2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Unsupervised Multiple Proactive Behavior Learning of Mobile Robots for Smooth and Safe Navigation

Arthicha Srisuchinnawong, Jonas Bæch, Marek Piotr Hyzy, Tsampikos Kounalakis, Evangelos Boukas, and Poramate Manoonpong, *Senior Member, IEEE*

***Abstract*— While different control approaches have been developed for smooth and safe navigation, they are limited by the needs for model-based assumptions, true training target/reward function, and/or large sample data. To overcome these limitations, this study proposes a model-free neural control architecture with a generic plug-and-play online Multiple Proactive Behavior Learning (MPL) module. The MPL adapts robot neural control policy in an online unsupervised manner with small sample data by correlating its sensory inputs to a local planner command. As a result, it allows a mobile robot to autonomously and quickly learn and balance various proactive behaviors related to smooth motion and collision avoidance. It also compensates for the limited planning update rates and the planning model mismatch of an arbitrary local motion planner. Compared with existing control approaches without the MPL, our control architecture with the MPL leads to (1) a 10% improvement in the smoothness of robot motion and 30% fewer collisions in a narrow static environment, and (2) trading motion smoothness for up to 70% fewer collisions in an unknown dynamic environment. Taken together, this study also demonstrates how to apply model-free neural control with unsupervised learning to existing model-based control (e.g., local motion planner) for efficient proactive behavior learning and control of mobile robots.**

I. Introduction

Autonomous mobile robots are used in different service applications [1, 2], which require smooth and safe autonomous navigation. Typically, the robots employ local motion planners to plan the motion/commands to safely react to obstacles and smoothly navigate toward sub-goals generated from global path planners [3].

Most local motion planners begin with identifying several input candidates using, for example, predefined motion primitive (Kinematic Tree (KT)) [2], consistent accelerations (Dynamic Window Approach (DWA)) [4], and random sampling (Rapid Exploring Random Tree (RRT)) [5]. Given those input candidates, future trajectories are simulated using a kinematics model. Each trajectory is then evaluated with a cost function. Finally, the corresponding candidate with the minimum cost is selected for the execution. Extending from this concept, some methods, such as Model Predictive Control (MPC) [6], employ an optimization algorithm, such as SQPsolver, to optimize the entire set of bounded commands and increase the performance.

A. S. and P. M. are with the Embodied Artificial Intelligence and Neurorobotics Laboratory, the University of Southern Denmark, Odense, Denmark, and the School of Information Science and Technology, Vidyasirimedhi Institute of Science and Technology, Rayong, Thailand.

J. B. and T. K. are with Danish Technological Institute, Odense, Denmark.

M. H. and E. B. are with the Technical University of Denmark, Lyngby, Denmark.

P. M. is the corresponding author (poma@mmmi.sdu.dk).

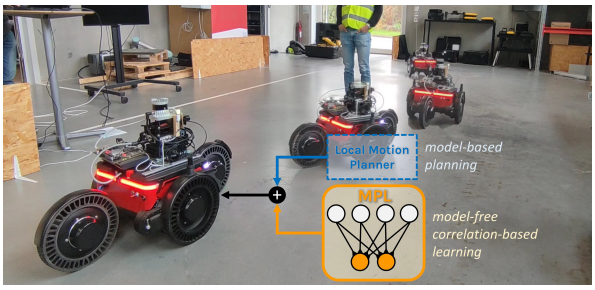


Fig. 1: Smooth and safe navigation under an integrated model-based motion planning and model-free neural control with unsupervised correlation-based learning. The video is available at https://youtu.be/yF7n7kNFumo.

However, in a complex (dynamic) environment, e.g., with moving obstacles, local planners require additional information for the anticipation, especially when their planning rates are limited to a low update frequency (1–10 Hz) due to computational limitations [5–8]. Examples of the additional information are predicted trajectories [6–8] or obstacle models [9], which are used to penalize possible collided movements. However, such predicted trajectory and obstacle information-based techniques require presumptions of operation modes [7], cooperative obstacles [7, 8], or homogeneity and omniscience [6, 9]. Consequently, they are unsuitable for complex environments that violate such assumptions. To address this, roboticists have shifted their focus toward learning-based methods [3, 10].

