Safety and path planning for collaborative applications based on an autonomous mobile platform

Nicolas Gautier[1], Peter Kmecl[2], Marko Munih[2] and Janez Podobnik[2]

ENIB, Brest, France [1] University of Ljubljana, FE, Ljubljana, Slovenia[2]
E-mail: nicolas.gautier29@protonmail.com

Abstract

This paper presents the improvement of a mobile robot platform developed in the Laboratory of robotics at the Faculty of Electrical Engineering. The final application will be a collaborative mobile manipulator for indoor environment such as warehouse or retail store. The main focus of this paper is the implementation of ADS communication for data exchange between the ROS controller and the embedded computer, the use of external setpoint generator for motor commands, the development of safety using laser scanners and autonomous navigation using the ROS (Robot operating system) navigation stack.

1 Introduction

Mobile robots are becoming more and more common, especially with the Industry 4.0. They are distinguished from other robots by their ability to orient themselves and navigate in an unknown and changing environment thanks to their sensors and artificial intelligence algorithms. This makes the robot capable of adapting to new situations and performing the tasks assigned to it. They are most often used in industrial applications to transport and handle materials and supplies, but also to assist humans in their work and even to substitute them in difficult, repetitive and dangerous tasks [1] [2].

Originally used in industrial applications, their scope has expanded in recent years. They are now used in many different areas such as museums, airports, warehouses, etc. They will also play a key role in the future of agriculture and medicine, which are key sectors that humans will have to transform in the next few years [3] [4].

Mobile platforms can be combined with robot manipulators to form mobile manipulators which benefit from the advantages of both types of robots. The robot can move safely through its environment and be positioned anywhere in the workspace to perform a variety of complex and precise tasks. It can perform production and manufacturing applications that previously required multiple stationary robots or a dedicated worker [5].

The mobile platform [6] shown in Figure 1 uses a Beckhoff CX5140 PLC (Programmable Logic Controller) to supervise the low level control of the platform (motors, joystick, laser field safety) and a high performance computer running a ROS controller that uses high level algorithms (path finding, SLAM, etc). The platform is equipped with 4 Mecanum wheels for holonomic movements, 2 SICK NanoScan3 laser sensors used for safety and mapping, a joystick to control the platform in manual mode and a battery to make the platform autonomous.

Figure 1: Mobile platform developed in the laboratory.

2 Upgrade and implementation

The mobile platform will be used in a collaborative application. Therefore, safety had to be improved to prevent any danger when it will be in contact with humans. To enable autonomous navigation, path planning algorithms had to be implemented. It was also relevant to improve the responsiveness of the platform through the control of the motors and the communication between the low and high level programs.

2.1 ADS communication

We considered replacing the current UDP communication with ADS communication to exchange data between the Beckhoff controller and the ROS controller. TwinCAT ADS "Automation Device Specification" is an interface created by Beckhoff for data exchange between TwinCAT modules. All messages are transmitted over a TCP/IP connections.

In our application, the ROS program needs the odometry of the platform for the navigation algorithms. The odometry is computed from the wheel motor encoders on the PLC controller. The odometry must be sent periodically, so we use ADS notification which allows TwinCAT variables to be received periodically without request.

We also need to send ROS velocity commands to the PLC to move the platform. Instead of sending each variable one by one, the ADS interface allows us to exchange data arrays up to 500 variables to reduce the communication time.

2.2 External setpoint generator

To control the motors, TwinCAT offers the possibility of using our own control signals (position, speed, acceleration) through an external setpoint which replaces the internal setpoint. This allows us to have more control on the trajectory and especially on the acceleration and the jerk. It also reduces the delay between the sending of the command and its execution.

For each wheel, we calculate the required acceleration α which is the difference between the desired angular speed ω_{goal} and the current angular speed ω_0 obtained using the encoders over one period of the program Δt (1).

$$\alpha = \frac{\omega_{goal} - \omega_0}{\Delta t}. \tag{1}$$

If the acceleration is too high to suit our application (smooth movement and avoid jerking), we clamp it. Then we integrate twice using the linear approximation to obtain the angular velocity ω (2) and angular position ϕ (3).

$$\omega = \alpha \Delta t + \omega_0, \tag{2}$$

$$\phi = \omega \Delta t + \phi_0. \tag{3}$$

2.3 Safety laser scanners

To enable autonomous movements, two sick nanoscan3 laser sensors were placed at two opposite corners of the mobile platform to allow a 360° view around the platform. The objective is to have a safety field around the platform. This allows warning fields for adapting the speed limit of the platform depending on the triggered field and to safely stop the platform (in software). In last resort in case of a very close obstacle this also allows to disconnect the motors from the power to stop the platform.

