DWA vs custom trajectory tracking controller

A comparison in ROS

G. Chiari L. Gargani S. Salvi

Politecnico di Milano

Academic Year 2022/2023

Table of Contents

Table of Contents

Frame title

In this slide, some important text will be highlighted because it's important. Please, don't abuse it.

Remark

Sample text

Important theorem

Sample text in red box

Examples

Sample text in green box. The title of the block is "Examples".

Frame title

This is a text in first column.

$$E = mc^2$$

- First item
- Second item

This text will be in the second column and on a second thoughts, this is a nice looking layout in some cases.

Project overview

Test...

Differential drive

Test...

Table of Contents

Paper formulation

Test...

ROS implementation

Test...

Table of Contents

Unicycle model control

Test...

Controller's architecture

Test...

Table of Contents

Experiment setup - the robot

The robot:

- differential drive
- $\mathtt{d} = 15$ cm (wheels distance)
- $\mathtt{r} = 3$ cm (wheels radius)
- pentagonal footprint

[IMAGE FROM RVIZ]

Experiment setup - the map

The map:

- empty, no obstacles around
- global, no need for a local one

[IMAGE FROM RVIZ]

Experiment setup - the trajectory

The trajectory:

- eight-shaped
- 2 x 1 meters
- discretized into multiple goals

[IMAGE FROM RVIZ]

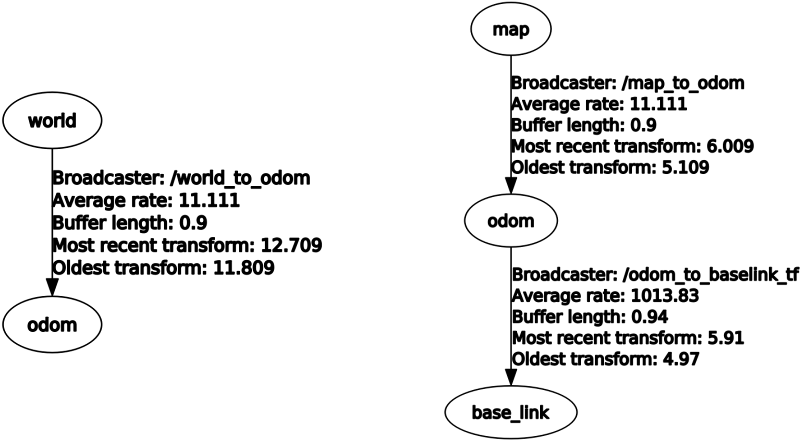
ROS architecture - overview (1)

Three packages:

- `diffdrive_kin_sim` → simulator
- `diffdrive_kin_ctrl` → custom controller
- `diffdrive_dwa_ctrl` → DWA controller

The two controllers are interchangeable and are meant to always be used together with the simulator, one at a time.

ROS architecture - overview (2)



ROS architecture - service & frames (1)

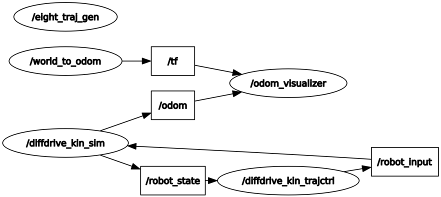
One service:

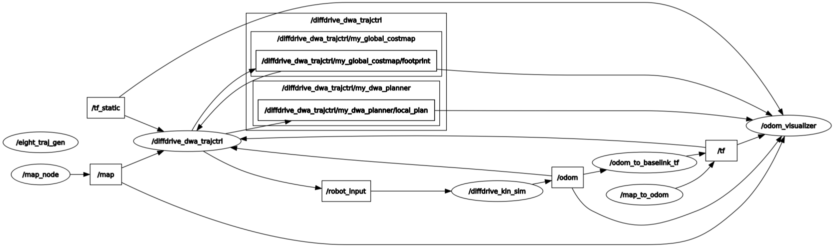
- `generate_desired_path_service` → used to generate the eight-shaped trajectory as soon as it is invoked by one of the two controllers

Three frames:

- `map` → for the empty map provided by the map server
- `odom` → representing the global reference system
- `base_link` → representing the moving reference system

ROS architecture - service & frames (2)





ROS architecture - nodes

Four nodes:

- `diffdrive_kin_sim_node`, $\rightarrow$ integration logic to update the robot pose given (ω_r,ω_l)
- `diffdrive_kin_trajctrl_node`, $\rightarrow$ computation of (ω_r,ω_l) to reach the next point in the trajectory, using the custom controller
- `diffdrive_dwa_trajctrl_node`, $\rightarrow$ interface between DWA library and simulator to compute (ω_r,ω_l)
- `odom_to_baselink_tf_node`, link between the `odom` and the `base_link` coordinate frames through a dynamic tf

ROS architecture - topics

Four nodes:

- `/clock`, $\rightarrow$ synchronization of all the nodes in the simulation
- `/odom`, $\rightarrow$ communication of the odometry information of the robot to DWA
- `/robot_state`, $\rightarrow$ communication of the odometry information of the robot to the custom controller
- `/controller_state`, $\rightarrow$ for visualization purposes
- `/robot_input`, $\rightarrow$ communication of (ω_r, ω_l) computed by the controllers

