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 non-movable obstacles into account and are designed to complete a task in a static environment. However, taking into account dynamic obstacles such as coworkers in motion has a strong impact on the overall navigation strategy. We propose an hybrid framework (mixing model-based and learned components) for NAMO in dynamic environments that considers the trade off among 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 collision with dynamic obstacles in narrow passages using a goal generator based on reinforcement learning. The proposed method is evaluated in two simulated home and warehouse scenarios and in a small scale real environment, showing its efficiency and safety to navigate in a dynamic environment.**

I. Introduction

Navigation among movable obstacles (NAMO) [1] typically assumes a static environment with the robot being the sole dynamic entity. Nonetheless, various scenarios may involve dynamic obstacles (DOs) such as humans or other robots within the workspace. Neglecting the DOs in the NAMO problem can lead to potential danger for both robots and humans, and deadlock in some situations where the robot is impeded from reaching its goal or maneuvering obstacles as intended.

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 (meaning when planning path and deciding where to move obstacles) to guarantee safety and goal reachability.

The second problem we tackle 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 general 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 path. There exist many approaches for navigation in dynamic environment (e.g., [3], [4] in the recent works), but a prevalent constraint is that the robot can only make a detour or wait for the human to free the path. These strategies will be effective in situations where there is ample space around the DO and its associated danger area. But in narrow passages, like home corridors or warehouse aisles, the existing methods are only applicable in situations (i) and (iii). In situation (ii), waiting in a narrow passage until it is cleared risks a deadlock where the DO can not exit and the robot is unable to proceed. A wise choice is that the robot plans and performs a backward motion to free the DO's path before waiting for the DO to pass.

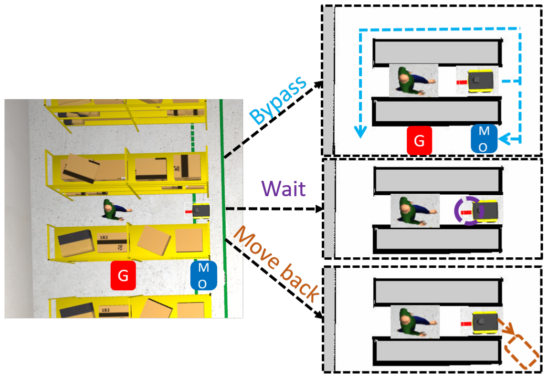

Fig. 1. Different reactions to dynamic obstacles. Overall, three strategies can be applied by the robot: bypass, stop and wait, move backward and wait. Red region G is the goal, dark blue region MO is the movable obstacle.

To keep safety and navigate efficiently, 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's path while the robot is still able to observe whether the human has left the aisle.

In this paper, we extend our previous work on NAMO [5] from static environments to dynamic environments. We integrate a trajectory prediction module to estimate the DO's intention in order to anticipate the collision and choose appropriate avoidance strategy between bypassing, waiting and a new moving backward behavior. To find a suitable backward place, we propose a backward goal generator based on reinforcement learning (RL). By navigating to the backward goal, the robot frees the DO's path and can proceed its navigation task when DO leaves. Besides, the predicted trajectory of DO is integrated in stock region prediction [5] for movable obstacles (to avoid putting obstacles in DO's path). We tested this solution in both simulation and real environments in order to show its efficiency and capacity to avoid deadlock situations and reach the goal safely.

In the following section II, we position our work with respect to the related work before describing the approach

Website: kai-zhang-er.github.io/namo-dynamic
[1] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
[2] U2IS, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
kai.zhang@cea.fr

