_x - V_y}{-\omega} = \frac{\frac{RPM_{Rl} \cdot R}{60 \cdot T} - V_x + V_y}{\omega}$$

I got l and w by making a proportion:

$$l = 0.5420054201 * (l + w)$$

$$w = 0.4579945799 * (l + w)$$

> V_x : linear velocity on the robot's x-axis
> V_y : linear velocity on the robot's y-axis
> ω : angular velocity of the robot
> u_{fl} : left front wheel speed
> u_{fr} : right front wheel speed
> u_{rr} : right rear wheel speed
> u_{rl} : left rear wheel speed
> dp : current position - previous position
> dt : current time - previous time
> RPM : revolutions per minute, in this project are rad/min
> X_G : coordinate x in the global reference frame (global because it defines the world in which the robot moves)
> Y_G : coordinate y in the global reference frame (global because it defines the world in which the robot moves)
> X_L : coordinate x in the local reference frame (the coordinate frame from the perspective of the robot)
> Y_L : coordinate y in the local reference frame (the coordinate frame from the perspective of the robot)

Robot parameters:

> R : wheel radius
> l : wheel position along x
> w : wheel position along y
> N : encoder CPR
> T : gear ratio