ROS architecture - launch files

Two launch files:

- `diffdrive_kin_trajctrl.launch` $\rightarrow$ start the simulation of the robot's behavior with the custom controller
- `diffdrive_dwa_trajctrl.launch` $\rightarrow$ start the simulation of the robot's behavior with DWA

Parameters tuning - trajectory

$$\begin{cases} x = \mathrm{a} \cdot \sin(\mathtt{w} \cdot t) \\ y = \mathrm{a} \cdot \sin(\mathtt{w} \cdot t) \cdot \cos(\mathtt{w} \cdot t) \end{cases}$$

Two main parameters:

- a $\rightarrow$ amplitude of the eight-shaped trajectory
- $\mathtt{w}$ $\rightarrow$ ratio $\frac{2 \cdot \pi}{T}$ where T is the time duration of each lap

Assigned values

$$\mathrm{a} = 1 \qquad \mathtt{w} = 1$$

Parameters tuning - custom controller

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\,\mathrm{d}\tau + K_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

Three main parameters:

- $K_p \rightarrow$ proportional gain of the PID controller
- $K_i \rightarrow$ integral gain of the PID controller
- $K_d \rightarrow$ derivative gain of the PID controller

Assigned values

$K_p = 0.8$ $\quad K_i = 0.8$ $\quad K_d = 0.0$

Parameters tuning - DWA

One main parameter:

- skipped_goals → number of points to skip when feeding the trajectory to DWA

Assigned values

$$\texttt{skipped\_goals} = 15$$

Table of Contents

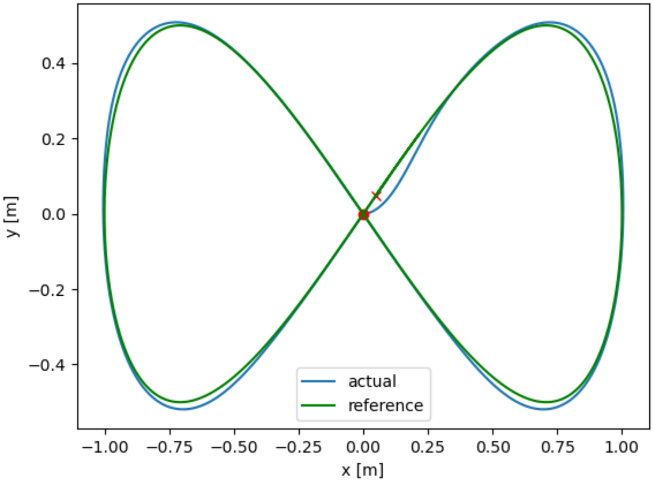
Python scripts

To visualize the experimental results of the experiment, two Python scripts can be used to analyze the recorded *bag* files:

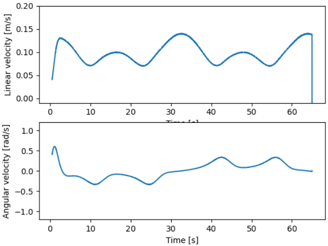
- `plot_result.py` → plot the result of a single *bag* file
- `plot_comparison.py` → compare the results of two *bag* files

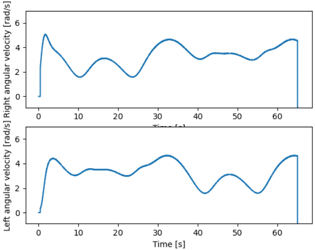
The following plots are obtained with an optimal parameters choice.

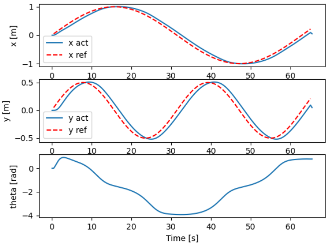
Trajectory tracking controller (1)

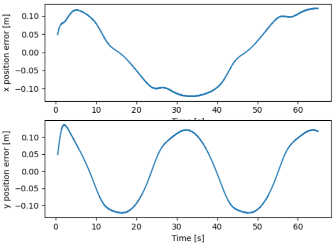


Trajectory tracking controller (2)

