Dynamic path planning in human-shared environments for low-resource mobile agents

Dynamic path planning in human-shared environments for low-resource mobile agents

Pangcheng David Cen Cheng, Marina Indri, Federico Maresca, Antonio Ragazzo, Fiorella Sibona
Dipartimento di Elettronica e Telecomunicazioni
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
{pangcheng.cencheng, marina.indri, fiorella.sibona}@polito.it
fmare94@gmail.com, antonio.ragazzo@outlook.it

***Abstract*—Mobile agents are widely employed in industrial applications. However, in an environment where humans and robots coexist, the robot should be able to navigate autonomously while safely avoiding moving obstacles, especially human operators.**

In this paper, we propose a dynamic path planner providing an additional costmap layer in which the area around the detected human is inflated by a Gaussian cost. The latter is proportional to the obstacle's speed and orientation, leading to a safer avoidance behaviour during navigation. The algorithm, suitable for low-resource mobile agents, has been developed in ROS1 and then experimentally validated on the Locobot mobile manipulator in a laboratory environment.

***Index Terms*—Dynamic path planning, human-obstacle avoidance, human detection**

I. Introduction

Navigation algorithms for mobile robots have been evolving quickly to satisfy the industry's requirements in terms of productivity, flexibility and safety. Among the intelligent mobile agents commonly used in an industrial context are Autonomous Mobile Robots (AMRs) and mobile manipulators. The first ones are usually employed to monitor and transport materials, while the second ones can also manipulate objects and perform complex tasks. Both robotic systems are required to navigate autonomously in the working environment, implying that they should be able to carry out the following tasks: (i) to map the environment and localize themselves within it, (ii) to plan an optimal route to the goal position, and (iii) to sense the environment so as to avoid collision with obstacles.

Currently, most of the robotic applications are developed with ROS (Robot Operating System). Within this framework, path planning methods are developed on two levels: global and local planners. The former considers the information provided by an occupancy map and generates a feasible path connecting the starting and final poses. On the other hand, the latter modifies the robot's global plan when an obstacle is detected, with the aim to overcome such an obstacle and resume the motion to reach the desired destination.

The choice of the global planning method depends on the navigation requirements and environmental constraints. For what concerns the local planners, they are often referred to as dynamic path planners, since their task is to deviate the robot when dynamic obstacles intersect its trajectory. How much the robot should deviate depends on the safety standards in the working place; in general, when the robot must overcome a human obstacle, it should proceed with a larger radius with respect to other obstacles. Other factors to be taken into account are the reaction time when the obstacle is detected and the precision with which a new path is replanned.

A dynamic path planner presented in [1] is able to increase the occupancy grids of generic moving obstacles, allowing the robot to anticipate their motion and plan a more optimistic path to reach the goal position. The approach has been developed in ROS2, but it was validated only in simulation. Also, the authors of [2] propose a costmap-based collision detection and avoidance algorithm (tested in simulation), which enlarges the obstacle footprint by convolution with a Gaussian distribution and inflates the obstacle's frontal area. Despite the latter being presented as a socially acceptable behaviour, the proposed method does not provide an actual identification of the encountered dynamic obstacles.

In fact, most of the traditional path-planning algorithms do not make any distinction between human obstacles and generic ones. In an environment where humans and robots coexist, safety is a crucial factor to be considered in the path-planning process of mobile agents, particularly mobile manipulators, since they may carry objects during motion. Various algorithms are available for taking care of human obstacle detection, as illustrated in [3], where the most used sensors and algorithms deployed in industrial robots to perceive the human worker are analysed. In particular, combining data from different sources, e.g., RGB-D cameras and LIDARs, allows the mobile robot to obtain robust measurements of the environment, as well as awareness about the presence of human operators.

In [4], a human-aware path planning is proposed. Specifically, it deploys a modified version of the Timed-Elastic Band (TEB) [5] algorithm, the Human Aware TEB, in which human-aware constraints are included in the optimization algorithm. Furthermore, to avoid collisions with high-speed obstacles, the authors in [6] introduce an additional obstacle layer integrated with a cut-off distance, which depends on the speed of the moving objects. Despite their approach exploits an object detection algorithm, they apply the same obstacle layer for both generic and human obstacles in the simulation and real-world tests.