Learning-based methods divide into three types: reinforcement (RL), supervised (SL), and unsupervised (UL) learning. Firstly, RL (such as AutoRL end-to-end training [10] and crowd interaction-aware network [11]) allows robots to anticipate the environment by learning from many training samples of trail and error (200–4k trials) [10, 11]. Secondly, SL (such as learning from human demonstration(s) [12]) might require less training; nevertheless, true training targets or supervised data are needed. Thirdly, UL does not require many training samples or known true target/reward function. Thus, it is a potential candidate for practical use.

A possible UL technique applied to this mobile robot control application is correlation-based learning [13–15], which is inspired by biological Hebbian learning. Using

this mechanism, one group focused solely on learning safe obstacle negotiation behaviors. [16] achieved a drone's proactive speed adaptation after 3–5 trials by correlating long distance-to-obstacle-based proactive feedback to short distance-to-obstacle-based reflex feedback. [17] achieved a hexapod's proactive obstacle negotiation after 10 trials by correlating the time-delay signals of the following legs to the torque-based reflex signals of the front legs. Another group, on the other hand, focused solely on learning smooth behaviors. [18] applied this UL technique to visual and audio inputs to obtain proactive turning speed adaptation with less oscillation, driven by a proactive (earlier) audio cue. [19] applied the technique to distance feedback and auditory inputs to obtain smooth acoustic navigation. Others also applied the technique to sensory inputs for unsupervised preprocessing [20] or learning to navigate toward goals [21, 22]. Nevertheless, since those correlation-based approaches employed nonadaptable proactive information (e.g., direct sensory feedback or handcraft signals, as summarized in Table I), their resulting proactive behaviors are uni-purpose, improving either smoothness or collision avoidance as predesigned [16–20], but not both under a single control.

TABLE I: Different correlation-based UL methods, where red text highlights nonadaptable proactive information.

Method	sensory feedback	proactive information
[16]	distance feedback	preprocessed distance
[17]	delayed signal	preprocessed torque
[18]	audio input	visual input
[19]	obstacle feedback	auditory input
[20]	sensory inputs	sensory inputs
[21]	sound detector	light detectors
[22]	sensor fields	closest sensor field
MPL	**obstacle feedback**	**local planner command** (adaptable proactive information)

To this end, this study proposes a control architecture that autonomously improves and balances both smoothness and collision avoidance in a mobile robot. The architecture includes a plug-and-play Multiple Proactive Behavior Learning (MPL) module added to an arbitrary local planner, as shown in Fig. 2. The MPL module aims to autonomously learn multiple proactive robot behaviors, including speed reduction and proactive steering, by correlating its sensory input (obstacle feedback) to the local planner command. This work also verifies that using the planner command as an adaptable proactive information can equip the mobile robot with different but compatible proactive behaviors. Taken together, the main contributions of this study include:

1) an adaptive control strategy of using a local planner command as an adaptable proactive control signal for developing multiple proactive behaviors.
2) the MPL plug-and-play module that quickly learns to anticipate the environment and act proactively without assumption and additional target/reward function.

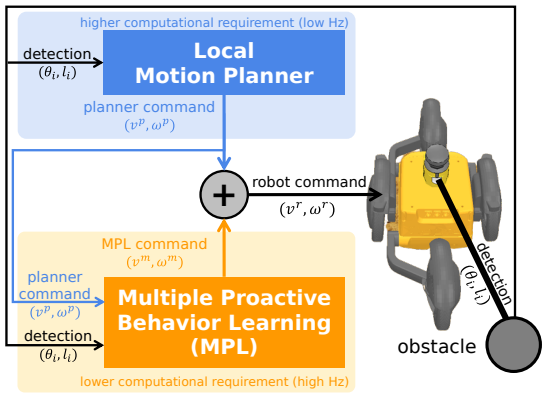


Fig. 2: Diagram of standard local motion planner in blue, with the proposed MPL in orange.

II. Capra Hircus Robot

A Capra Hircus Robot [1], illustrated in Fig. 3, is employed to verify the MPL. The robot is 60 cm wide, 110 cm long, and 37 cm high. It consists of four wheels: two active wheels on the side and two passive wheels on the front and back. The yaw of each passive wheel is actively controlled, while the pitch is passively adjusted by a compliance joint.

