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 odometry provided by the manufacturer to estimate it
    - 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:

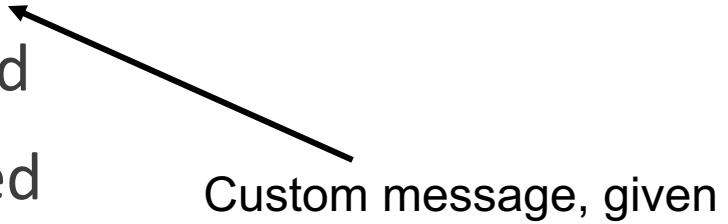
```
nav_msgs/Odometry odom
std_msgs/String method
```

- `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
- 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 ☺

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.

GENERAL REQUIREMENTS

The project must be written in C/C++

- (No Python unless previously discussed)

You should provide us with **1 single launch file** that starts **everything** needed except for the bag (i.e. all needed nodes, parameters, etc. must be setup in there).

DO NOT use absolute path

- (We should be able to run your code on our machines)

TIPS FOR DEBUG

You can use rviz to visualize given odometry, ground truth pose and TF.

You can use it also to visualize your computed odometry and TF.

To do so, you might have to define a static transform between the frame of your odometry ("odom") and the frame of the ground truth pose ("world").

Remember also to set the odometry initial pose correctly.

Tip: the robot always keeps still for a few seconds at the beginning of every bag.

FILES

Everything is in our

shared folder

(link on the course website)

Deadlines and requested files

IMPORTANT: PLEASE READ CAREFULLY

- Create a .tar.gz archive containing:
  - a text file with instructions to test your project (details in next slide)
  - the source code for every package you created (entire package folders, including their `CMakeLists`, `package.xml`, etc.)
    - Include all the files you think are important, as we should be able to properly recreate your workflow with what you send us
    - Please DO NOT send us your entire catkin environment (with `build` and `devel` folders); we only need the packages you created inside of your `src` folder
- Name the archive with your Student ID ("Person Code")
- Send this via e-mail to **all of us**: Paolo Cudrano, Simone Mentasti and Matteo Matteucci, email subject must be “FIRST ROBOTICS PROJECT 2021”

Deadlines and requested files

The file with instructions must be a .txt file and **must** contain (at least):

- Student ID ("Person Code"), name and surname of all team members
- small description of the files inside the archive
- name and meaning of the ROS parameters
- structure of the TF tree
- structure of the custom message
- description of how to start/use the nodes
- info you think are important/interesting

Deadlines and requested files

Deadline: 16 May 2021 (at 23:59:59 Italian time)

Max 3 student for team (unless previously discussed)

Questions:

- write to me → Paolo Cudrano (paolo.cudrano@polimi.it)
  - do not write only to Prof. Matteucci
- ask on Teams (MATTEUCCI MATTEO 089013-ROBOTICS (745305))