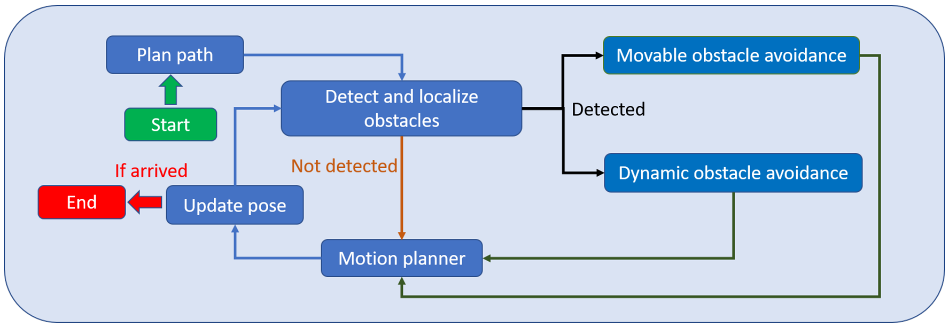
Plan path
Start
Detect and localize obstacles
Movable obstacle avoidance
Detected
Dynamic obstacle avoidance
If arrived
Not detected
End
Update pose
Motion planner

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

in section III, and presenting results in simulation (section IV) and on a 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*

Benefiting from their interaction ability, robots can navigate to the goal faster by making some blocked navigation path passable [6], [7]. There are mainly two types of interaction when the robot meets a movable obstacle (MO), either bypass or remove it. Most NAMO algorithms rely on predefined strategies, which means to remove all MOs that block the path [8] or bypass them whenever possible. However, some methods provide the choices for the robot based on either the estimation of the obstacle movability [9], [10] or the estimated time cost to move the obstacle [5]. If the robot cannot push the MO away or if the removal action will be time-consuming, 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 system 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 classic methods to avoid collision with DOs firstly model the motion for the DOs and then control the robot to avoid their future occupied 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 intruding the DO's path while in [15], RL is used to learn to map observations to linear and angular speeds directly for navigation in a crowd environment.

Most of the current methods are applied in environments with sparse obstacles so the robot could bypass the DOs, especially the humans. However, in narrow spaces, the robot either bypasses the humans without respecting the safety distance or waits for the DO to clear the path, potentially leading to deadlock situations and making coworking impractical. In contrast, we give high priority to DOs and optimize robot's trajectories to actively avoid disturbing the DOs as much as possible for harmonious side-by-side operation. Besides, as illustrated in table I, the proposed method is the first one to take into account the three types of obstacle (static, movable and dynamic) in the navigation task, thus showing potential to be applied in general environments.

TABLE I
Obstacle types taken into account in collision avoidance algorithms from the state of the art.

	Static	Movable	Dynamic
ORCA [16]	×		×
NAMOP [8]	×	×	
CrowdNav [12]			×
DRL-VO [15]	×		×
NAMOT [5]	×	×	
Ours	×	×	×

III. Method

We will first describe the task and the system structure, before presenting the DOs avoidance strategy in details.

A. *Task description and method overview*

The task is to navigate from a starting position to a goal with the presence of static, movable or dynamic obstacles. The robot has to avoid collision with obstacles during navigation 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 using a LiDAR (e.g., [17]), and that the obstacle types are easily identified using a camera.

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 unexpected obstacles. If an obstacle is detected, the avoidance algorithm is applied according to the obstacle type, 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.