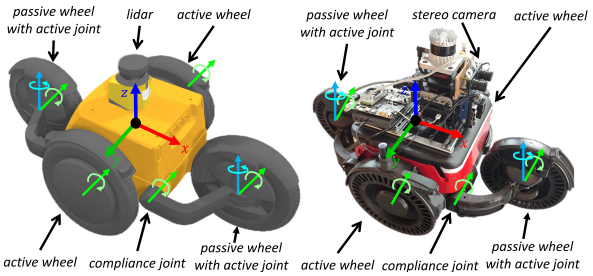


Fig. 3: Simulated and physical Capra Hircus Robot [1] presented along with the indicated components, frames of reference, and rotation axes.

The robot receives a steering command, consisting of linear and angular velocity commands (v^r and ω^r), which are converted to wheel speed commands by an onboard controller to control the robot. Additionally, a lidar sensor is installed at the center of the robot. To make the task more challenging, the maximum detection range is set to be 2 m (merely 6–10% of the others [6, 8, 12]). After preprocessing, the sensory feedback signals are the detection angles ($\theta_i[t] \in [0, 2\pi]$) and detection signals ($l_i[t] \in [0, 1]$), where $l_i[t] = 0$ indicates no obstacle detected and $l_i[t] = 1$ indicates an obstacle locating at the robot center.

III. Multiple Proactive Behavior Learning (MPL)

In order to improve the performance of the model-based local planner and develop the various proactive behaviors, the MPL is proposed and added as illustrated in Fig. 2. The MPL is modeled as a simple neural control (Section III-A), mapping from the sensory feedback (l_i and θ_i) to the

MPL steering command (v^m and ω^m), as shown in Fig. 4. The mapping is learned through an online neural learning mechanism [14, 15] (Section III-B) based on the embodied robot-environment interaction and the correlation between two types of input information: the sensory feedback (l_i) and the adaptable local planner steering command (v^p and ω^p, correlated adaptable proactive information). Because of this simple mechanism, the mapping parameters are quickly learned online without any true target/reward function and large sample data. Furthermore, the outputs can be updated at a high frequency. Finally, the steering command from the local planner and that from the MPL are summed together as the robot command (v^r and ω^r), which is used to control the robot, allowing the MPL to learn various proactive strategies and effectively cooperate with the local planner. In principle, such proactive behaviors are derived from the abilities of the MPL to i) learn to encode how the local planner command associates with the sensory information in an online unsupervised manner and ii) allow for a higher update frequency than the local planner itself (outdating control command)[1].

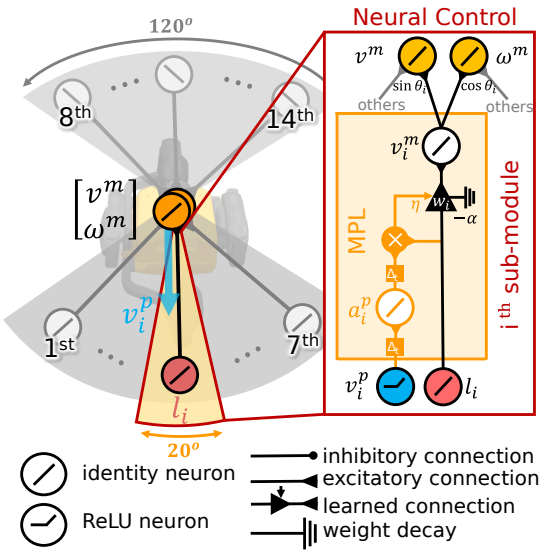


Fig. 4: MPL neural control network, consisting of 14 sub-modules, which share the same structure. Note that $v_i^p = f(v^p, \omega^p)$ denotes a component of the local planner command along/corresponding to the direction of the i$^{\text{th}}$ sub-module, computed as shown in Eq. 4.