The set of warning fields used for the two sensors depends on the direction in which the platform is moving. To do this, we calculate the angle θ between the velocity vector v and the x axis (4).

$$\theta = arctan(\frac{-v_y}{v_x}). \tag{4}$$

There are five sets of warning fields for each sensor, for all the possible direction in which the platform is moving (see Figure 2). For each set there are three warning fields (20 cm, 1 m and 2.5 m). These fields are used to determine the nearest obstacle and adapt the speed limit. There is no specific field for rotation because in this case the platform will not move over large distances and the default fields will be sufficient.

There is also a permanent set of fields all around the platform (see Figure 2d.), consisting of the single protective field that disconnects the power if it is triggered. This field is used for obstacles within 16 cm. The set also includes a warning field (18 cm), which has the same function but stops the platform in the TwinCAT program.

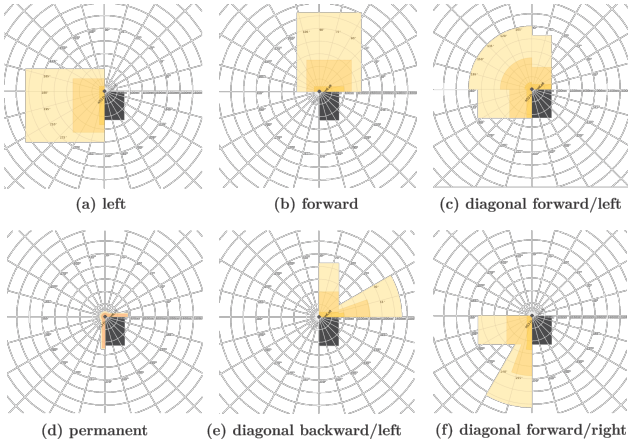

Figure 2: Set of fields used for a laser scanner mounted front left, the fields are active depending on the direction of the platform. Grey rectangle represents the platform.

2.4 ROS navigation

ROS navigation provides packages for autonomous navigation and trajectory planning. It uses odometry computed from the wheel encoders and the data from the two laser scanners.

For localization we use ROS rtabmap package [7] which is an implementation of VSLAM algorithm (Visual Simultaneous Localization And Mapping). It uses the sensors and odometry of the platform to build and clear a map. The map is recreated each time the program is started because the platform will be used in different places and should be able to adapt to new environments.

ROS navigation uses two path planners. The global planner computes a global trajectory to reach the final position. It usually implements shortest path algorithms such as Dijkstra or A* algorithms [8]. In our application we use the default settings of the ROS navigation global planner [9].

The local planner computes the final trajectory that tries to follow the global planner but handles local static and dynamic obstacles. For local planner we use ROS TEB (timed elastic band) local planner package [10]. The algorithm optimizes the robot trajectory according to the execution time, the distance to static and dynamic obstacles and with regard to the platform dynamics. The user can adjust the weights of the different parameters to best fit the desired behavior. This planner is linked to a local cost map that is regularly updated with sensor values. The resolution is low and the map is small to avoid a long computation time.

Figure 3 shows an example trajectory using the entire navigation stack described above. The global cost map (grey field) was created after a few moves of the platform in the lab. The green trajectory is the path calculated by the global planner to reach the final position. The local cost map (white square field) shows the nearest obstacles and their inflation radius. The local planner creates a local trajectory (red path) that follows the global trajectory but adapts it to the local cost map information and user

parameters.

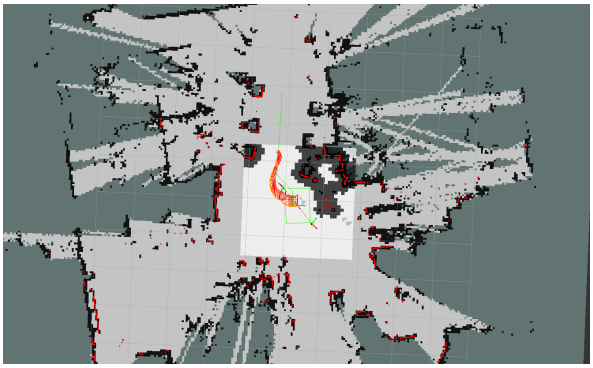

Figure 3: Maps and trajectories using the navigation stack.

