FIRST ROBOTICS PROJECT

ROBOTICS

POLITECNICO
MILANO 1863

THE ROBOT

SCOUT 2.0

AgileX Robotics

Unmanned ground vehicle (UGV)

Skid steering

4 Motors (1 for each wheel)

Links: Webpage, User Manual

THE ROBOT

Disclaimer: this video is only for demonstrational purposes. It does NOT show the actual path followed by the robot during our data acquisition.

THE PROJECT

Given:

- 4 motor speed (RPM)
- simple odometry provided by the manufacturer
- ground truth pose of the robot (acquired with OptiTrack)

THE PROJECT

Goals:

I. compute odometry using skid steering (approx) kinematics
   - using Euler and Runge-Kutta integration
   - ROS parameter specifies initial pose

II. use dynamic reconfigure to select between integration method

III. write 2 services to reset the odometry to (0,0) or to a pose(x,y,θ)

IV. publish a custom message with odometry value and type of integration

I. COMPUTE ODOMETRY

- Use message filters to synchronize motor speed messages
- Estimate linear and angular velocity of the robot from motors speed
  - You can use the manufacturer odometry to estimate the apparent baseline (required) and the gear ratio (optional)
    - ROS tools can help you here: rviz, rqt_plot, plotjuggler, …
  - Publish as geometry_msgs/TwistStamped
- Compute odometry with skid steering approx kinematics
  - Start with Euler, add Runge-Kutta later
  - A ROS parameter defines initial pose of the robot (x,y,θ)
  - Publish as nav_msgs/Odometry and TF
- Optional: use the ground truth pose to calibrate the apparent baseline for skid steering instead of the manufacturer odometry

II. INTEGRATION METHOD SELECTOR

- Use dynamic reconfigure to select the odometry integration method
- Use an enum with 2 values: Euler, Runge-Kutta

III. RESET SERVICE

- Define a service to reset odometry to (0,0)
- Define another service to reset odometry to any given pose (x,y,θ)

IV. CUSTOM MESSAGE

- Publish a custom message with prototype:

```
nav_msgs/Odometry odom
std_msgs/String method
```

- `method` can be either `"euler"` or `"rk"`

DATA

ROS bag file, with topics:

- motor speed for all 4 motors (f: front, r: rear; r: right, l: left)
  - /motor_speed_fr – robotics_hw1/MotorSpeed
  - /motor_speed_fl – robotics_hw1/MotorSpeed
  - /motor_speed_rr – robotics_hw1/MotorSpeed
  - /motor_speed_rl – robotics_hw1/MotorSpeed

Custom message, given

- simple odometry provided by the manufacturer
  - /scout_odom – nav_msgs/Odometry
- Ground truth pose of the robot (acquired with OptiTrack)
  - /gt_pose – geometry_msgs/PoseStamped

DATA

Additional information:

- Wheel radius: 0.1575 m
- Real baseline: 0.583 m (This is the actual distance between a pair of left and right wheels. The skid steering *apparent baseline* will be larger than this.)
- Apparent baseline: you will have to calibrate it ☺
- Gear ratio: ~~as in the robot manual~~ to be calibrated (optional)
  - Between 1:35 and 1:40

DATA

Files provided:

- 3 ROS bag with data:
  - Use bag1 as main data source
  - Use bag2 and bag3 to double-check
- Package `robotics_hw1` with:
  - message definition for msg `MotorSpeed`

IMPORTANT NOTICE

The ground truth pose is measured with an Optitrack system, which is based on cameras. For this reason, this information might sporadically not be available due to occlusion. This should not affect your project, just be aware of it.

To complete this project, you can use any number of nodes.