A. Neural Control

To facilitate the learning speed, the neural control is modeled as a simple discrete-time feedforward neural network, divided into 14 sub-modules ($N = 14$), covering 120° in the front and back, as shown in Fig. 4. Each sub-module detects the closest obstacle within its coverage (20^o), as illustrated in Fig. 4. These values are selected such that the robot successfully operates in the conditions presented in Section IV at a high update frequency (30 Hz). At each timestep (t), the outputs ($v^m[t]$ and $\omega^m[t]$) are computed from the detection signals ($l_i[t]$) and the detection angle ($\theta_i[t]$) of the closest obstacle within the coverage, as:

$$\begin{bmatrix} v^m[t] \\ \omega^m[t] \end{bmatrix} = -\sum_{i=1}^{N} \left(w_i[t]\, l_i[t] \begin{bmatrix} \cos\theta_i[t] \\ \sin\theta_i[t] \end{bmatrix} \right), \tag{1}$$

where i denotes the index of sub-module and the corresponding parameters ($N = 14$, $i \in \{1, ..., N\}$). $w_i[t]$ denotes the mapping weight of the i sub-module at timestep t, learned according to Section III-B. Note that, for each sub-module (i.e., different θ_i), $\cos\theta_i[t]$ and $\sin\theta_i[t]$ are used to extract the frontal component (i.e., speed reduction) and the sideways component (i.e., left/right turning) of the counteract command, respectively.

B. Online Neural Learning Mechanism

To learn the proper set of the neural control mapping weights, the unsupervised learning rule is formulated as:

$$w_i[t+1] = \text{ReLU}\left(\alpha w_i[t] + \eta \Delta w_i[t]\right), \tag{2}$$

where $w_i[t]$ denotes the i^{th} output mapping weight at timestep t. $\Delta w_i[t]$ denotes the weight update computed from Eq. 3. ReLU() denotes the rectified linear function, used to prevent negative weights (i.e., moving toward obstacles). α denotes a decay rate, selected as 0.999 to avoid being unable to reach goals due to too strong proactive obstacle avoidance (i.e., too high weights). η denotes a learning rate, empirically set to a low value of 0.01 to stabilize learning and achieve the behavior shown in Section IV.

Based on unsupervised differential Hebbian learning in Eq 3 [14, 15], the weight update is formulated by the correlation between the sensory feedback ($u[t] \rightarrow l_i'[t]$) and the discrete temporal differentiation of the breaking signal (i.e., the corresponding negative acceleration signals, $v[t] \rightarrow a_i^p[t]$), as shown in Eq. 3.

$$\begin{aligned} \Delta w_i[t] &= u[t] \frac{dv[t]}{dt}, \\ &= l_i[t]\, \Delta\left(-a_i^p[t]\right), \\ &= -l_i[t]\, \Delta a_i^p[t], \end{aligned} \tag{3}$$

where $\Delta w_i[t]$ denotes the weight update at timestep t. $u[t]$ denotes a proactive signal at timestep t. $v[t]$ denotes a reflexive signal at timestep t. $l_i[t]$ denotes the detection signal timestep t. $a_i^p[t]$ denotes the local planner acceleration along/corresponding to the i^{th} sub-modules at timestep t, which is computed from Eq. 4.

$$\begin{aligned} a_i^p[t] &= \text{ReLU}(\Delta v_i^p[t]), \\ &= \text{ReLU}(\Delta |v^p[t]| \cos\left(\phi[t] + \theta_i[t]\right)), \end{aligned} \tag{4}$$

where $v_i^p[t]$ denotes the local planner velocity along/corresponding to the i^{th} sub-modules at timestep

[1]Note that only the ability to update at a higher frequency (e.g., SL behavior cloning with neural network) without the proactive behavior learning yields the similar output as the planner, as shown at https://github.com/Arthicha/Multiple-Proactive-Behavior-Learning, as such the robot performance cannot be improved.

t. ReLU() denotes the rectified linear function. $v^p[t]$ and $\omega^p[t]$ denote the local planner twist command at timestep t. $\phi[t]$ denotes the estimated direction of the command, which is computed from $\phi[t] = \text{atan2}(\omega^p[t], v^p[t])$. $\theta_i[t]$ denotes the detection angle corresponding to the i^{th} sub-modules at timestep t.