A supervisory planner algorithm, previously proposed in [7], intervenes at a higher planning level, keeping the robot navigating in a safe route, while avoiding human obstacles. However, the online replanning process may take a longer time, since it does not embed a mechanism to quickly react to dynamic obstacles.

The aim of this paper is to propose a dynamic path-planning method for a mobile agent, to make it able to navigate autonomously and safely in an environment shared with human operators. The proposed approach is developed to be applicable to low-resource mobile agents, starting from the analysis of the state-of-the-art solutions for the main issues to be addressed (i.e., human detection and tracking, path prediction), and adding a social navigation costmap layer for a safe human avoidance. ROS1 has been preferred to exploit the greater amount of libraries already available in the worldwide community, and provide an approach useful for several potentially interested users, experimentally tested on a Trossen Robotics Locobot mobile manipulator [8].

The remainder of the paper is structured as follows: Section II unfolds the development steps of the proposed approach. Then, Section III presents the experimental setup and illustrates the experimental testing outcomes. Finally, Section IV draws some conclusions and open issues, as well as future works.

II. Development of the proposed safe dynamic path planning architecture

The proposed approach addresses the problem of path planning in dynamic environments shared with human co-workers when low-computational power mobile agents are used. Even though the algorithm concepts are platform independent, the implementation is based on a Locobot mobile manipulator [8], allowing to provide a working example for the research community. Some preliminary results have been presented in [9], whose code is available at [10]. Further details regarding the specific Locobot model and sensors/hardware setup can be found in Section III-A. The proposed method makes use of a costmap with an added navigation layer. This layer, similarly to the inflation layer of standard costmaps, contains an inflated Gaussian-shaped cost around humans, detected by exploiting both LIDAR and RGB-D information.

This section provides a top-down description of each element of the approach, with the relative choices of available methods and ROS1 packages, and some considerations about parameter settings. Figure 1 gives an overview of the proposed dynamic path planning approach.

A. Human detection and tracking

When taking into consideration humans as obstacles, the velocity direction may change unexpectedly, with no connection to current velocity and pose. Thus, the robot must be able to correctly detect human obstacles and react, dynamically changing its path plan and adapting it to the new detected conditions. Various techniques exist to tackle the human detection problem. In a highly controlled and predictable setting, the use of QR codes or other easy-to-read markers on both human and non-human obstacles would be a safe and optimal solution, since it removes the guesswork and CPU load of machine learning and other probability-based techniques. However, this reduces the plug-and-play capabilities of the software, requiring prior environment preparation by placing markers on objects and obstacles, with the risk that some obstacles may become invisible to the robot in case the identifying code is misplaced or not placed at all. Nevertheless, in shared workspaces, successfully detecting human obstacles is crucial for safety reasons.

1) Recognition: Object recognition has been a rich field of study ever since the birth of Viola-Jones Detectors [11], which use Haar wavelet to look for facial features, and Histogram of Oriented Gradients (HOG) [12], which identify objects by looking at the distribution of edge directions. Neural networks have emerged as a natural fit for this task and are widely used in industrial applications [13], together with traditional image processing [14]. Also, they usually require a GPU for real-time detection, but are stable and can identify humans where other more traditional methods fail. In particular, convolutional neural networks are at a mature stage of study, and many models are being used for human recognition. Recently, for real-time human detection, speed up in algorithms together with hardware improvements have made possible to use these algorithms in an online fashion [15]. Other least computationally intensive classifiers can be used, e.g., Random Forest (RF), HOG, Haar Cascade.

The Locobot has an Intel NUC mini desktop with no dedicated GPU, which led to narrow down the selection to algorithms that could be reliable, while needing as few resources as possible. Therefore, a RF classifier for the LIDAR sensor, coupled with a HOG-based Point Cloud Library (PCL) detector [16] for the RGB-D sensor has been chosen for human detection.

Not only objects should be recognized but, if they are moving, they must be tracked to update the movement prediction that is being made online.

2) Path Prediction: Obstacle tracking and path prediction are relative to the robot's ability to analyze obstacle information and predict the obstacle movement direction or goal; this is much more complicated than mere recognition, since the path prediction may change together with the type of object. Path prediction can be subdivided into 3 major methods: (i) physics-based, (ii) pattern-based, and (iii) planning-based [17]. In particular, the first methods use some dynamic equation to predict human motion, without taking into account human decision-making, and by simply modelling people as static linear velocity objects. Note that a linearization of previous tracking information to infer future paths is simple but rudimentary and can lead to errors, especially when tracking humans. Instead, learning human movement patterns with neural networks is effective but complex and hardware intensive. Also, datasets and annotation can be difficult for this kind of pattern learning, since the amount of data to be learnt is much larger than, for example, a simple colour image for a classifier.

