Navigation among movable obstacles in dynamic environments

Kai Zhang[1,2], Eric Lucet[1], Julien Alexandre dit Sandretto[2] and David Filliat[2]

***Abstract*— Solutions for navigation among movable obstacles (NAMO) usually take movable and static obstacles into account and are designed to complete the task in a static environment. However, taking dynamic obstacles such as coworkers into account could have a strong influence on the global navigation strategy. We propose a framework for NAMO in dynamic environments that considers the trade off between making detour, moving obstacles and waiting for dynamic obstacle to move in order to optimize the time to goal. In particular, we integrate the possibility to move backward to avoid the deadlock situations with dynamic obstacles using a goal generator based on reinforcement learning. The proposed method is evaluated in two simulated home and logistic scenarios and in a small scale real environment, showing its efficiency and ability to avoid deadlock situations.**

I. Introduction

Navigation among movable obstacles (NAMO) [1] usually considers that the environment is static and the robot is the only dynamic object in the environment. However, many situations can include dynamic obstacles (DOs) in the workspace, either humans or other robots. Failing to take DOs into account in the NAMO problem can lead to potential danger for both robots and humans, and deadlock in some situations where the robot could be prevented to reach its goal or to displace an obstacle as planned.

The main challenge of NAMO in dynamic environment that we address is to correctly take into account the changing danger zones linked to DOs. This danger zone is taken as the future trajectory of the DOs in our work, but can also include social preferences of humans [2]. Since the NAMO usually includes navigation and obstacle removal actions, the dynamic danger zones need to be considered in both cases (i.e. when planning path and deciding where to move obstacles) to guarantee safety and goal reachability.

The second problem we address is the blocking in front of humans in situations where we want the human to have priority over robots. During navigation, the robot can meet DOs in three broad scenarios: (i) moving on the same path in the same direction, (ii) on the same path in the opposite direction or (iii) on different but intersecting paths. There exists many approaches for navigation in dynamic environment (e.g., [3], [4] in the recent works), but a common limitation is that the robot can only make a detour or wait for the human to free the path. These strategies will work well in situations where there is space around the DO and its associated danger area. But when a passage is narrow, like in a corridor, it will only work in situations (i) and (iii). In situation (ii), if the robot encounters a human moving in the opposite direction, it will need to plan and perform a backward motion to free the human path before waiting for him to pass. To keep safety and plan efficient paths, all three behaviors (bypass, stop/wait and moving backward) should therefore be compared to choose the best option. An example can be found in Fig. 1. A particular difficulty for the backward motion is to decide where to move the robot so that it will free the human path while still being able to see when the human has freed the path.

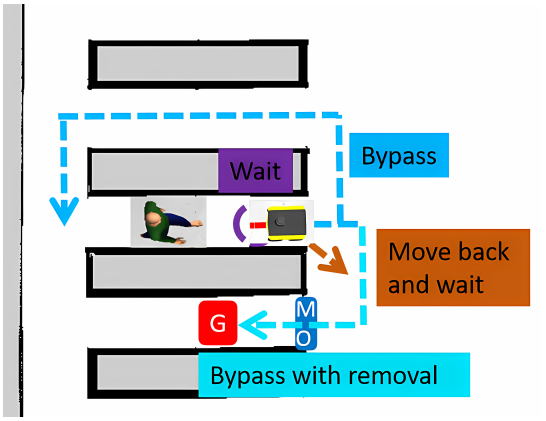

Fig. 1. Different reactions to the DO. Overall, three strategies can be applied when the robot meets a DO: bypass, stop and wait, move backward and wait. The red region is the goal while the dark blue region is the movable obstacle.