Intuitively, according to Eq. 3, a weight is increased ($\Delta w_i[t] > 0$) when the robot detects an obstacle ($l_i[t] > 0$), and the local planner reduces the speed or steers away ($a_i^p[t] < 0$). This learning strengthens proactive obstacle avoidance. Conversely, a weight is reduced ($\Delta w_i[t] < 0$) when the robot detects an obstacle ($l_i[t] > 0$), and the local planner accelerates or steers toward the obstacle ($\Delta w_i[t] > 0$). The weights then converge when there is a homeostasis or an oscillation between these two conditions, implying a match/correlation between the detection signals and the adaptable local planner command. Therefore, the interaction with the unknown environment is the crucial factor controlling the weight update in this learning rule [15].

According to the neural control and learning described above, when an obstacle is detected, the corresponding sub-module contributes the counteract command to reduce the speed and steer the robot away from the obstacle, as illustrated in Fig. 5. The sensitivity and strength of these are determined by the mapping weights $w_i[t]$, learned online using UL. Finally, the final robot command ($v^r[t]$ and $\omega^r[t]$) are the summation of the local planner command ($v^p[t]$ and $\omega^p[t]$) and the MPL command ($v^n[t]$ and $\omega^n[t]$), clipped between a safety limit ($v^r[t] \in [-0.5, 0.5]$ and $\omega^r[t] \in [\pi, \pi]$).

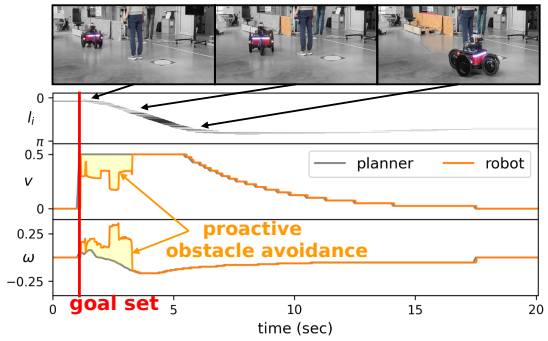
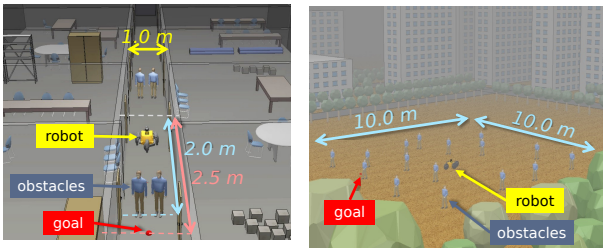


Fig. 5: Snapshots of the physical robot avoiding a human with the local planner (DWA)+MPL, the corresponding obstacle detection signals (l_i) between 0 and π rads, where the intensity represents the magnitude of the signals, and the command generated with the MPL (orange) and without the MPL (gray). Note that the MPL generates the counteract command highlighted in yellow to proact to obstacles[1]. The video is available at https://youtu.be/yF7n7kNFumo.

IV. Experiments and Results

A. Experimental Setup

1) Testing Environments: To evaluate and present the proposed MPL, two challenging simulation environments were selected for comparison. In the first testing environment, shown in Fig. 6a, the robot was in a narrow corridor measuring 1.0 m in width. Its mission was to move toward the goals, switching between 2.5 m and -2.5 m along the corridor after the robot achieved the acceptance radius of 0.7 m. To make the task more challenging, at 2.0 and -2.0 m, four standing people prevented the robot from reaching the goals. Therefore, the robot had to learn to proactively reduce its speed while avoiding collision with the people and walls. In the second testing environment, shown in Fig. 6b, the robot was in a 10.0 × 10.0 m^2 two-dimensional space among 16 walking people acting as moving obstacles. Those people followed different predefined trajectories/paths at different random speeds between 0.2–0.3 m/s (50±10% of the robot's maximum speed and reset at the end of the paths); they cannot see or avoid the robot. Thus, even if having the non-holonomic constraint, the robot had to move toward goals using the same acceptance radius while avoiding collisions in a nearly unpredictable environment, where the obstacles can be present in different locations.



(a) (b)

Fig. 6: Setup of the (a) static narrow corridor and (b) free space with dynamic obstacles.

2) Testing Conditions: Since this study focuses mainly on local motion planning, environment mapping and global path planning were omitted. As a result, the local goals were random to be at different locations in the testing space, and the robot merely observed the environment and detected the obstacles within a relatively small area of size 2-m radius (6–10% of the lidar range used in other works [6, 8, 12]).

