Robotics project Part 1

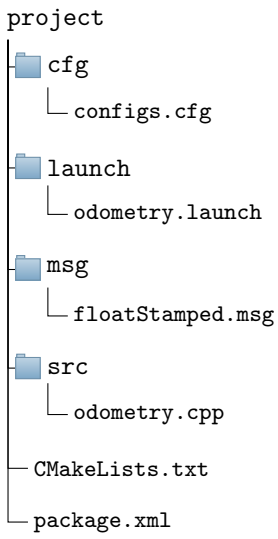
Luca Massini [844049]
Roland Reylander [868917]

May 29th 2019

1 Files inside the archive

The project folder contains the following files:

- `configs.cfg` used for dynamically set:
  - Differential Drive Kinematics or Ackerman model
  - initial x and y coordinates
- `odometry.launch` to launch the main node
- `floatStamped.msg` used for retrieve data from the bag
- `odometry.cpp` where the code of the main node is written
- `CMakeLists.txt` and `package.xml` to specify what will be used.



2 Description of how to start/use the nodes

Fist place the folder inside your catkin workspace and run in the terminal:

```
$ catkin_make
```

There are two ways to start the node:
using rosrun

```
$ roscore
$ rosrun project odometry
```

using roslaunch

```
$ roslaunch project odometry.launch
```

After this, it is ready to read the bag file running:

```
$ rosbag play bag_name.bag
```

2.1 Read the published odometry

The Node works as subscriber of the bag as well as publisher of the odometry. To view the published content run:

```
$ rostopic echo /odom
```

Whereas to see the published tf run:

```
$ rostopic echo tf
```

Both the tf and the odometry topic make use of the `geometry_msgs`, so the pose has been published using `x` and `y` coordinates, whereas for the rotation the quaternion system has been used.

2.2 Dynamically configuration

To dynamically reconfigure the parameteres it is possible to use the following commands:

```
$ rosrun dynamic_reconfigure dynparam set /odometry x VALUE
```

to set the `x` coordinate to a specified VALUE

```
$ rosrun dynamic_reconfigure dynparam set /odometry y VALUE
```

to set the `y` coordinate to a specified VALUE

```
$ rosrun dynamic_reconfigure dynparam set /odometry diff_acker CONDITION
```

set the `CONDITION` to `true` for Differential Drive Kinematics or `false` for Ackerman model

3 How it works

The initial idea was to read directly from the bag file but later we decided to subscribe to the bag's topic and retrieve the data like this. So the node works as subscriber of the bag's topic. The next step was to read the data, at the beginning we tried to read the `speed_R`, `speed_L` and `steer` topics, but we had problems with synchronizing the three topics, so the alternative was to subscribe to the `speedR_stamped`, `speedL_stamped` and `steer_stmped` topics. For doing this we needed a custom message that included the timestamp in the Header and a `float` variable.
After this we wrote funcions to calculate the fomulae and to publish from the same node, so every time the bag published the speed of the wheels and the steer the node could publish the pose from the same node.
Finally we added funcions to dynamically reconfigure the parameters.

4 Tests

To check if the odometry was correctly calculated, we made use of `rviz`. We could observe that in both Differential Drive and Ackerman model the car was moving in circles.
Using the play and stop mode of `rosbag play` the pose is not precise. This is due to the time that is not taken from the bags timestamp, but the node uses the ROS function `ros::Time::now()` instead. So when the playing bag is temporary stopped, the delta of the time increases and the formula does not work anymore.
Other tools we used where `rqt_graph` and `rqt_plot` to check the topics and the pose of the car. Moreover we used also the `rqt_reconfigure` for gui based reconfiguration of the parameters. Lastly we used the `ROS_INFO` function that allowed us debug the formulae and see in real time if every part of the code was working correctly.