PUTTING MOTION CONTROLLERS TO THE TEST

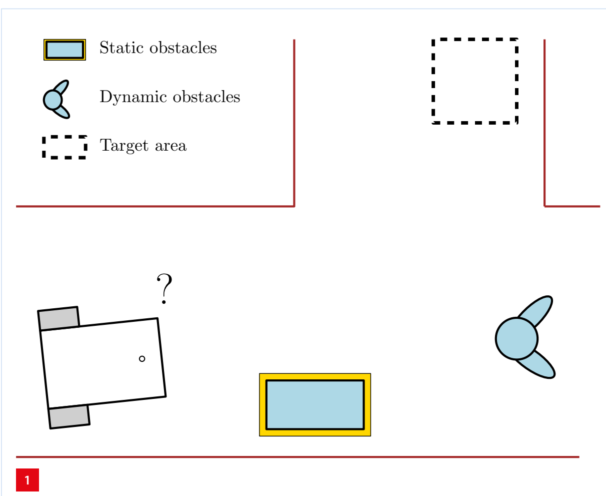
There is an increasing need for deploying Autonomous Mobile Robots (AMRs) in the care sector, the public domain (hospitality and surveillance) and the industry (inspection, maintenance, logistics and agriculture). One of the challenges when developing mobile robot navigation technology is selecting the right, application-specific motion controller. The different motion controllers available in the ROS2 robot operating system framework have been tested on two robots with a different footprint performing various tasks. The results have been translated into a concise selection guideline.

CÉSAR LÓPEZ, MUKUNDA BHARATHEESHA, BRAM ODROSSLIJ AND FRANK SPERLING

Introduction

AMR technology evolved to solve issues associated with AGVs (Autonomous Guided Vehicles), such as docking accuracy, localisation, navigation in unknown environments, and obstacle avoidance. One of the main enablers that has accelerated the growth of AMR technology is the open-source Robot Operating System, i.e. the ROS (latest version ROS2) middleware framework.

In this article, the focus is on mobile robot navigation technology (see Figure 1), which must ensure a safe, reliable and efficient operation in an environment with fixed structures (walls, doors, etc.), static yet movable obstacles and dynamic obstacles such as people. Motion planning algorithms deal with the problem of finding a sequence of velocity and steering commands that will result in the mobile robot successfully reaching the desired target, subject to certain constraints and performance criteria, such as maximum time, velocity and accuracy.



1

The mobile robot navigation problem: how to generate a set of velocity and steering commands to let the robot reach its target area?

AUTHORS' NOTE

César López (robotics designer), Mukunda Bharatheesha (robotics engineer) and Frank Sperling (director Technology) all work at Nobleo Technology in Eindhoven (NL), an engineering firm specialised in autonomous intelligent systems. Bram Odrosslij performed his M.Sc. graduation project as a student of Mechanical Engineering (Eindhoven University of Technology) at Nobleo Technology. This article presents a summary of their whitepaper "Mobile Robot Navigation in ROS2 – Motion Controllers Comparison", which can be downloaded at www.nobleo-technology.nl/news/whitepaper-mobile-robot-navigation-ros2/. References can be found in the whitepaper.

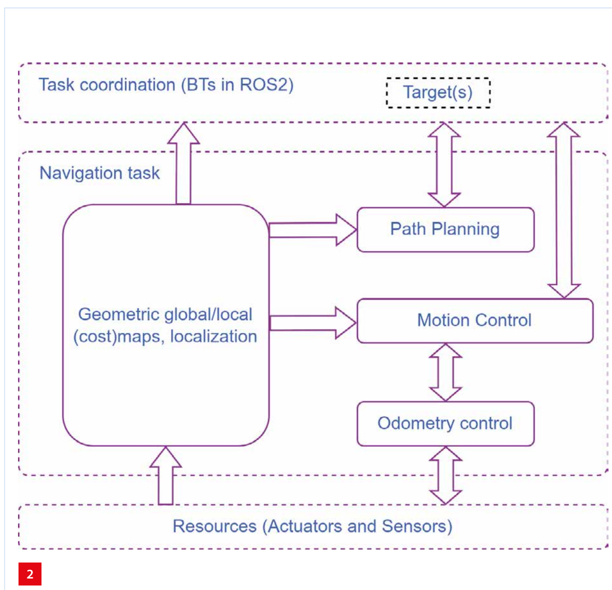
frank.sperling@nobleo.nl
www.nobleo-technology.nl

ROS2 navigation architecture

ROS2 navigation uses a modular, configurable architecture that is gaining momentum as a standard for mobile robot navigation, comprising three main blocks: environment representation, global path planning, and local motion control; see Figure 2.