Four different state-of-the-art local motion planning techniques: the DWA, MPC, DWA with obstacle prediction, and MPC with obstacle prediction, were employed for comparison under three desired planning rates (ROS rates): 5 Hz, 10 Hz, and 20 Hz. The DWA [4], which is provided by ROS navigation package [23], was selected as the representative of standard local planner techniques [2–5], and the MPC [6] as that of the state-of-the-art optimization-based planning techniques [3, 6]. They were employed to demonstrate the compatibility of the proposed MPL to various local motion planning approaches, regardless of their processes. Besides, a simplified obstacle prediction was also included as the representative of obstacle prediction methods, which extend from this concept by performing uncertainty estimation and filtering [5, 6], including social interaction

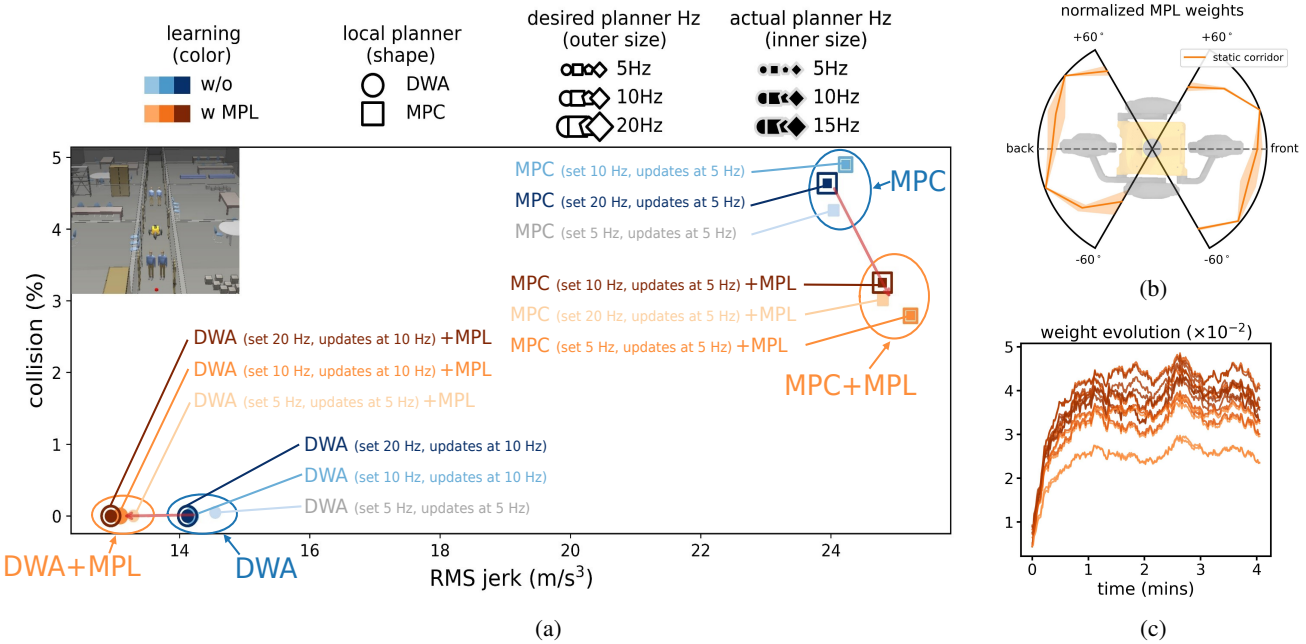


Fig. 7: (a) Collision rate (percentage) and RMS jerk obtained from different conditions in the static corridor testing environment, where the conditions with the MPL are indicated by orange-ist colors and the ones without by blue-ish colors, the types of local motion planners by the marker types, the desired planning rates by the size of the outer edge, and the actual planning rates by the size of the inner marker. The red arrow pointing from the upper left to the lower right indicates that the MPL reduces collision while increasing motion jerk, and that pointing from the upper right to the lower left indicates that the MPL reduces both collision and motion jerk. (b) Average normalized weights obtained when the MPL converged, and (c) evolution of the weights in the first four mins. The video of the experiment is available at https://youtu.be/yF7n7kNFumo. Note that, in this testing environment, the conditions with obstacle prediction (DWA with obstacle prediction and MPC with obstacle prediction) are omitted due to the non-moving obstacles.