At first, we tested the Leg Detector package introduced

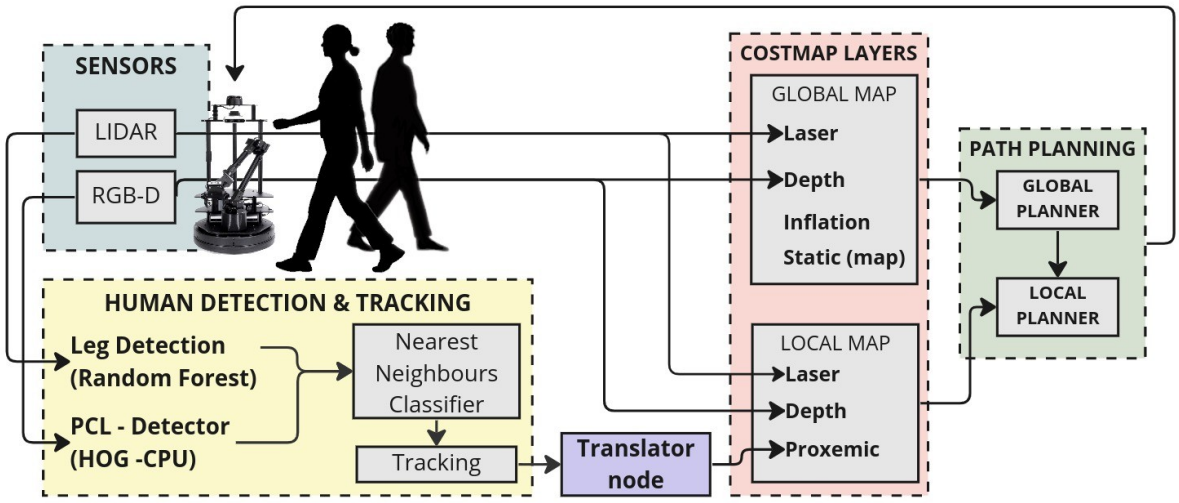

Fig. 1: Dynamic path planning overview.

in [18], as part of the People package, a software stack that contains various algorithms used for people tracking and detection. Nonetheless, the available Leg Detector resulted in a high amount of false positives. Also, its implementation based on Python affected the performance of other ROS nodes.

Thereby, the package used for human detection and tracking comes from the Spencer People Tracking package, presented in [19] and [20]. Some of the detectors available in this framework can be leveraged by the Locobot, since they make use of low computational effort algorithms, like Nearest Neighbours and RF classifiers. The framework uses also a ground HOG classifier that, however, we could not use, since it requires an NVIDIA GPU with CUDA library capabilities. The detection and tracking models are compatible with our sensors, but were both trained and tested using other sensors with different configurations. For instance, the RF model for the leg detector was trained with front and rear 2D LIDAR SICK LMS500 at 70 cm and 75 cm height, respectively, with a more accurate angle resolution. The RGB-D data were taken from an ASUS Xtion PRO LIVE at a height of 1.6 m, whereas the Locobot has a stereo camera at around 60 cm height.

To adapt the package to the available platform, settings have been edited as follows:

- To achieve satisfactory human detection and tracking, some parameters have been tuned. In particular, the settings in the *tracking_on_robot.launch* file have been adapted to the Locobot package.
- Their laser-based leg detector uses a random tree classifier model, trained on a 2D LIDAR SICK LMS500, which is positioned at a different height from the LIDAR in our setup. Nevertheless, by tuning the leg detector settings, stable human detection has been achieved. To correctly set the used model, the `model_prefix` parameter has been edited. Finally, it is noteworthy that the `decision_threshold` and `laser_max_distance` parameters were the most important settings to change in order to achieve detection.
- The PCL used in our implementation is modified from the one available in [16]. It uses a HOG support vector machine to identify full-body images of humans. We chose this detector since the other one in the framework, an upper-body classifier, needs a dedicated GPU to work. Moreover, to allow the detector to read the depth information from the sensors, input topics have been accordingly modified.
- Settings for the people tracker in the *freiburg_people_tracking.launch* file have been kept as the original ones. The People Tracker fuses detections coming from all available sensors, and uses a Nearest Neighbour data association filter to track detections.