In this paper, we extend our previous work on NAMO [5] from static environment to a dynamic environment. We integrate a new module to predict DO trajectory in order to avoid collisions and prevent blocking human motions. This prediction is integrated in the prediction of stock region for movable obstacles (to avoid putting obstacles in DO paths), and in the global navigation strategy to decide between bypassing, stopping (already proposed in [5]) and a new moving backward behavior. For this, we enlarge the action space by integrating backward motions besides the going forward and stop actions, and propose a backward goal generator using reinforcement learning (RL) to free the DO path. We test our methods in both simulation and real environments in order to show its efficiency and capacity to avoid deadlock situations compared to a baseline NAMO algorithm and the standard ROS navigation strategy.

In the following, we position our work with respect to the related work before describing the approach in section III, and presenting results in simulation (Section IV) and on a

Website: kai-zhang-er.github.io/namo-dynamic

[1] List, CEA, Université Paris-Saclay, Palaiseau, France kai.zhang@cea.fr

[2] U2IS, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France

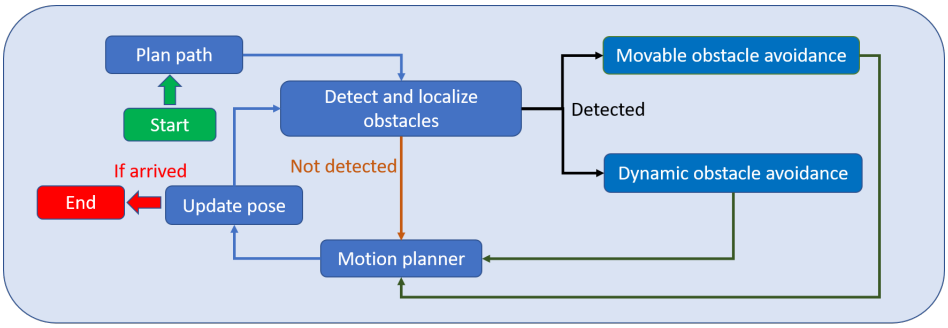

Fig. 2. System overview. The movable obstacle avoidance method is from [5] and we focus on the dynamic obstacle avoidance in this paper.

real robot (Section V).

II. Related Work

We review the recent works on NAMO and dynamic obstacle avoidance respectively and emphasize the specificity of our contribution.

A. Navigation among movable obstacles

NAMO algorithms are usually applied in an environment with static and movable obstacles (MO). Benefiting from the interaction ability, robots can navigate to the goal faster [5] or make the blocked navigation path passable [6], [7]. There are mainly two types of interaction when the robot meets the MO, either bypass it or remove it. Most NAMO algorithms use predefined strategies, which means to remove all MOs that block the path [8] or bypass them all if possible. However, some methods provide the choices for the robot based either on the estimation of the obstacle movability [9], [10] or on the estimated time necessary to move the obstacle [5]. If the robot cannot push the MO away or if it will take too much time to remove the MO, it is ordered to bypass the MO.

However, all the NAMO algorithms are applied in a static environment without the presence of any dynamic objects, like humans or other robots. Therefore, this paper aims to extend the NAMO proposed in [5] to dynamic environments.

B. Navigation in dynamic environment

Navigation tasks tend to be more complex in a dynamic environment that contains DOs, like robots or humans. The classical methods to avoid collision with DOs firstly model the motion of the DOs and control the robot to avoid their action space [11]. Since the behavior of the DOs, especially the humans, can be complicated and hard to model, machine learning based methods are proposed to either predict the action of DOs [12], [13] or teach the robot the proper reactions from its observation [14], [3]. For example, [12] predicts the intention of the DOs and controls the robot to avoid intrusion in the DO path while in [3], RL is used to learn to map observations to linear and angular speed directly for navigation in a crowd environment.

Most current methods are applied in environment with sparse obstacles so the robot could bypass the DOs, especially the humans. If there is space for detour, the robot bypasses the humans at a close distance or waits the DO to offer the path, which makes the coworking difficult and results in potential danger. In contrast, in our experiment, we give high priority to the DOs and optimize robot trajectories to avoid disturbing the DOs trajectories as much as possible.

III. Method

We start with the description of the task, then explain the structure of our system. In the method details, we focus on the introduction of our strategy for the dynamic obstacle avoidance.

