POLITECNICO DI MILANO

ROBOTICS 2020-2021
prof. Matteo Matteucci

DELIVERABLE

Homework 1

TITLE

First robotics project

SCENARIO

SCOUT 2.0 AgileX Robotics

GROUP

Chicago

Team Members

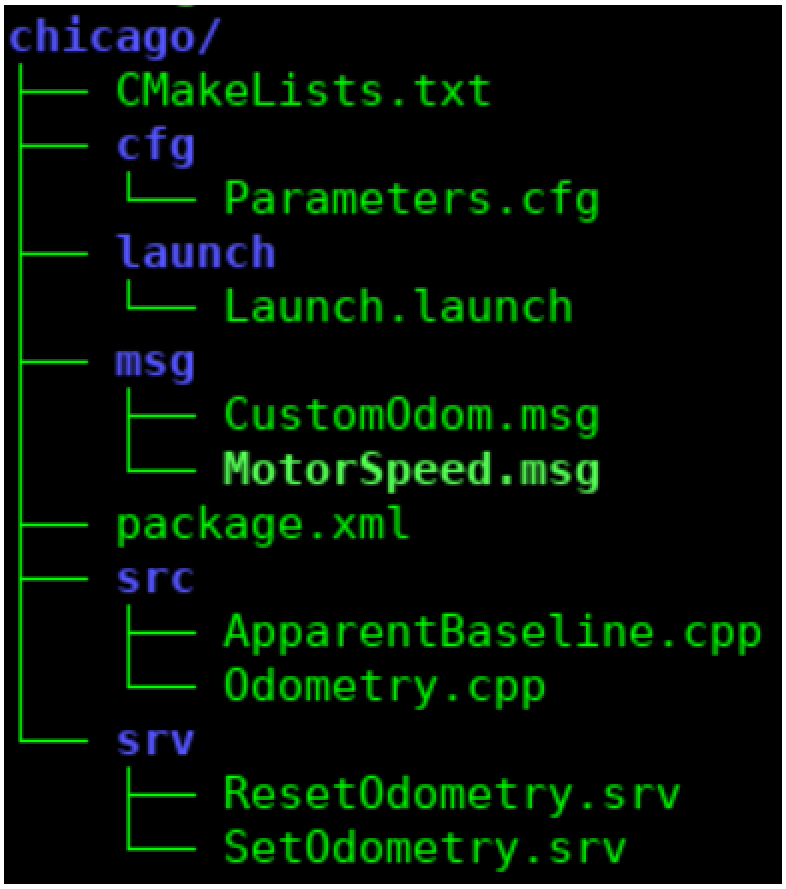
ID	Surname	Name
10529039	Cappelletti	Andrea
10568052	Castelli	Francesco
10674448	Foini	Davide

INDEX

1. FILES INSIDE OUR PROJECT

In this section, we present all the files inside our project. To be more clear, we report the tree structure of our project through the command line

```
:~$ tree
```



```
chicago/
├── CMakeLists.txt
├── cfg
│   └── Parameters.cfg
├── launch
│   └── Launch.launch
├── msg
│   ├── CustomOdom.msg
│   └── MotorSpeed.msg
├── package.xml
├── src
│   ├── ApparentBaseline.cpp
│   └── Odometry.cpp
└── srv
    ├── ResetOdometry.srv
    └── SetOdometry.srv
```

Our project directory is called “chicago”.
Inside the project directory we divided the subdirectories into cfg (config), launch (the launch configuration for the project), msg (the messages directory), src (the source code in C++ of the project) and srv for the services.

Inside the project directory chicago there are also the two files needed for the build of the entire project: package.xml and CMakeLists.txt.

We used a coherent naming convention for the entire project, more in detail:

chicago/cfg/

Parameters.cfg contains dynamic reconfig params

chicago/launch/

Launch.launch contains the directive to start the project

chicago/msg/

MotorSpeed.msg contains the defined message
CustomOdom.msg contains the output odometry

chicago/src/

ApparentBaseline.cpp the node to compute the apparent baseline

Odometry.cpp the node to compute the odometry

chicago/srv/

SetOdometry.srv service to set the odometry
ResetOdometry.srv service to reset the odometry

2. ROS Parameters

In this section, we present the parameters we defined in our project.

Static

Starting coordinates

- gear_rateo, double, [0.027] (calibrated)
- wheel_radius, double, [0.1575]
- apparent_baseline, double, [1.0625] (calibrated)

We have computed the gear rateo and the apparent baseline taking into account some consideration that we report in the sections below (additional info).

The wheel radius is already defined from the project specification and reported in the project structure.