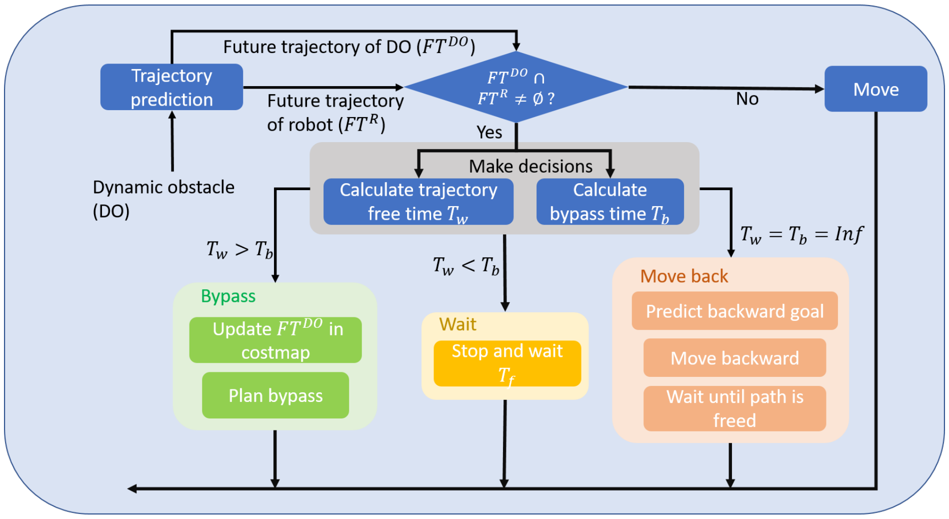

Fig. 3. Pipeline of the dynamic obstacle avoidance. Three reactions can be made to avoid the collision: bypass obstacle, wait for the obstacle to pass and move back to let the obstacle pass. Our approach evaluate goal reaching time for each option and choose the fastest one.

For movable obstacles, the avoidance strategy chooses between bypass and removal, which means the robot could move the obstacle to clear the path for navigation. The choice between these two options is made by comparing the time cost of each option, 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 the robot. If there is intersection between these two trajectories, in other words, a collision may happen later, we first estimate the time necessary for the detour and for waiting for the passage of the DOs. If one of these options is infeasible, the time cost is set as infinite and the option with smaller cost is adopted. If both options are infeasible, resulting in infinite time cost, we plan a backward motion of the robot to free 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 and DO avoidance are handled independently. After moving backward, if the robot detects a MO, it will decide whether to remove or bypass it based on the time cost.

All components are detailed 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 static obstacles, such as walls, that are 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 and inspired by [18], we modify the dynamic window approach (DWA) [19] 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 the objective function (1) with the assumption that the DO avoids obstacles and follows approximately constant motion so its linear and angular velocities do not change quickly:

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

where (v,w) are linear and angular velocities of the DO, C_{clear} is the inverse value of the time until collision 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 [20] 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. The DO's trajectory is predicted over the next $T_{pred} = 8$ seconds. To reduce the impact of the constant velocity assumption in DWA, the prediction refreshes at 30 Hz to make sure the estimated trajectory is recent and reliable.

C. Decision making

After obtaining the future trajectory of the DO, noted FT^{DO}, and of the robot (using the planned trajectory), noted 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 collision-free shortest path to the goal. 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 for 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 record the time when $min(dist(FT^{DO}, FT^R)) > TH_{dist}$. Fig. 4 illustrates a case where the distance between robot and DO is safe for navigation at $T_w = 2$. 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 $T_w = Inf$.

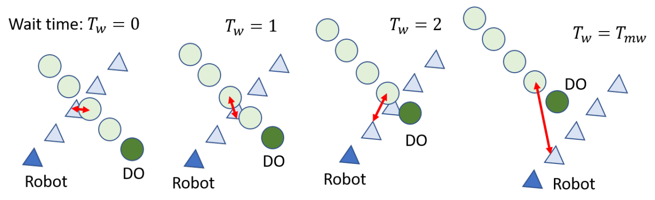

Fig. 4. Examples of calculating the wait time. The triangles and the circles are the future trajectories of the robot and the DO respectively. The red arrows show minimum distance between these two trajectories.

When both bypass and wait options are not applicable, that is, when $T_b = T_w = Inf$, which usually occurs in a narrow passage where the DO comes in the opposite direction of the robot path, the robot is ordered to move backward and clear 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 detects a DO in a narrow passage moving in the opposite direction, the bypass and wait options are infeasible. 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 takes minimum time to reach from the robot current position. (b) The passage is visible from the backward goal so that when the DO leaves the passage, the robot observes it and continues the navigation task. (c) The goal should be outside of the DO path so that the robot won't obstruct the DO.