A. Task description and method overview

The task is to navigate from a starting position to a goal position with the presence of static, movable and dynamic obstacles. The robot should avoid collisions with obstacles during navigation and complete the task while minimizing a cost, which corresponds to the task completion time in our experiment. We assume that the static map of the environment is known, since it can be easily obtained from SLAM methods (e.g., [15]), and that we have access to the robot position and the detection and tracking of movable and dynamic obstacles.

As shown in Fig. 2, given the start position and goal position, a path is planned based on the static map. Then the robot follows the path and detects the unexpected obstacles. If an obstacle is detected, the avoidance algorithm is applied according to the obstacle types, either MO or DO. After a series of reactions to the obstacles, the robot continues to follow the trajectory and iterates the above operations until reaching the goal.

For movable obstacles, the avoidance strategy chooses between bypassing and removal, which means the robot could move the obstacle to clear the path for navigation. The choice between the two options is made by predicting a potential MO destination using RL and estimating the removal cost using supervised learning, as detailed in [5].

For the DOs avoidance, we propose a novel approach, detailed in Fig. 3. When a DO is detected, its moving intention is estimated by predicting its future trajectory. The potential collision is identified by comparing the future trajectories of DO and robot. If there is intersection between the two trajectories, in other words, a collision may happen, we first estimate the time necessary to bypass the DO and to

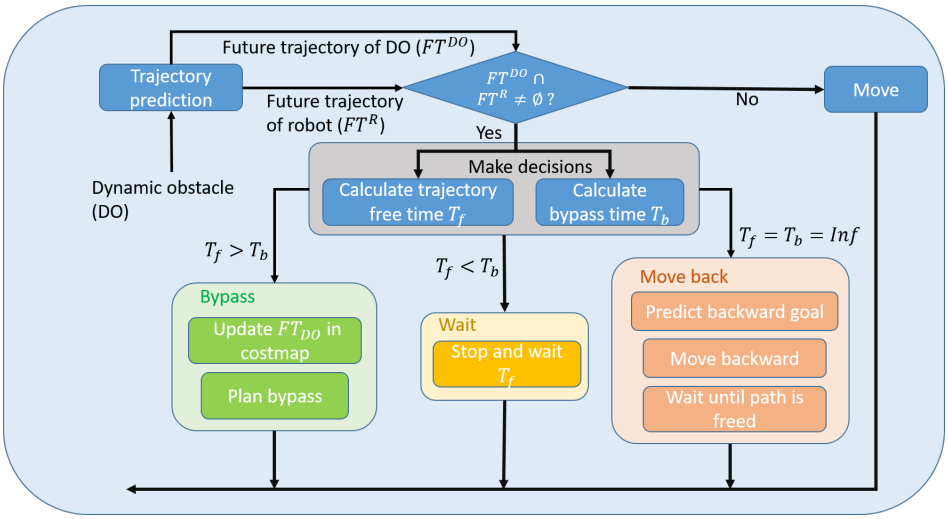

Fig. 3. Pipeline of the dynamic obstacle avoidance. Three reactions can be made to avoid the collision, including bypass, wait and move back.

wait for the DO to pass. If one of this option is feasible, the fastest one is chosen. If both options are infeasible, resulting in infinite time cost, we plan a backward motion to leave the path for the DO. Once the DO exits the path, the robot starts from the waiting point and continues the navigation task. With this implementation, MO avoidance and DO avoidance are handled independently. After moving backward, if the robot detects the MO, it will decide whether to remove or bypass the MO based on the time cost.

We detail all the components in the next sections.

B. Trajectory prediction

The trajectory prediction of DOs aims to estimate the moving intention to be used in the decision process. In our experiment, the intention of DO is affected by the static obstacles, such as walls that should be taken into account in the prediction. Besides, from the view of the robot, it could only observe the past trajectory of the DOs and has no knowledge of their goals. Considering these constraints, inspired by [16], we modify the dynamic window approach (DWA) [17] to perform the trajectory prediction task. The reason of using DWA is that it is simple and doesn't need any dataset for training. It can be replaced with any other more advanced trajectory prediction method.