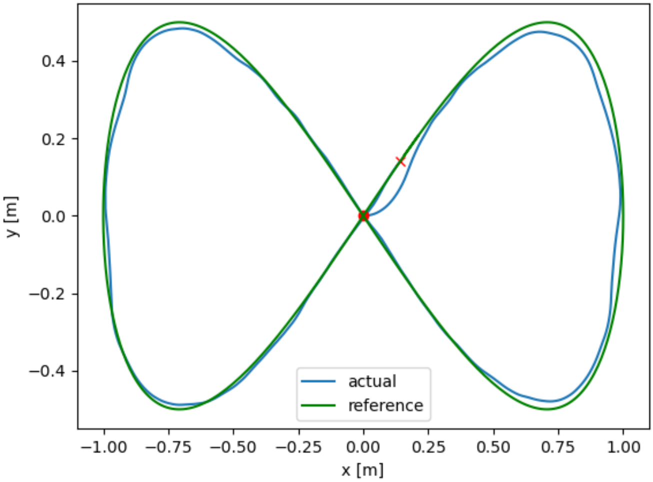




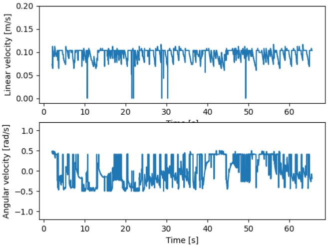


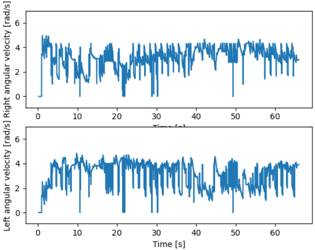


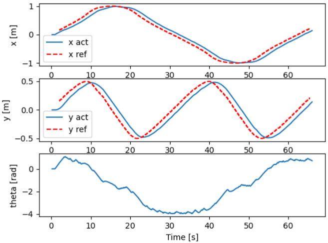
DWA (1)

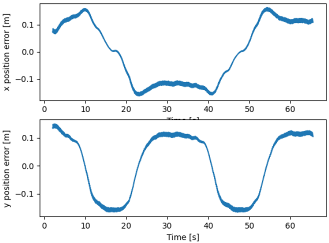

0.4
0.2
0.0
−0.2
−0.4
y [m]
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x [m]
actual
reference

DWA (2)

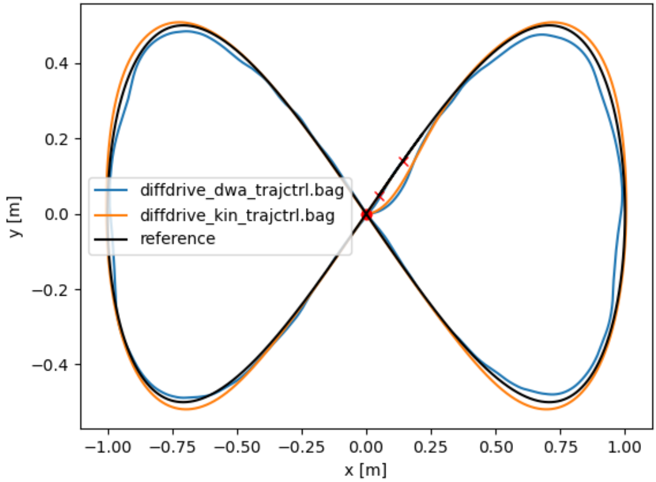

Linear velocity [m/s]
Angular velocity [rad/s]
Time [s]


Right angular velocity [rad/s]
Left angular velocity [rad/s]
Time [s]

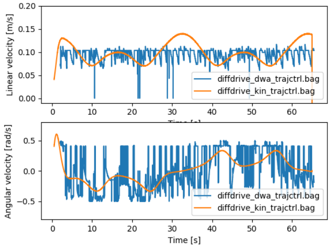

x [m]
x act
x ref
y [m]
y act
y ref
theta [rad]
Time [s]

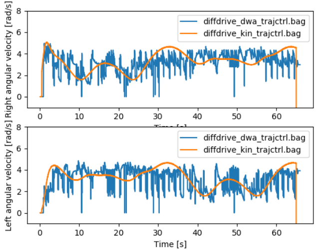

x position error [m]
y position error [m]
Time [s]

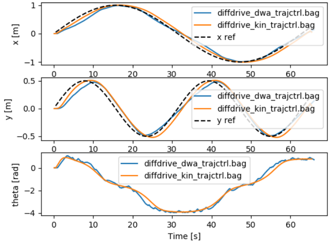
Comparison (1)



Comparison (2)







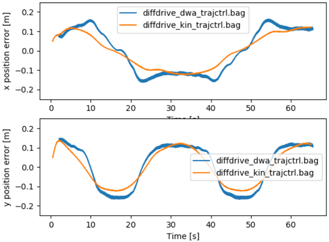


Table of Contents

Deprecated parameters

There are some DWA parameters that are named differently between the ROS Wiki and the library source code.

[ROS library source code warning]

Unadvertised warning!

Without using `nav_core`, the warnings are not raised.

DWA used standalone

DWA expects information on certain topics (`/odom`, `/goal`, `/map`). When used standalone, we must provide such information manually. Moreover, the few lines of code in the ROS Wiki explaining how to setup DWA standalone are no longer working:
[screen of the two snippets]

Multiple goals

Generally, DWA takes only a single final goal and computes the full trajectory on its own.

In this project the trajectory is predefined and must be explicitly forced. This is done by continuously changing the goal.

We must find a suitable 'density' for the goals vector:

- too many goals $\Rightarrow$ unsteady profile in the velocities
- too few goals $\Rightarrow$ large deviation from the reference trajectory

Table of Contents

Conclusion

Regarding the trajectory tracking aspect only, the two control methods behave in a similar way even though the custom controller performs moderately better than DWA.

However, the most noticeable difference is in the smoothness of the plots of linear and angular velocities: the custom controller produces velocities that are much smoother and realistic than those produced by DWA.