B. Costmap Layers

Autonomous navigation of mobile platforms in human-shared environments must take into account how humans perceive and react to the motion of a robot in their proximity. To this aim, social navigation takes care of the path planning and behaviour of a robot in a social setting; it can include human behaviour prediction, such as reading social cues. This aspect of navigation also includes the study of proxemics, i.e., the study of human behaviour in space, social cues and interactions [21]. The most relevant aspect of proxemics is the robot's ability to respect a human's personal space and infer his/her movement through detection and tracking. Hall divided human interpersonal distances from closest to farthest, namely, *intimate space*, *personal space*, *social space* and *public space*. To respect these spaces, a generic obstacle can be passed by more closely, while the human must be kept farther away.

1) Social Navigation: A work presented in [22] divides path planning for social navigation into *reactive planning* and *predictive planning*. Reactive planning takes into account, at all time steps, every possible movement of the robot and removes any movement that may cause a collision. This planning approach is the most conservative and safe, but it cannot take into account human behaviour or read social cues.

It can also incur into the Freezing Robot Problem [23], where the robot stops moving, since any course of action could lead to a collision. On the other hand, predictive planning tries to model human behaviour to plan a path optimally when near people and moving in a "human-friendly" way. However, being a probabilistic approach, predictive planning can lead to a collision, if the movement prediction is wrong and the robot cannot stop in time to avoid it.

The work presented in this paper implements a reactive approach to human-obstacle avoidance, together with a physics-based method for path prediction. To do so, a social navigation layer has been added for human avoidance during path planning with a Gaussian-based costmap approach. The Social Navigation Layer package [24] adds two custom layers to the Costmap 2D package [25]. Namely:

- `social_navigation_layers::ProxemicLayer`: This layer uses the theory of proxemics to add a Gaussian cost to the detected humans; this cost forces the path planner that reads it to take this cost into account during the path calculation.
- `social_navigation_layers::PassingLayer`: This layer introduces a similar cost to that of the proxemic layer, but adds a Gaussian cost area to a desired side of the detected human. This way, if the robot has to pass by the human, it is forced to plan a path on the other side.

The Gaussian cost added through the `Proxemic Layer` is deformed by the velocity of the person. In fact, when stationary, the Gaussian results in a circular shape with a cost that decreases along the radius. Instead, when a person is moving, the velocity deforms the Gaussian function towards the direction of movement, resulting in an oval shape that envelops the space in front of the person. This is done to take into account the temporal aspects of a moving obstacle, so as to discourage path plans that use the cells in front of it. The faster the speed, the more deformed the Gaussian.

The described layers use information from the */People* topic to add an inflated cost to the obstacles recognized as humans. The */People* topic is provided by the People package, exploited here for its custom ROS messages functionalities.

C. Tracked People Translator Node

Having set up the previously described packages, we needed a way for the detections to be seen by the social navigation layers. The Spencer People Tracking package uses its own ROS messages to publish detections and tracking information, and publishes them to the */spencer/perception/tracked_persons* topic. On the other hand, the Social Navigation Layers package expects detections to be published to the */People* topic. To merge these two packages we implemented a C++ ROS translator node, named *tracked_people_translator*. This node is launched together with the detectors and subscribes to */spencer/perception/tracked_persons*. When a new set of detections is published, the Translator node reads them and translates them to *people_msgs/People* messages. Detections are only translated and published if they contain a matched ID, that is, if the track is currently matched by a detection. These are then published to the */People* topic.

D. Path planning and navigation

For our implementation, the standard ROS1 Navigation Stack has been used. Specifically, the global and local planners have been chosen as follows:

- Global Planner: the standard *navfn* package comes by default as a global planner package; however, it only allows for Dijkstra's algorithm for path calculation. On the other hand, the *Global Planner* package has the option of using the A* and has much more flexible settings. Note that there is a trade-off between the two path planners: Dijkstra is better for smoother and shorter paths, while A* leads to faster computing time. For a dynamic environment, A* is better, since it is able to recompute a global path when it encounters a moving obstacle so large to prevent the local path planner to keep the global plan within the set constraints. However, A* paths tend to be angular and non-natural, taking the shape of stair-like patterns (causing most of the distance losses with respect to Djikstra).
- Local Planner: depending on the application requirements or limitations imposed by the available hardware/software, one may choose among the three available local planners: Dynamic Window Approach (DWA), Elastic Band (EBand) or TEB. It is worth noting what follows: DWA requires fewer computational resources and has high repeatability, the results obtained from the EBand are more accurate, while the TEB has a quick reaction to dynamic obstacles but requires more computational power [26]. Moreover, the authors in [27] pointed out that the TEB performed better in different test scenarios, while the DWA showed some issues when dealing with dynamic obstacle avoidance, and the EBand had difficulties computing local path in the case of static obstacles. The local costmap, based on which the local plan is computed, has been integrated with the `Proxemic Layer`, by adding it to the plugin list in the configuration files and suitably adjusting its parameters. Among these, the most relevant are the amplitude and the factor parameters.
  - `amplitude` scales the size of the Gaussian. If this value is too small, the inflation radius will overtake the proxemic inflation and this plugin will have no effect; if it is too large, the path planners will have trouble finding an optimal path.
  - `factor` is a multiplicative factor affecting the Gaussian function deformation in a directly proportional manner. A too small value will result in a circular Gaussian even when a human is moving at fast speeds, while a too large value will lead to overcorrection.

  The default values were too high for our environment. An amplitude value of 77 meant that the Gaussian generated occupied most of the corridor available for testing, imposing more than $2.5\,\mathrm{m}$ of free space between the robot and the human.

This caused the robot to always stop when encountering a person, since a too limited space was left for replanning. We found that a value of 40 was more reasonable, and to increase the effect of speed on the shape of the Gaussian we doubled the value of the factor to 10. The settings used for the plugin are reported in Table I.

Social Layer Parameters	Default Value	Set Value
`enabled`	True	unvaried
`cutoff`	10.0	unvaried
`amplitude`	77.0	40.0
`covariance`	0.25	unvaried
`factor`	5.0	10.0
`keep_time`	0.75	unvaried

TABLE I: Social Layer plugin parameter settings

A good example of the combination of global and local planners is the one proposed in [27], which combines the A* global path planner with a local planner, allowing for path smoothing that shortens the resulting path, either TEB or EBand. In our approach, we decided to employ the A* as the global planner and the TEB as the local planner, since it complements the global path planner A* well, by smoothing the global plan's stair-like pattern.

III. Experimental validation

A. Hardware and software setup

The Locobot WX250 mobile manipulator by Trossen Robotics, whose technical specifications are available in [8], has been chosen to test the proposed approach. In particular, it is composed of a Kobuki mobile platform and a WidowX250 6-DOF manipulator. The mobile platform has differential wheels and active bumpers to sense if there is any contact with obstacles. Moreover, it is equipped with an RPLIDAR A2M8 (360° 2D LIDAR) and an Intel RealSense D435 (Stereo RGB-D), used for both manipulation and navigation tasks.

As previously stated, the proposed approach has been developed on top of ROS1 packages. Indeed, exploiting the ROS middleware avoids rewriting the entire software stack in case of slight modifications in hardware and driver packages. A ROS-based solution favours the reuse of the provided code with other setups. It is worth noting that the RTAB-Map (Real-Time Appearance-Based Mapping) algorithm [28] has been employed to solve the SLAM problem. In fact, this algorithm allowed us to take advantage of the complete set of sensors the mobile robot is equipped with (unlike other tested packages, e.g., SLAMToolbox [29], which exploits 2D LIDAR data only).

B. Testing

The proposed approach has been tested in a research laboratory setup, whose map can be seen in Figure 2. A brief demo video of the approach experimental validation is available at [30].

Figures 3-5 highlight some of the features of the proposed dynamic path planning approach. In order to test the navigation safety when humans move near the robot, different scenarios have been considered, for instance, a human stopping in front of the mobile robot, and two humans moving at different speeds in the mobile agent's vicinity. Each figure shows a shot from the real scenario (on the right), and the corresponding view in `rviz` (on the left) for the three considered cases.