We therefore use DWA to predict a trajectory with constant linear and angular velocities: we sample a set of trajectories and select the one with the lowest cost. We use an objective function with the assumption that the DO avoids obstacles and follows approximately constant motion so its linear and angular velocities do not change quickly:

$$F(v,w) = \alpha C_{clear}(v,w) + \beta C_{angle}(w) + \gamma C_{linear}(v) \quad (1)$$

where (v,w) are linear and angular velocities of the DO, C_{clear} is the time to collide with obstacles at the speeds of v and w, C_{angle} and C_{linear} represent the change of velocity compared to the mean velocities inferred from the past trajectory. α, β, γ are coefficients to balance the values. In our experiment, we set $\alpha = 1000$ to avoid the collision with static obstacles, $\beta = 30$ and $\gamma = 10$.

In the experiment, the DO is detected using an ArUco marker [18] to facilitate the estimation of DO pose. Besides, to reduce the impact of observation noise, we apply a Kalman filter on the observation of DO pose to estimate its current velocity. We predict the trajectory of the DO in next $T_{pred} = 8$ seconds.

C. Decision making

After obtaining the future trajectory of the DO, FT^{DO}, and of the robot (using the planned trajectory), FT^R, we check if their trajectories have intersections:

$$\min_i dist(FT_i^{DO}, FT_i^R) < TH_{dist}$$

where FT_i^{DO} and FT_i^R are the predicted positions of the DO and robot at time step $i \in [0, T_{pred}]$. TH_{dist} is a distance threshold of collision, we set $TH_{dist} = 1.0\,m$ in the experiment.

If the trajectories intersect, which means the DO and robot may collide later, the decision making module chooses a strategy between bypass, wait and moving backward, in order to minimize the time to goal.

To estimate the bypass time T_b, we mark FT^{DO} as an obstacle in the global map and use the global path planner to plan a shortest path to the goal avoiding the danger zone. The navigation distance is divided by the average speed of the robot to get T_b. If there is no detour to bypass the DO, the time is set at infinity, $T_b = Inf$.

The wait option means that the robot stops and waits the DO to pass first. The wait time T_w is calculated based on the FT^{DO} and FT^R. By iterating T_w from 1 to a maximum wait time T_{mw}, we compute the minimum distance between FT^{DO} and FT^R points and keep the time when $min(dist(FT^{DO}, FT^R)) > TH_{dist}$. As shown in Fig. 4, when $T_w = 2$, the distance between robot and DO is safe for navigation. If until $T_w = T_{mw} = 20s$, the condition

$min(dist(FT^{DO}, FT^{R})) < TH_{dist}$ is true , which means that the collision can still happen, the wait option is considered invalid and we set $T_w = Inf$.

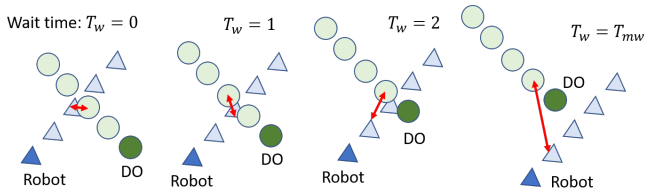

Fig. 4. Examples of calculating the wait time. The triangles and the circles are the future trajectories of the robot and the DO respectively. The red arrows show the minimum distance between two trajectories.

When both bypass and wait options are not practical, i.e., $T_b = T_w = Inf$, which usually happens in a narrow passage with the DO coming in the opposite direction of the robot path, the robot is ordered to move backward and free the path for the DO until the DO exits the narrow passage as described in the next section, before the robot continues to navigate to the destination.