Environment representation

Global and local cost maps are used to represent the fixed environment, as well as the static and dynamic obstacles. Exact collisions cannot be detected at this stage since they depend on the robot's orientation. The fixed infrastructure is typically represented in the global cost map, which uses the map of the environment as input. The static and dynamic obstacles are represented in the local cost map, which uses inputs from sensors such as lidar, sonar and 3D cameras.



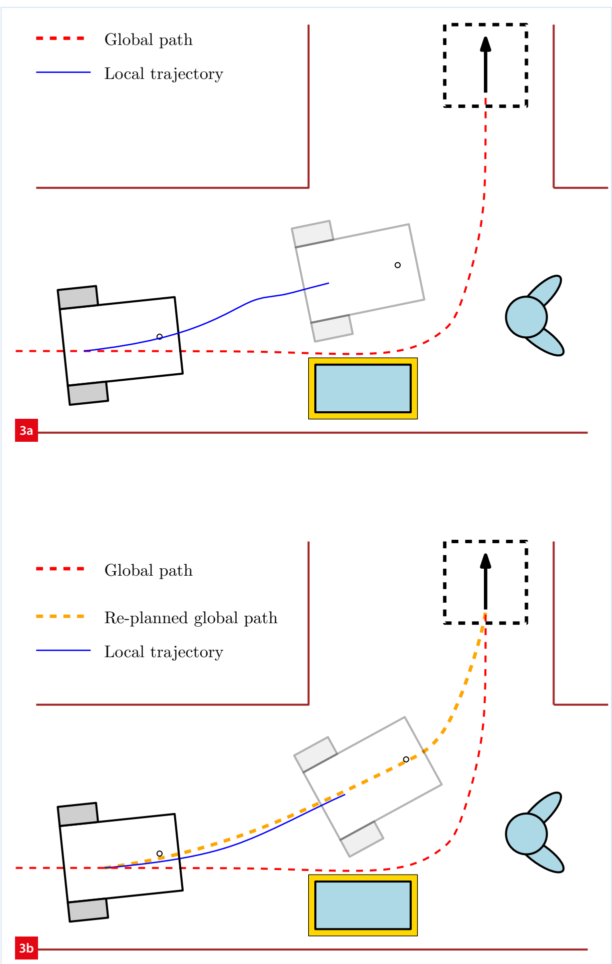
2

ROS2 robot navigation architecture.

Global path planning
Path planning is in charge of finding a collision-free path, i.e. a sequence of poses (positions and orientations) that connect a starting pose to a target pose. The path planner uses the global cost map together with the geometric footprint of the robot to assess whether it will be in collision with the fixed environment. In addition, path planners take into account the robot's kinematic constraints; for example, the most common driving mechanism in industrial robots is the differential drive, which cannot drive the robot sideways.

For path planning there exist multiple algorithms, from classical A* and Rapidly-exploring Random Trees to machine-learning-based path planning.

Local motion control
Motion control is in charge of generating the actual velocity commands, using the generated global path to guide the robot towards the target pose along a sequence of poses known as the local trajectory. Many motion controllers deal internally with unknown obstacles and generate velocities that drive the robot around them, resulting in local trajectories that can largely deviate from the original global path (Figure 3). An alternative is to request global path re-planning.



For previously unknown obstacles, two approaches can be followed.
(a) The motion controller planning a new trajectory.
(b) The path planner generating a new global path.

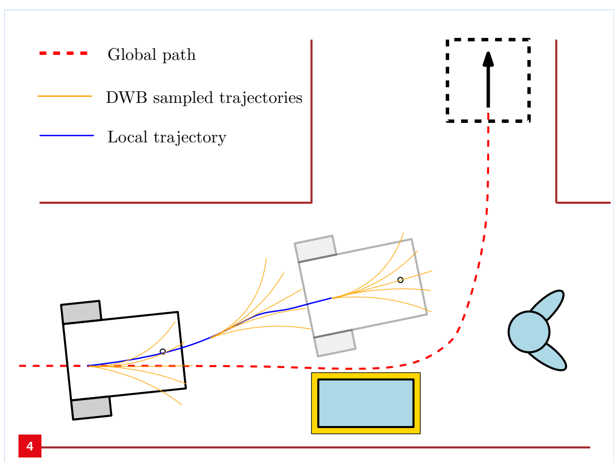
Execution
The high-level component task coordination is thus in charge of harmonising path planning and motion control to complete the navigation task. Finally, odometry control makes sure that the velocity and steering commands are properly executed in the robot.