We propose an exploration method based on RL to learn a policy able to quickly find a backward goal reachable from current robot position respecting these constraints. The policy illustrated in Fig. 5 uses three binary images as input:

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 current position (M_{visi}). The map is generated by using LiDAR data to find the visible regions around the robot. The white (resp. black) region is the visible (resp. invisible) region from robot current position. Moreover, it is also used to ensure that the DO's path is still visible so the robot can detect it when DO exits the passage.

The outputs are linear and angular accelerations, a_v and a_w, chosen because they can generate smooth trajectories feasible for the real robot. We set a time interval $D_t = 0.5s$ for each step after which the robot moves with a_v, a_w and reaches a new pose. 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 DO's path, the episode stops and the current position of the robot is considered as the backward goal. At each step, we compute a reward R_i:

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

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

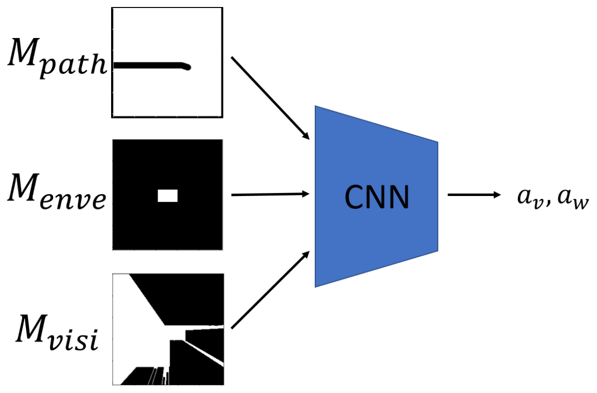

Fig. 5. Structure of backward goal generator. The inputs include three binary images: path map M_{path}, envelope map M_{enve} and visibility map M_{visi}. PPO [21] is used for training and the actions include linear and angular accelerations a_v, a_w.

We use PPO [21] 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 simulated 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 10 hours. During testing, in simulation or on real robot, the pretrained policy is applied to predict the backward goal, that is sent to the path and 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 [22] environments. As shown in Fig. 6, an office environment and a warehouse environment are built to analyse the scalability of our method. We add an actor as the DO as well as a 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.

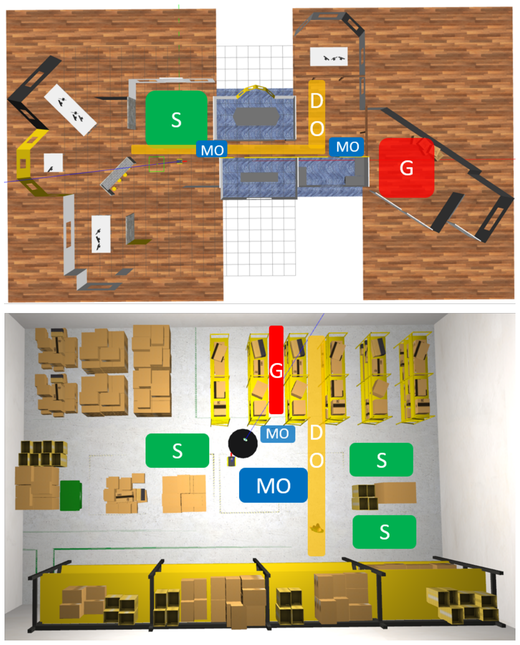

Fig. 6. Simulation environments. The image on the top is the office environment while the one on the bottom is the warehouse environment. The starting point is selected from the green region while the goal is picked from the red region. The MO is set in the blue region while the DO moves in the yellow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

All the experiments are implemented using ROS [23] 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 evaluate our backward goal proposition policy, we also implement exploration algorithms to search the backward place, including Dijkstra [24] and RRT [25]. Since the goal searching is an exploration task[1], we set a random far away goal to stimulate the exploration and check if the success conditions are satisfied at each explored position. These three algorithms search the backward place in a 2D map with resolution of 0.02 m/pixel. At each step, the search region is limited to a circle region around the robot with a radius of $r_{visi} = 4$ meters, which equals to the size of local visible map.