Starting coordinates

- x, integer, [0]
- y, integer, [0]
- th, integer, [0]

Starting integration method (Euler)

- method, integer, [0]

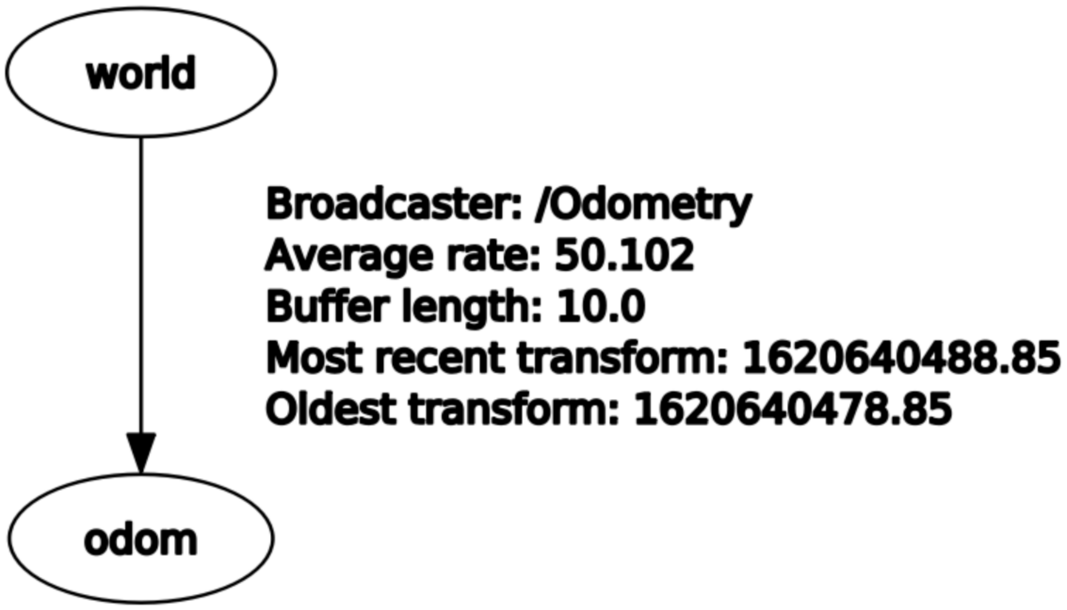
The integration method is defined as an enum with two values, respectively "euler" and "rk".
"euler" has index 0, meanwhile "rk" has index 1.

Dynamic

- method_enum, enum, [euler, rk]

3. TF TREE structure

Below you can find the TF TREE structure of our project: it is very simple because we chose to use just one frame “odom” that represents the scout transformed to the “world” frame.



4. START NODES

In order to launch the nodes, run this command from terminal

```
:~$ roslaunch chicago Launch.launch
```

We organized our work in just two nodes: one for estimating the apparent baseline of our robot, and the second to compute all the required tasks.

Services

As presented above, in our project we have two services: one to set the odometry and one to reset the odometry.

To launch the services run the following commands

SetOdometry.srv

```
rosservice call /SetOdometry [x] [y] [th]
```

Substituting [x], [y] and [th] with the values.

ResetOdometry.srv

```
rosservice call /ResetOdometry
```

Dynamic Reconfiguration

In order to perform the dynamic reconfiguration, run the following command

```
rosrun dynamic_reconfigure dynparam set
Odometry method [index]
```

Where [index] is the method chosen

0: Euler
1: RK

5. ADDITIONAL INFO

Apparent Baseline

To estimate the apparent baseline, we imported the information about the motor speeds from the bags, and we converted them into linear velocities for the (approximated) two wheels of the robot.

Then by just applying the Skid Steering (approximate) Kinematics we computed the apparent baseline.
The main problem was that we had a lot of values for it, and lots of them were not physically possible (i.e zero or negative values or infinite values).

So what we decided to do was to put some boundaries on the acceptable apparent baseline values and then compute an average of them.

This led us to an acceptable value for the apparent baseline that we then set as a parameter.

Calibration

After we completed the odometry computation we decided to try to calibrate the apparent baseline and the gear rateo to obtain a better result.
Helping us with visualizing the TF on rviz after some tries we obtained that the values **1.0625** for the apparent baseline and **0.027** for the gear rateo that combined can get closer to the odometry given by the manufacturer.

Default parameters

The default parameters x, y, theta can be changed in the launch file Launch.launch or via command line with the command

```
rosparam set [param] [value]
```

param: x/y/th
value: any integer value

Regarding the default integration method, it can be changed in the Parameters.cfg file modifying the default value of the enum parameter “method”:

- 0 for Euler;
- 1 for Runge-Kutta.