D. Backward goal prediction

When the robot meets a DO in a narrow passage in the opposite direction, the bypass and wait options are impossible. The only solution for the robot is to move backward to a free place and leave a clear path for the DO. An ideal backward goal has three characteristics: (a) It has short navigation distance from the robot previous position. (b) The robot can observe the passage from its position so that when the DO leaves the passage, the robot can continue the navigation task. (c) The goal should be outside of the DO path so that the robot won't collide with DO at the backward goal.

We propose a method based on RL to search the backward goal, inspired by the method used for movable object displacement [5]. We train a policy shown in Fig. 5. The input includes three binary images:

1) Path map of the DO (M_{path}). The black region shows the path while the white region is outside of the path. Since there is no prior knowledge on the goal of the DO, it's path is based on the prediction presented in Sec III-B. Note that we assume that the goal of DO is outside of the passage so that the passage will eventually be freed.
2) Envelope map of the robot (M_{enve}). The white region shows the envelope of the robot. It could be of any shape and is drawn according to the orientation of the robot. It is also used for collision checking.
3) Visibility map of the robot at initial position (M_{visi}). The map is generated by using ray casting algorithm to find the regions that can observe the passage and DO. The white (resp. black) region is the region from which the DO path is visible (resp. invisible). It's important to find a backward goal in visible region so the robot can decide the time to enter the passage.

The output is a discrete action chosen among 7 backward moving directions. The interval between these directions is 30 degrees. The absolute angle value of the direction is related to the orientation of the robot and all of them point towards the back of the robot. We chose discrete actions for the smaller training time and easier convergence.

The policy network outputs an action a_i at each step and the robot moves backward in the corresponding direction at a fixed distance, 5 pixels (0.1m) in the map. Correspondingly, the $M_{path}, M_{enve}, M_{visi}$ are updated based on the new robot position and orientation (by modifying the images, without resorting to a simulator). Finally, when the robot leaves the path and remains in the visible region, the iteration stops and the current position of robot is considered as the backward goal. At each step, we compute a reward R_i:

$$R_i = \begin{cases} p_{collision} & \text{if } M_{enve} \wedge M_{visi} \wedge M_{path} \neq M_{enve} \\ 0 & \text{if success} \\ p_{step} & \text{otherwise} \end{cases} \quad (2)$$

When the robot enters the invisible region (black) from M_{visi}, a collision penalty is applied. The step penalty p_{step} is applied to encourage the RL to find the closest backward goal. $p_{collision} = -10$ and $p_{step} = -1$ in our experiments.

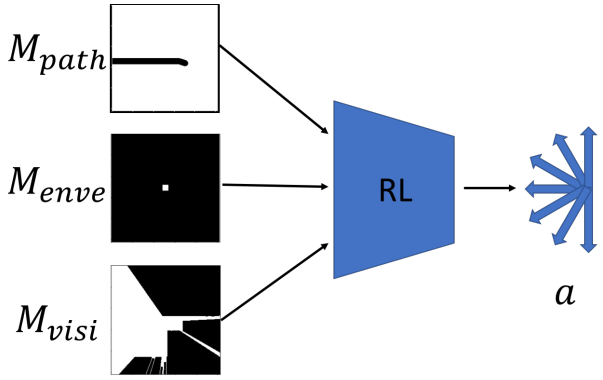

Fig. 5. Structure of policy for backward goal generator. The inputs include three binary images: path map M_{path}, envelope map M_{enve} and visibility map M_{visi}. PPO [19] is used for training and the actions are the directions of movement.

We use PPO [19] as the RL algorithm to train the policy. For training, we generate multiple 2D corridor environments that include random corridor angle, random robot position, random DO position, random size and shape of the robot. The training is performed in 2D maps without using simulation nor the real robot. The total number of training steps is 100 000 and the maximum number of steps per episode is 50. Training takes about 2 hours. During testing, in simulation or reality, the pretrained policy is applied to predict the backward goal online that is sent to the motion planner to guide the robot to this goal.