TABLE II
Quantitative comparison of different backward goal prediction methods on 50 trials

Method	Step (m)	AMD (m)	APT($\mu \pm \sigma$) (s)	MPT (s)
Dijkstra [24]	0.20	2.37	1.52 ± 1.75	8.17
	0.40	2.36	1.23 ± 1.16	5.63
	0.60	**2.34**	1.16 ± 1.02	4.87
RRT [25]	0.20	3.25	1.31 ± 0.84	3.54
	0.40	3.22	1.11 ± 0.80	3.21
	0.60	3.25	1.09 ± 0.85	3.25
Ours	0.25	2.70	**0.79±0.36**	**1.49**

For the Dijkstra and RRT algorithms, the search region is generated from the waypoints of the DO's future path with radius of r_{visi} to guarantee that the path is always visible and the distance between two neighboring waypoints is fixed. As for the proposed RL method, the moving direction and distance is calculated from the outputs. These three methods are evaluated in an office environment with random configurations for 50 trials.

We compare in Table II the average moving distance (AMD) to the backward goal, average prediction time (APT) to find a suitable goal and the maximum prediction time (MPT) showing the performance at the worst case. We set different waypoints intervals (column Step in Table II) for Dijkstra and RRT algorithms to analyze its impact on the performance. The reported step length of the proposed method is the mean value of the 50 tests. From the quantitative results, we can see that Dijkstra algorithm could find the closest backward place but suffers from longer searching time. Particularly in the worst case, it takes much longer planning time, which could lead to long waiting time for the DO waiting for the robot to move. In contrast, our method finds the goal in an efficient way with little sacrifice on the moving distance. Even in the worst case, the prediction time is much less than the future collision checking time $T_{pred} = 8s$. Therefore, the backward action could be safely executed before colliding with the DOs.

C. Simulation results

For experimental results, we use two Gazebo environments with random starting positions and goals to simulate various 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 ORCA [16] and Timed-Elastic-Bands (TEB) [26], two motion planners designed

[1]Note that this prevents us to use informed algorithms such as A*.

to bypass all the obstacles that block the path, and with NAMOT [5], designed to solve NAMO in static environment. 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 III. The navigation time (NT) of ORCA and TEB is calculated only from success cases, which explains its low value. Results show that our solution could complete the navigation tasks without collision with obstacles, while the ORCA and TEB methods incur collisions with DOs. Besides, it also fails to find its way for some trials in situations like cases (b) and (d) in Figure 7, where the sole path to the goal is blocked or there is no ample space for detour. Comparing with NAMOT, the DO avoidance module proves its effectiveness to avoid the collision with DOs in a dynamic environment.

Additionally, to prove the real-time capability, we show in Table IV the running time of trajectory prediction, the planning time when the robot avoids a MO (NAMO) and a DO (NADO). The predicted trajectory can be updated nearly 38 times per second, which means that the robot can estimate the latest intention of the DOs and choose proper reaction. The planning time of NAMO and NADO is less than 1 second, demonstrating that our planning modules have small influence on the real-time navigation performance.

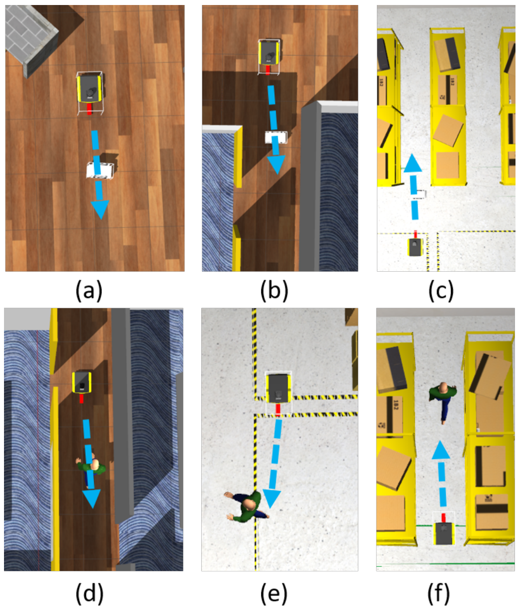

Fig. 7. Demonstration of various cases happening in the experiments. (a) The MO blocks the path and the detour is slightly longer; (b) The MO blocks the only passage to the goal; (c) The MO blocks the path and the detour is very long. (d) DO and robot move in a corridor towards opposite directions; (e)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