Publishers

We chose to use three publishers:

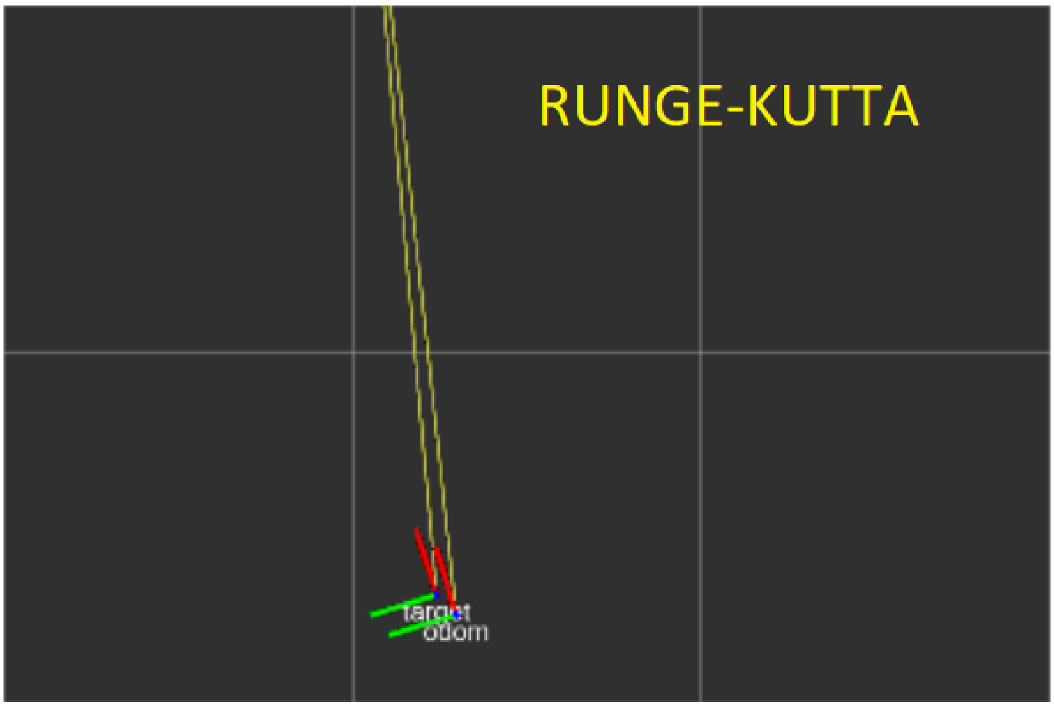
- **twist_pub** that advertises to the topic /twist the TwistStamped messages;
- **custom_pub** that advertises to the topic /custom_odom the CustomOdom messages;
- **odom_pub** that advertises to the topic /odometry the Odometry messages;

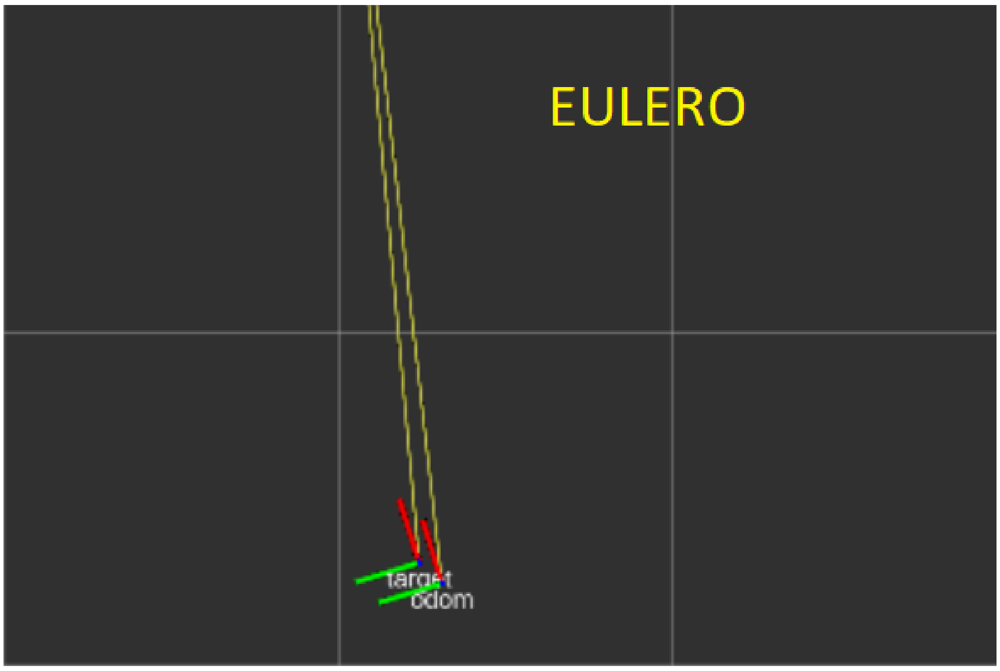
6. FINAL CONCLUSION

Once started the node, it can be seen thanks to rviz that the trajectory of our robot follows faithfully the given path, even though the two trajectories are slightly different.

In addition, changing the integration method from Euler to Runge-Kutta slightly increases the precision of the trajectory.

This frames are taken 60 seconds into bag1:




EULERO
target
odom