[7, 8], or using neural networks [11]. These prediction-based approaches are employed to further demonstrate that the MPL can further improve other proactive navigation techniques. For each local motion planner type, the comparison was performed between using the local planner solely and using the local planner along with the proposed MPL, where the hyper-parameters/gains were the same throughout the entire study.

As the MPL does not require any additional costs/objective functions, all conditions employ the same local planner cost function, which is:

$$C = D(p_r[T], p_g) + \lambda \exp\left(-\frac{\min \mathrm{D}\left(p_r[\tau], p_o[\tau]\right)^2}{\sigma^2}\right), \quad (5)$$

where C denotes the cost. $D(p_r[T], p_g)$ denotes the two-dimensional Euclidean distance between the robot position at the prediction horizon ($p_r[T]$) and the goal position (p_g). $p_r[\tau]$ denotes the robot position at timestep τ. $p_o[\tau]$ denotes the obstacle position at timestep τ. T denotes the horizon, selected as 2.0 s to the future to balance the computational speed and performance. λ denotes the obstacle avoidance gain, selected as 10 to ensure safety. σ denotes another gain, selected as 0.5 such that the ideal safety distance from a static obstacle in free space is $\approx$ 1 m (i.e., two times of the robot radius). Additionally, in case of obstacle prediction: $p_o[\tau + \Delta\tau] = p_o[\tau] + (p_o[\tau] - p_o[\tau - 1])\Delta\tau$.

3) Comparison Matrices: Three matrices, including, the average local planner update rate, Root Mean Square (RMS) jerk, and average collision percentage, are selected for comparison, where a statistical significant t-test was employed to highlight the significant comparison and account for statistical coincident results. Firstly, the update rate was recorded directly from the planner command generation rate, and it is presented also as the ratio to the desired ones. It represents the complexity of the tested conditions. Secondly, the RMS jerk is computed from the third-order temporal differentiation of the recorded robot positional trajectories. A higher RMS jerk indicates rapid changes in robot accelerations/trajectories and implies a lower smoothness. Thirdly, the collision percentage is computed from the ratio of the number of collisions across the testing duration, and therefore, a lower percentage indicates a safer condition. In this work, for each condition, all matrices are computed from over 100 repetitions/goals.

B. Experimental Results in a Static Corridor

Fig. 7a presents that the MPL improves the performance of the motion planners in the narrow corridor testing environment. With the base DWA planner, the MPL reduces the RMS jerks by 10% on average (p-value $<$ 0.05, t-test),