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

Env	Methods	SR(%)	NT(s)	CR(%)	NP(%)
Office	ORCA [16]	71	**51.60**	15	14
	TEB [26]	67	55.68	19	14
	NAMOT [5]	81	66.27	19	0
	Ours	**100**	79.49	**0**	0
Warehouse	ORCA [16]	76	**54.49**	24	0
	TEB [26]	71	58.41	29	0
	NAMOT [5]	90	59.97	10	0
	Ours	**100**	62.57	**0**	0

TABLE IV
AVERAGE RUNNING TIME OF DIFFERENT PLANNING MODULES IN 20 TESTS

	Trajectory prediction (s)	NAMO (s)	NADO (s)
Time	0.026	0.108	0.964

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 generated using Gmapping [17]. The robot identifies the obstacles and estimates their poses with the help of Aruco markers [20].

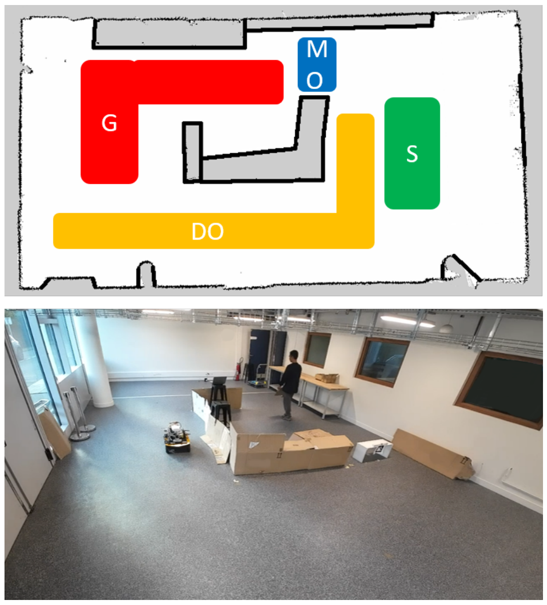

Fig. 8. Real testing environment. The robot starts from a random starting point in green region and reaches a goal selected from red region. The MO is in the blue region while the DO moves in the yellow region.

TABLE V
QUANTITATIVE RESULTS OF 9 TESTS IN REAL ENVIRONMENT. BOLD TEXT INDICATES BEST PERFORMANCE. SEE TEXT FOR DETAILS.

Methods	SR(%)	NT(s)	CR(%)
TEB [26]	67	**52.81**	33
NAMOT [5]	78	55.23	22
Ours	**100**	57.10	**0**

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 V. Since there are two passages in the environment, the TEB 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 TEB and NAMOT, our method reaches the goals more safely with the ability to choose avoidance strategies with little sacrifice of navigation efficiency.

VI. CONCLUSION AND FUTURE WORK

We propose a new framework to complete the NAMO tasks in a dynamic environment. When encountering a DO, 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, which usually happens in a narrow passage, the robot can move backward to a free place and allow the DO to go through. 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 may fall off the arm due to incorrect loading pose caused by the action uncertainty. We plan to solve these issues by integrating replaning strategy in future work. We also plan larger scale experiments in more realistic environments with more DOs and MOs.

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] Z. Xie and P. Dames, "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles," *IEEE Transactions on Robotics*, 2023.

[16] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 3–19.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ IROS*, vol. 3. IEEE, pp. 2149–2154.

[23] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system.” [Online]. Available: https://www.ros.org
[24] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[25] S. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” *Research Report 9811*, 1998.
[26] C. Rösmann, F. Hoffmann, and T. Bertram, “Kinodynamic trajectory optimization and control for car-like robots,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5681–5686.