IV. Simulation Experiments

In this section, we first introduce the experiment environments. Then, in the ablation study, our RL based backward goal is compared with other methods. Finally, we demonstrate the overall performance of our method.

A. Simulation environments

We implement our system and simulate it in two Gazebo environments. As shown in Fig. 6, an office environment and a logistics environment are built to analyse the scalability of our method. We add an actor as the DO as well as an MO in the environment. For the robot, we use the Jackal robot from Clearpath and an arm with the ability of lifting objects. Besides, the robot is equipped with a lidar and a camera, which are used for localization and obstacle recognition.

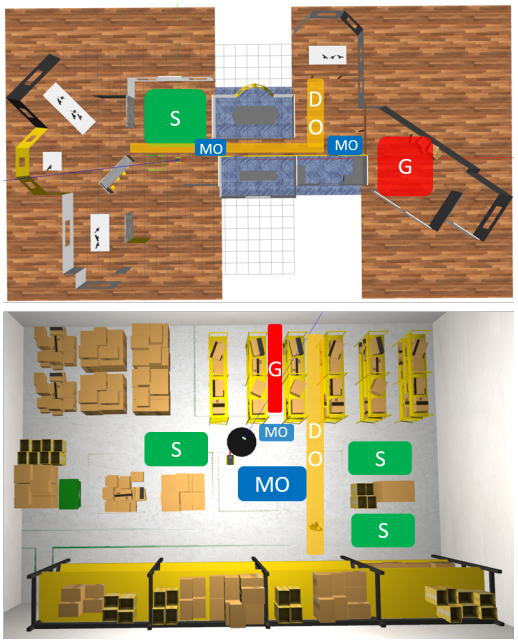

Fig. 6. Simulation environments. The image on the top is the office environment while the one on the bottom is the logistics environment. The starting point is selected from the green region while the goal is picked from the red region. The MO is set in the blue region while the DO moves in the yellow region.

All the experiments are implemented using ROS [20] and Python with the PyTorch deep learning library. We use an NVIDIA GTX1070 GPU with 8G memory and Intel i5-9600F CPU with 16G memory for all the quantitative results.

B. Backward goal prediction results

In order to compare our backward goal proposition policy, we also implement path planning algorithms to search the backward goal, including Dijkstra and RRT algorithms. Since the goal searching is an exploration task, we set a far goal for the path planning algorithms to simulate the exploration and check if the success conditions are satisfied at each path planning step.

These three algorithms are executed in an environment with random configurations for 50 times. We compare the average moving distance (AMD) and average prediction time (APT). The results can be found in Table I. From the quantitative results, we can see that Dijkstra algorithm could find the closest goal but suffers from high searching time, which results in long planning time when avoiding DO. RRT proposes both farther goals and long computation times. Such long reaction time would lead to more risks of collision with the DO. In contrast, our method finds the goal in an efficient way so the navigation could be safely executed before collision with the DO.

TABLE I

THE QUANTITATIVE COMPARISON OF DIFFERENT BACKWARD GOAL PREDICTION METHODS IN 50 TRIALS

Method	AMD(m)	APT(s)
Dijkstra	1.50	2.97
RRT	2.91	2.28
Ours	2.83	0.66

TABLE II

QUANTITATIVE RESULTS OF 21 TESTS IN SIMULATION. BOLD TEXT INDICATES BEST PERFORMANCE. SEE TEXT FOR DETAILS.

Env	Methods	SR(%)	NT(s)	CR(%)	NP(%)
Office	MoveBase	67	**55.68**	19	14
	NAMOT	81	66.27	19	0
	Ours	**100**	79.49	**0**	0
Logistics	MoveBase	71	**58.41**	29	0
	NAMOT	90	59.97	10	0
	Ours	**100**	62.57	**0**	0

C. Simulation results

We test our methods in both Gazebo environments with various task configurations. We choose random starting positions and random goals to generate multiple navigation tasks. There are mainly 6 MO and DO avoidance scenarios happening during these simulation experiments, which are shown in Fig. 7.