ROS2 motion controllers
In ROS2 a number of motion controllers are available.

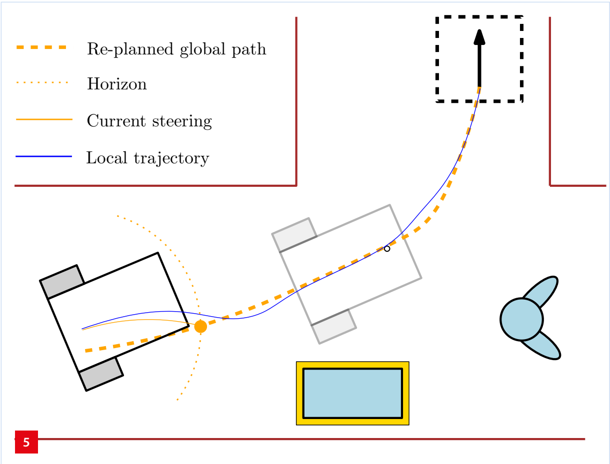
DWB: enhanced Dynamic Window Approach
Fundamentally, DWB is a modularised and enhanced version of the Dynamic Window Approach (DWA), featuring a configurable selection of (customisable) scoring functions, which can increase the efficiency of the controller and help to prevent navigation failures.

In its basic form, DWA/DWB (Figure 4) uses a trajectory generation and selection approach in an iterative process comprising four steps:
1. Discretely sampling the robot's control space.
2. Performing a forward simulation of each sampled control to predict its effect.
3. Scoring each resulting trajectory, using a metric that incorporates characteristics such as proximity to obstacles, proximity to the goal, proximity to the global path, and velocity.
4. Picking the highest-scoring trajectory and using the associated controls.

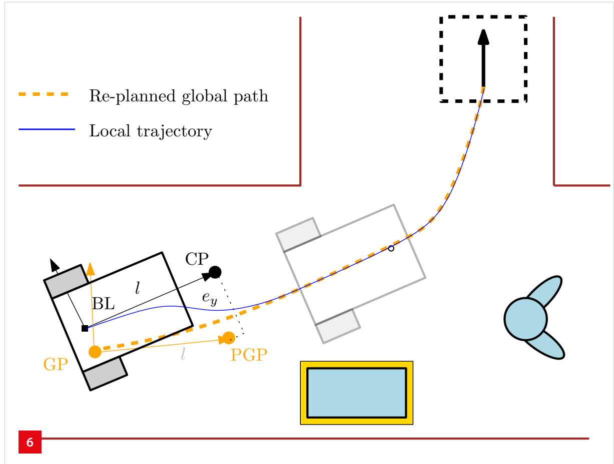
DWA has an inherent ability to deal with dynamic obstacles.



DWB strategy: discrete sampling of the robot control space followed by a selection based on several criteria.



RPP strategy: looking ahead on the path to compute the carrot target and the feedforward steering to reach it.



PTPID strategy: computing the carrot target by projecting the closest pose on the path (PGP); velocities are computed using a PID controller to minimise the error e_y between the control point (CP) and the PGP.

RPP: Regulated Pure Pursuit

RPP enables a mobile robot to track a global path by continuously moving towards a target point called look-ahead point (or carrot target) on incremental segments of a global path. For every lookahead point, control commands are generated so that the robot starts moving towards it. To determine the most relevant lookahead point (indicated by the yellow dot in Figure 5), typically lying on a circular horizon around the robot's control point, the global path that the robot has to follow is used as a cue.

RPP extends the basic Pure Pursuit algorithm to enhance its applicability to a wider scope of practical problems. This is achieved via the introduction of active collision detection, velocity-scaled lookahead points (enabling a larger lookahead horizon at higher speeds), and velocity modulation (slowing down) when the robot approaches the desired goal.