3 Evaluation and testing

3.1 ADS communication

Tests were performed to evaluate the frequency, delay and jitter of the ADS communication for transmission. We sent a large number of arrays and measured the time required for transmission. Figure 4 shows the final result, the data was transmitted with an average of 1kHz with some peaks at 500 Hz. The results are more consistent and stable than the previous UDP communication.

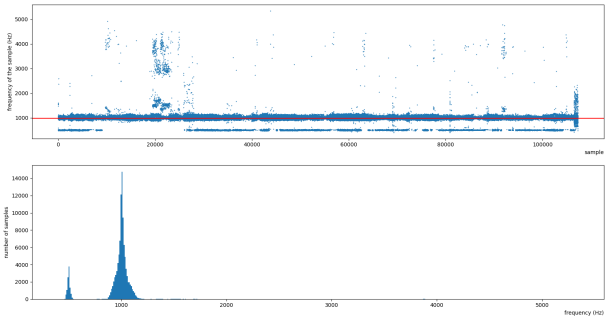

Figure 4: ADS communication frequency experiment.

3.2 External setpoint generator

The movements using the external setpoint generator are smooth and there are no jerks. This allows us to have better control over the maximum acceleration limit. This is important because the platform is quite heavy and high acceleration could represent a danger in a collaborative application. The response time is a few milliseconds and the platform can stop almost instantly in case of an obstacle. The movements are accurate to within 2-3 cm.

3.3 Safety laser scanners

Tests were conducted to ensure the reliability of the system by placing obstacles such as boxes and chairs in the path of the platform and verifying that the program stops the platform. We also tested with dynamic obstacles (human cutting the path) when the platform was moving at a higher speed to ensure that it prevented collisions.

3.4 Navigation behavior

We performed many tests in different configurations to adjust the local planner settings to best suit the desired behavior:

- Smooth trajectory without jerking and large accelerations.
- Adapt speed in the presence of obstacles and keep a reasonable distance from them.
- React to dynamic obstacles by slowing down and finding a new path around them if necessary.
- Navigate in narrow spaces without triggering the safety fields.
- Precision of few centimeters on the final position due to the future use of a manipulator.

3.5 Navigation accuracy

Measurements were performed to evaluate the accuracy of the navigation to reach its final position. We wanted to compare the odometry calculated from wheel motor encoders, the position computed by ROS navigation stack which combines the odometry and the laser sensor data.

For the reference position of the platform, we used the laboratory's Optotrak Certus NDI motion capture equipment, which has high accuracy of 0.1 mm. The camera was placed above the platform and aimed at a 3 x 4 m area where the platform was able to move. Three markers were placed on the robot to be able to compute the x, y and θ coordinates.

50 experiments were performed in a row with the same trajectory loop to observe the shift of the behavior over time and especially to ensure that the navigation remained accurate. Figure 5 shows the first trajectory. We can observe that the odometry is initially quite close to the real position (about 15 mm).

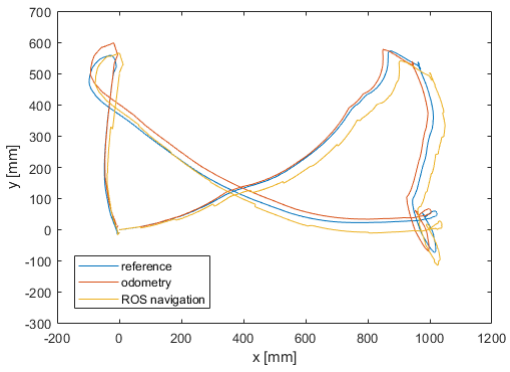

Figure 5: Comparison of odometry, transforms and real position on the 1st trajectory.

Figure 6 shows the 46th trajectory where the odometry is completely off from the real position (more than 100 mm). At the same time, the accuracy of the ROS navigation remains constant with an average difference of 25

mm compared to the real position. We can also notice that the trajectory rotates from initial trajectory, this is due to the small difference between the initial and final orientation that accumulates over the experiments (commands are sent in the local coordinates system of the platform).