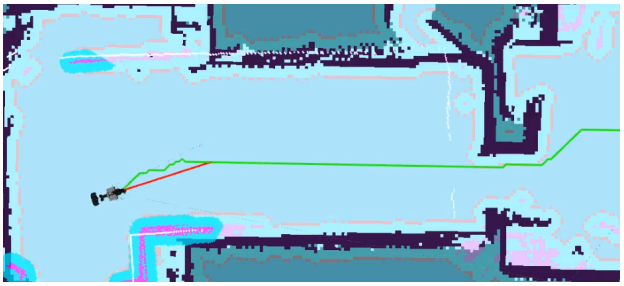

Fig. 2: Testing map visualization in `rviz`. Global path and local path plans are in green and red, respectively.

From Figure 3, it can be seen that the robot conservatively moves away from the human thanks to the Gaussian area, and replans the motion to reach the final goal. Note that the testing scenario has been enriched with static non-human obstacles (cardboard boxes). This was done to showcase how the planning reacts to generic obstacles, compared to how the Gaussian cost added on a detected human ensures the planned path to overcome the obstacle in a more conservative and socially acceptable way. Figure 4 and 5 capture the planner's reaction to the human moving at different speeds, showing how the shape of the inflated area is larger when the human moves faster, alerting beforehand the replanning procedure of the local planner.

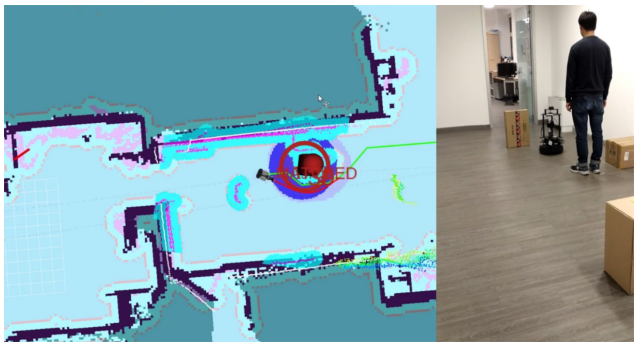

Fig. 3: Human stopping in front of the robot.

IV. Conclusions and future works

The work presented in this paper provides a human-centred dynamic path planning approach along with a working and open-source example for its use on a mobile agent, developed in ROS1 on a low-resource mobile agent and tested in a real context. The ability to detect human obstacles and distinguish them from generic ones allows to take advantage of shortest path algorithms when a safe distance is not necessary and, instead, to have more acceptable behaviour -from a human perception- when it comes to navigate in the vicinity of a human operator.

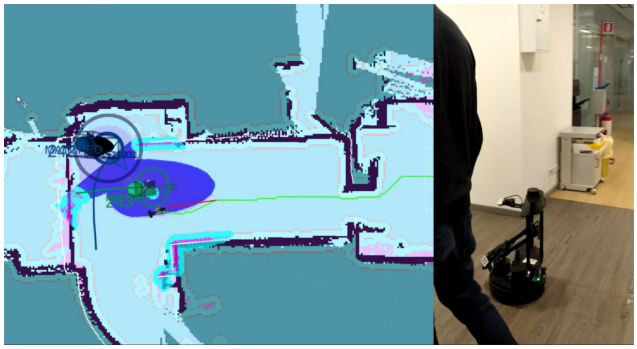

Fig. 4: Human walking slowly near the robot.

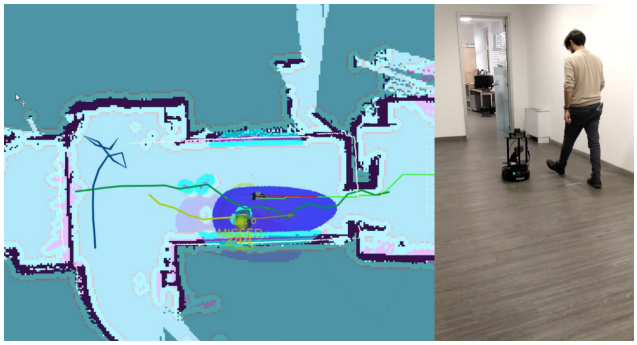

Fig. 5: Human walking fast near the robot.

However, a method like ours, which relies on the proper recognition of obstacles that are encountered on the path, can be affected by false positives and issues with tracking. So, a time-consuming but necessary improvement would be the creation of customized datasets for training. Furthermore, delay problems related to the low computational power of the on-board computer could be improved by upgrading to more powerful hardware, depending on the application requirements.