PTPID: Path Tracking PID

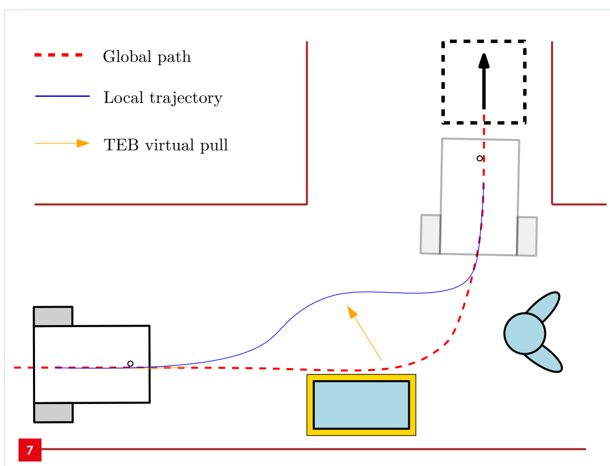
The PTPID motion controller addresses the challenge of accurately following a predetermined path. Nobleo has developed the first open-source high-performance path-tracking algorithm for mobile robots. The PTPID concept relies on two aspects (Figure 6): projection of the global path, and accurate control of a point ahead of the robot, known as a carrot point.

Global path projection involves computing a path that if followed accurately with a carrot point (or control point, CP), will result in the base link (BL) of the robot, defined at its rotation point, following the original global path. At each update step, the closest point from the base link to the path (the global pose, GP) is found and a projected global pose (PGP) is computed. The carrot point CP is simply calculated by projecting the base link in the direction of the robot's orientation. The objective is to get the CP close to the PGP, their locations staying together ideally. To this end, PTPID uses concepts applied in high-precision control, mainly the combination of feedback and feedforward.

TEB: Timed Elastic Bands

TEB formulates its task as a nonlinear (least-squares) optimisation problem. A key difference of TEB with respect to other motion controller methods is that it can efficiently solve the relevant equations, making it usable for mobile robots without the need for powerful computers.

TEB relies on solving an optimisation problem with constraints; see Figure 7. Virtual springs are added at certain locations along the path and time stamps are added to them. Far from obstacles, the virtual springs are not active, and the resolved trajectory would be very close to the original path. When the robot approaches an obstacle, the virtual springs are activated, and the trajectory is 'pulled' away from the original path.



TEB strategy: formulating an optimisation problem such that virtual springs with time stamps are added along the global path; the springs allow the path to be elastic and malleable in order to avoid obstacles.

The two robots used for the comparison of motion controllers.
(a) SARA.
(b) IDA.

Motion controllers benchmark

To make a valid and realistic comparison of the different ROS2 motion controllers, two robots (Figure 8) used in real-life applications were selected: a care robot and a logistics robot, with different sizes and placements of the rotation point.

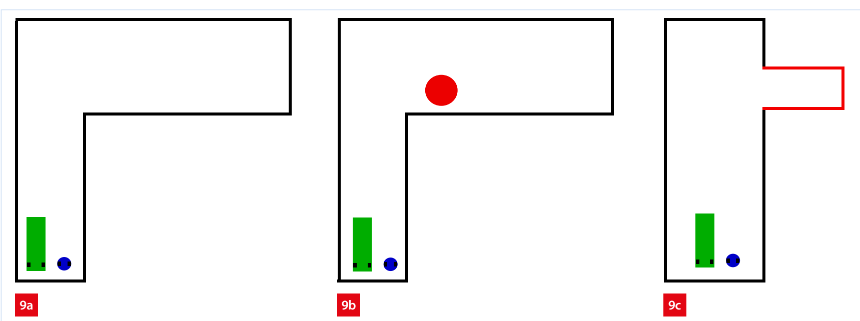
Robots

SARA (Figure 8a) is used to support tasks in healthcare; Nobleo had collaborated with SARA Robotics to bring new functionalities to their product. Its footprint is relatively small with respect to its environment. Sara can move sideways, but is treated here as a differential-drive robot, because differential-drive design is commonplace in industrial AMRs.

IDA (Figure 8b) is a manually driven pallet truck that has been made fully autonomous by Nobleo. Its rotation point is located close to the back of the robot, while in the front it has a steering wheel. In this comparison, we assumed the steering wheel can rotate very fast to make a fair comparison with a differential drive.

Test scenarios

Three test scenarios (Figure 9) were 'played' in the benchmark.



Test scenarios; for size comparison, the footprints of SARA and IDA are shown as a blue circle and a green polygon, respectively.
(a) Turning from hallway to hallway.
(b) Turning from hallway to hallway with an unknown obstacle (red circle) present.
(c) Parking in a tight space (in red).