We compare our method with the baseline method MoveBase from ROS navigation stack [20], which uses the Timed-Elastic-Bands (TEB) planner as the motion planner to bypass all the obstacles that block the path. Besides, NAMOT [5], designed to solve NAMO in static environment, is also compared with the proposed method. Four evaluation metrics are used to analyse the performance: success rate (SR), navigation time (NT), collision rate (CR) and no path found rate (NP). SR describes the ability of achieving the task without collision with any obstacles while the CR presents the percentage of collision cases with DO. The NP happens when the MO blocks the only path to the goal.

We evaluate the performance on 21 trials (3 for each case in Fig.7 and a case without meeting DO) and the quantitative results can be found in Table II. The navigation time (NT) of MoveBase is calculated only from success cases, which explain its low value. From the results, we can find that our method could complete the navigation tasks without collision with obstacles while the MoveBase method suffers from the collision with DO. Besides, it also fails for some trials in situations like case (b) in Figure 7, where the path to the goal is blocked and there are no alternative paths. Comparing with NAMOT, the DO avoidance module proves its effectiveness for avoiding the collision with DOs.

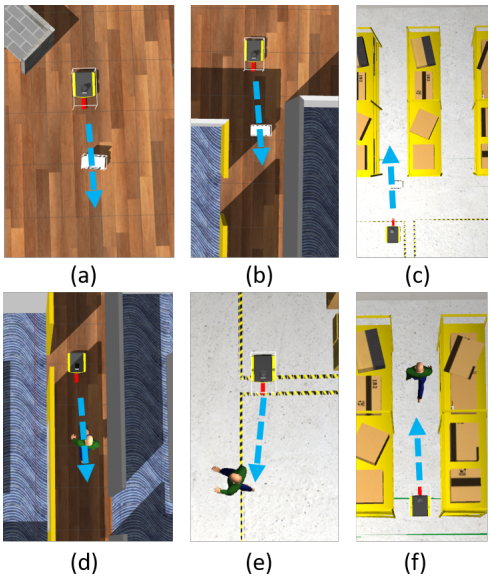

Fig. 7. Demonstration of various cases happening in the experiments. (a) The MO blocks the path and the detour is slightly longer; (b) The MO blocks the only passage to the goal; (c) The MO blocks the path and the detour is very long. (d) DO and robot move in ta corridor in opposite directions; (e) DO and robot move at random directions; (f) DO and robot move in the same direction.

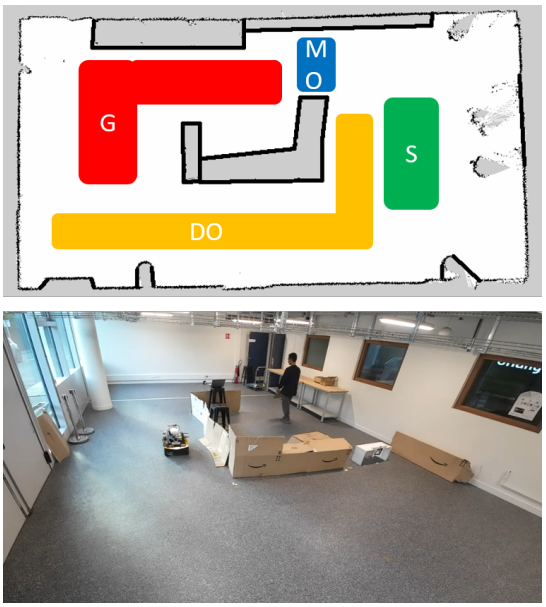

Fig. 8. Real testing environment. The robot starts from a random starting point in green region and reaches a goal selected from red region. The MO is in the blue region while the DO moves in the yellow region.