Finally, industrial applications need to be real-time and guarantee a certain level of security that ROS1 cannot provide. Therefore, future work is likely to prefer a ROS2 implementation, which is more industrial-oriented.

REFERENCES

[1] P. D. Cen Cheng, M. Indri, F. Sibona, M. De Rose, and G. Prato, "Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2022, pp. 1–8.

[2] C. A. Silva, S. Dogru, and L. Marques, "Mobile Robot Navigation in Dynamic Environments Taking into Account Obstacle Motion in Costmap Construction," in *ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics, Volume 1*. Springer, 2022, pp. 235–246.

[3] A. Bonci, P. D. Cen Cheng, M. Indri, G. Nabissi, and F. Sibona, "Human-robot perception in industrial environments: A survey," *Sensors*, vol. 21, no. 5, p. 1571, 2021.

[4] P. T. Singamaneni, A. Favier, and R. Alami, "Human-aware navigation planner for diverse human-robot interaction contexts," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5817–5824.

[5] "ROS Timed-Elastic Band," http://wiki.ros.org/teb_local_planner, [Online; accessed February 2023].

[6] C. S. Chen, C. J. Lin, C. C. Lai, and S. Y. Lin, "Velocity Estimation and Cost Map Generation for Dynamic Obstacle Avoidance of ROS Based AMR," *Machines*, vol. 10, no. 7, p. 501, 2022.

[7] M. Indri, F. Sibona, P. D. Cen Cheng, and C. Possieri, "Online supervised global path planning for AMRs with human-obstacle avoidance," in *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)*, vol. 1. IEEE, 2020, pp. 1473–1479.

[8] T. Robotics, "LoCoBot WidowX-250 6-DOF," https://docs.trossenrobotics.com/interbotix_xslocobots_docs/specifications/locobot_wx250s.html, [Online; accessed February 2023].

[9] F. Maresca and A. Ragazzo, "ROS-based autonomous navigation and object recognition for a mobile manipulator operating in a warehouse environment," Master's thesis, Politecnico di Torino, 2022.

[10] A. Ragazzo and F. Maresca, "GitHub repository," https://github.com/AntoRag/thesis, [Online; accessed February 2023].

[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[13] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie *et al.*, "Yolov6: a single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[14] R. A. Ciora and C. M. Simion, "Industrial applications of image processing," *Acta Universitatis Cibiniensis. Technical Series*, vol. 64, no. 1, pp. 17–21, 2014.

[15] Y. Li, Z. Hao, and H. Lei, "Survey of convolutional neural network," *Journal of Computer Applications*, vol. 36, no. 9, p. 2508, 2016.

[16] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, May 9-13 2011.

[17] C. Medina Sánchez, M. Zella, J. Capitán, and P. J. Marrón, "From perception to navigation in environments with persons: An indoor evaluation of the state of the art," *Sensors*, vol. 22, no. 3, p. 1191, 2022.

[18] "Leg Detector Package," http://wiki.ros.org/leg_detector, [Online; accessed February 2023].

[19] T. Linder, S. Breuers, B. Leibe, and K. O. Arras, "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 5512–5519.

[20] T. Linder and K. O. Arras, "People detection, tracking and visualization using ROS on a mobile service robot," in *Robot Operating System (ROS)*. Springer, 2016, pp. 187–213.

[21] E. T. Hall, "A system for the notation of proxemic behavior," *American anthropologist*, vol. 65, no. 5, pp. 1003–1026, 1963.

[22] J. Cheng, H. Cheng, M. Q.-H. Meng, and H. Zhang, "Autonomous navigation by mobile robots in human environments: A survey," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 1981–1986.

[23] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[24] "Social Navigation Layers Package Planner," http://wiki.ros.org/navigation_layers, [Online; accessed February 2023].

[25] Eitan Marder-Eppstein, "Costmap 2D Package," http://wiki.ros.org/costmap_2d?distro=noetic, [Online; accessed February 2023].

[26] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ros-based mobile robot," in *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 2019, pp. 104–111.

[27] M. Pittner, M. Hiller, F. Particke, L. Patino-Studencki, and J. Thielecke, "Systematic analysis of global and local planners for optimal trajectory planning," in *ISR 2018; 50th International Symposium on Robotics*. VDE, 2018, pp. 1–4.

[28] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831

[29] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.

[30] "Experimental test video demo," https://youtu.be/zqKbRZy0Ukw, [Online; accessed February 2023].