The first test (turning without obstacle, Figure 9a) was performed to check the basic ability of different controllers to drive safely. The second test (turning with obstacle, Figure 9b) involved dealing with previously unknown obstacles. Finally, the parking test (Figure 9c) dealt with accurately following the path and producing accurate movements in a tight environment.
For the tests, the Webots simulator was used, since it allows for easy (re)spawning of maps, robots and obstacles.
The tests had a maximum execution time, and when this was exceeded, the test was deemed unsuccessful.

Test results

The test results are shown in Figure 10.

- Turning without obstacle:
  For SARA as well as IDA, all tests succeeded, without major differences in execution time.
- Turning with obstacle:
  As expected, PTPID and RPP failed, since both do not circumvent unforeseen obstacles.
  For SARA, the TEB local planner succeeded since it is quite flexible and handles a circular footprint easily. DWB was also able to drive the robot safely around the obstacle, although it required more execution time.
  For IDA, TEB and DWB succeeded, showing that DWB can also be used with large footprints, given there is enough space for manoeuvring, although especially DWB required additional tuning effort.
- Parking:
  For SARA, all controllers succeeded – unsurprisingly, given its relatively small, circular footprint.
  For IDA, only TEB and PTPID succeeded, due to their flexibility and accuracy, respectively. DWB drove both the robots close to the path but not very accurately, hitting a corner. RPP got stuck because by design it follows a point ahead in the path, making the robot 'cut corners'.

Discussion and conclusion

The results show that TEB, unlike the other planners, has great flexibility to operate in different scenarios, and for different footprints, although it faced numerical issues

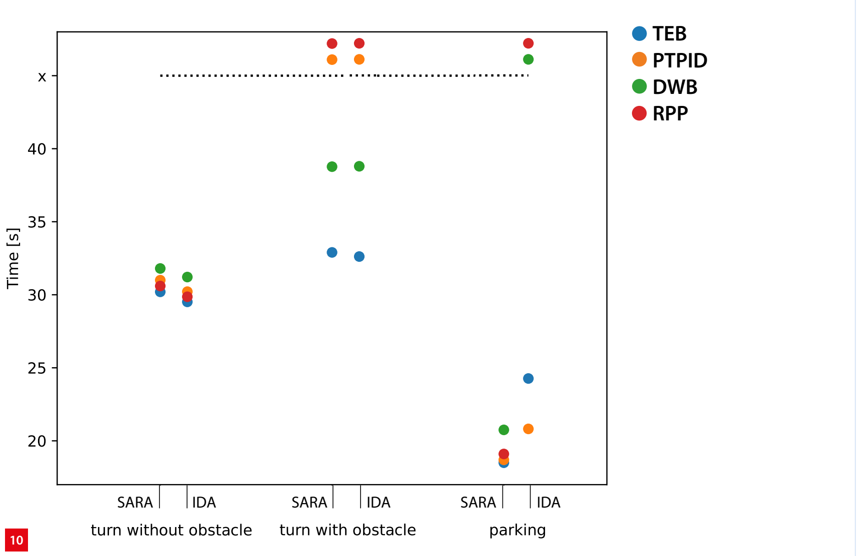
Table 1
ROS2 motion controller selection guideline.

Criterium	DWB	RPP	PTPID	TEB
Static known environments	+	+	+	+
Unknown obstacles	++	–	–	++
Small/symmetric footprint	+	+	+	+
Large/asymmetric footprint	+	–	++	+
Accuracy	+	–	++	+
Robustness/reproducibility	+	+	++	–
Computational simplicity	+	++	++	– –

with IDA, due to the nonlinearities induced by its non-symmetric footprint.
PTPID cannot cope with obstacles but is able to accurately follow the original global plan. Also, its algorithm is deterministic and it produces reliable results.

Based on their control concept and the test observations, each controller was scored on various criteria, resulting in a concise selection guideline; see Table 1. DWB is seen to have a good balance among the selected criteria, consistently providing obstacle avoidance behaviour. RPP performs well for robots with small footprints in a known environment and is computationally simpler than DWB. PTPID specialises in accuracy and reproducibility, and finally, TEB offers great flexibility but requires higher computational power and may lack robustness in specific situations.

Ultimately, proper selection of a motion controller greatly depends on the application requirements.



10 Test results in terms of execution time for the three tests; for each test, SARA results on the left and IDA results on the right. Results above the horizontal dashed line indicate failure.