V. Real Experiments

A. Environment description

To demonstrate the effectiveness of our method in real applications, we employ the proposed method on a real Jackal robot with a realsense camera and a lidar. An arm that can move in the vertical direction to lift the MO is installed in the front part of the robot. We build a simple office environment with presence of MOs and DOs, whose layout is shown in Fig. 8. The static map of the environment is initially generated using Gmapping [15] and we add some danger zones, like tables and chairs. The robot identify the obstacles and estimate their poses with the help of Aruco markers [18].

B. Experiment results

We randomly select the goal positions in the environment and control the robot to reach them. The MO is set at different positions to simulate the cases mentioned in Fig. 7. Besides, a DO, which is a person in the experiment, walks towards different directions to test the robot's ability to avoid the DO.

We conduct 9 experiments (3 for each case (d-f) in Fig. 7) in the environment, and the quantitative results are shown in Table III. Since there are two passages in the environment, the MoveBase can always find a detour when the robot encounters the MO. The main factor to affect the success rate is the collision with the DO. In contrast to MoveBase and NAMOT, our method reaches the goals more safely with the ability to choose avoidance strategies with little sacrifice of navigation efficiency.

TABLE III

Quantitative results of 9 tests in real environment. Bold text indicates best performance. See text for details.

Methods	SR(%)	NT(s)	CR(%)
MoveBase	67	**52.81**	33
NAMOT	78	55.23	22
Ours	**100**	57.10	**0**

VI. Conclusion and Future Work

We propose a new framework to complete the NAMO tasks in a dynamic environment. When encountering a dynamic obstacle, the robot can choose to bypass it or wait for it to pass first. The decision is made by comparing the time cost of each choice. If neither option is feasible, the robot can move backward to a free place and let the DO go through the passage. This is achieved through a fast backward goal generator trained using RL. Experiment results from simulation and real environments demonstrate the effectiveness of the proposed method.

Some limitations are linked to the observation uncertainty and action uncertainty. For example, when estimating the location of the DO, localization noise could lead to wrong decisions, and when moving the MO, it can fall off the arm because it is not well picked due to the action uncertainty. We plan to solve these issues by integrating replaning strategies in future work. We also plan larger scale real experiments in more realistic environments.

Acknowledgment

This work was part of the OTPaaS project with French government funding from the "Cloud Acceleration Strategy".

REFERENCES

[1] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[2] A. Bera, T. Randhavane, R. Prinja, and D. Manocha, "Sociosense: Robot navigation amongst pedestrians with social and psychological constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 7018–7025.

[3] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1140–1146.

[4] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6057–6063.

[5] K. Zhang, E. Lucet, J. Alexandre dit Sandretto, and D. Filliat, "Navigation among movable obstacles using machine learning based total time cost optimization," 2023.

[6] J. Muguira-Iturralde, A. Curtis, Y. Du, L. P. Kaelbling, and T. Lozano-Pérez, "Visibility-aware navigation among movable obstacles," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 083–10 089.

[7] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4583–4590.

[8] K. Ellis, H. Zhang, D. Stoyanov, and D. Kanoulas, "Navigation among movable obstacles with object localization using photorealistic simulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1711–1716.

[9] M. Wang, R. Luo, A. Ö. Önol, and T. Padir, "Affordance-based mobile robot navigation among movable obstacles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2734–2740.

[10] J. Scholz, N. Jindal, M. Levihn, C. L. Isbell, and H. I. Christensen, "Navigation among movable obstacles with learned dynamic constraints," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3706–3713.

[11] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. Ieee, 2008, pp. 1928–1935.

[12] S. Liu, P. Chang, Z. Huang, N. Chakraborty, K. Hong, W. Liang, D. L. McPherson, J. Geng, and K. Driggs-Campbell, "Intention aware robot crowd navigation with attention-based interaction graph," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 015–12 021.

[13] T. Gu, G. Chen, J. Li, C. Lin, Y. Rao, J. Zhou, and J. Lu, "Stochastic trajectory prediction via motion indeterminacy diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 113–17 122.

[14] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7829–7835.

[15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[16] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8620–8626.

[17] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[18] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org