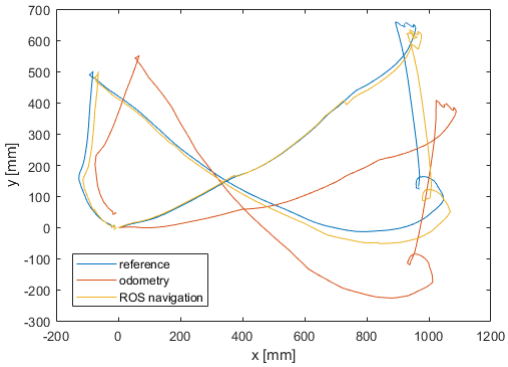

Figure 6: Comparison of odometry, transforms and real position on the 46th trajectory.

We have plotted the accuracy to reach the desired position by comparing the initial position and the final position for each experiment. Figure 7 shows the results, we obtained an average difference of 13 mm which does not deteriorate over time. This is a good result for our application because we wanted an accuracy of less than 50 mm for the manipulator.

We also plotted the average difference between the odometry and the transforms compared to the real positions. The odometry diverges linearly from the real position and quickly becomes unusable while at the same time the transforms remain around 25 mm from real position.

We observed similar results with the orientation. An average difference of 0.05 °between the transforms and the real orientation and an average difference of 0.17°on the final orientation.

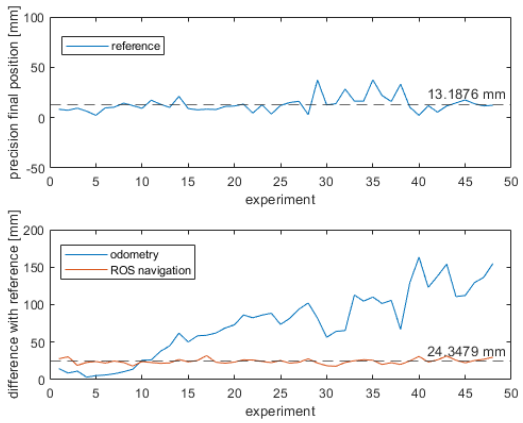

Figure 7: Accuracy of the platform to reach the final position; Precision of the odometry and its correction over the experiments.

4 Conclusion

The use of TwinCAT ADS and the external setpoint generator has improved communication between the low-level and high-level program, increased the responsiveness of the platform, and providing better control over the motor commands.

The safety improvement allows the platform to be used in collaborative applications. It has prevented any collisions with humans and obstacles, even in difficult scenarios.

The ROS navigation stack enable the platform to navigate autonomously in its environment. It can move both on large areas and in narrow passages. It can also adapt its trajectory to the presence of static and dynamic obstacles and can reach the targeted goal with accuracy.

References

[1] Spyros G Tzafestas. *Introduction to mobile robot control*. Elsevier, 2013.

[2] Giuseppe Fragapane, Dmitry Ivanov, Mirco Peron, Fabio Sgarbossa, and Jan Ola Strandhagen. Increasing flexibility and productivity in industry 4.0 production networks with autonomous mobile robots and smart intralogistics. *Annals of operations research*, pages 1–19, 2020.

[3] Hani Hagras, Martin Colley, Victor Callaghan, and Malcolm Carr-West. Online learning and adaptation of autonomous mobile robots for sustainable agriculture. *Autonomous Robots*, 13(1):37–52, 2002.

[4] ZR Struzik, K Yoshiuchi, M Sone, T Ishikawa, H Kikuchi, H Kumano, T Watsuji, BH Natelson, and Y Yamamoto. “mobile nurse” platform for ubiquitous medicine. *Methods of Information in Medicine*, 46(02):130–134, 2007.

[5] Biao Zhang, Carlos Martinez, Jianjun Wang, Thomas Fuhlbrigge, William Eakins, and Heping Chen. The challenges of integrating an industrial robot on a mobile platform. In *2010 IEEE International Conference on Automation and Logistics*, pages 255–260. IEEE, 2010.

[6] Peter Kmecl, Matjaž Mihelj, Marko Munih, and Janez Podobnik. Vodenje sodelujoče mobilne robotske celice. *ERK*, 2020.

[7] Mathieu Labbe. *ROS RTAB-Map*. `http://wiki.ros.org/rtabmap_ros`.

[8] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018, 2018.

[9] David Lu. *ROS global planner*. `https://wiki.ros.org/global_planner`.

[10] F. Hoffmann C. Rösmann and T. Bertram. *ROS Timed Elastic Band local planner*. `https://wiki.ros.org/teb_local_planner`.