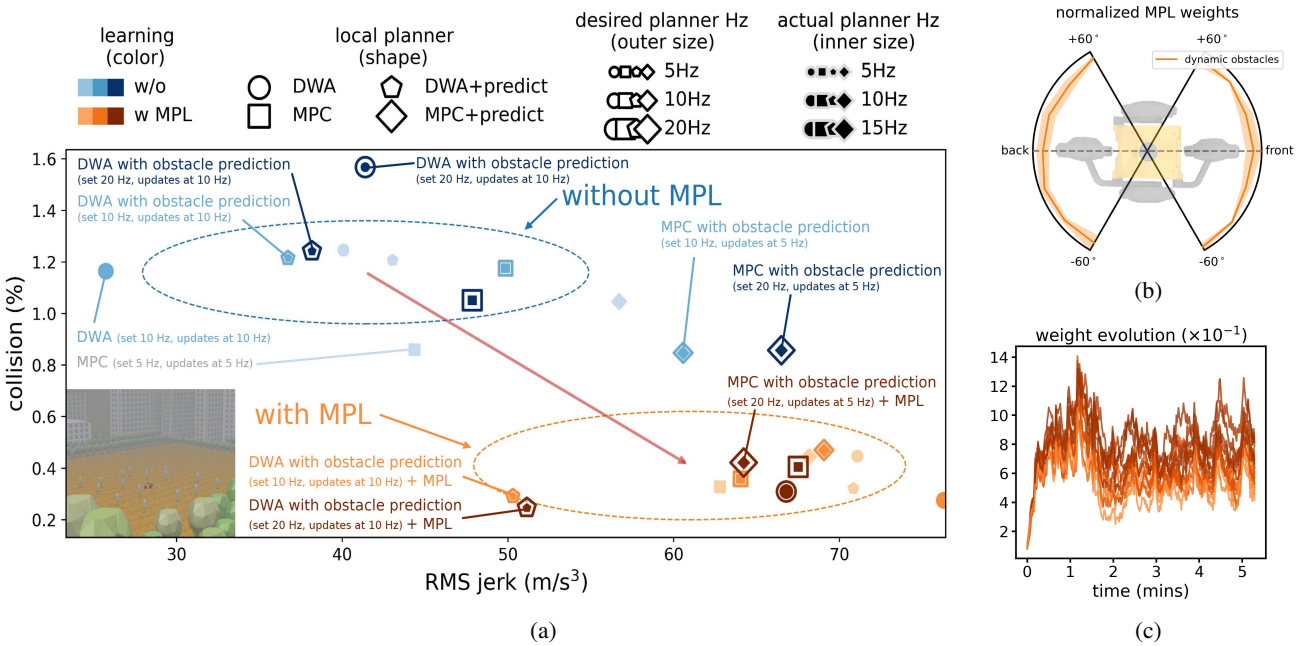


Fig. 8: (a) Collision rate (percentage) and RMS jerk obtained from different conditions in the free space with dynamic obstacles, where the conditions with the MPL are indicated by orange-ist colors and the ones without by blue-ish colors, the types of local motion planners by the marker types, the desired planning rates by the size of the outer edge, and the actual planning rates by the size of the inner marker. The red arrow pointing from the upper left to the lower right indicates that the MPL reduces collision while increasing motion jerk. (b) Average normalized weights obtained when the MPL converged, and (c) evolution of the weights in the first five mins. The video of the experiment is available at https://youtu.be/yF7n7kNFumo.

while the collision rate remains the same at 0%. Increasing the planning rate tends to improve the smoothness; however, with the current implementation and testing conditions, it is not possible to increase the planning rate over 10 Hz due to computational limitations. Thus, using the MPL can be a possible solution to achieve a smoother motion along a confined space when increasing the DWA planning rate is not possible.

While the DWA achieves nearly 0% collision rate, the MPC planner has around 4.5% collision on average due to the sensitivity to minor model mismatch (i.e., differential drive two-wheel robot model used for the planning and the actual four-wheel Capra robot) and a higher RMS jerk due to the commands at all timesteps being optimized. Besides, the maximum planning rate of the MPC is around 5 Hz. Adding the MPL, in this case, demonstrates reducing the collision by 1.6% (p-value < 0.05, t-test) in exchange for a 3.7% higher RMS jerk (p-value < 0.05, t-test). Thus, in the case of the MPC, using the MPL can also improve navigation safety in a narrow space.

For further analysis, the average converged weights are normalized and plotted in Fig. 7b. The figure reveals that the robot develops high weights on the front for proactive deceleration using the frontal obstacle information (i.e., the standing people) as cue; additionally, it develops relatively lower weights on the side for maintaining its position between the walls, behaving as the another cue. An additional test, presented in https://youtu.be/yF7n7kNFumo, demonstrates that simply tuning fixed equal weight values is prone to instability if the value is too high. For example, when all the weights were over 0.1 ($\approx$ two times the average value obtained by the MPL), the system became unstable as the robot always collided with the walls/standing people or could not reach the goals under the same testing time. Thus, the MPL can also be employed as a fast online autonomous weight adaptation mechanism, where the weights converged in $\approx$ 1 min as depicted in Fig. 7c. Note that after the weights converged, the correlation-based learning mechanism would tend to slightly increase them, while the decay term in Eq. 2 counteracted this by decreasing their values. This interplay could dynamically balance the weight changes and prevented the weights from getting stuck at a value that wouldn't allow the robot to reach the acceptance radius.

C. Experimental Results in a Free Space with Dynamic Obstacles

Fig. 8a presents that the MPL manages to improve the performance of the motion planners also in the testing environment with dynamic obstacles. With all local motion planner types (the DWA, MPC, DWA with obstacle prediction, and MPC with obstacle prediction), the MPL reduces the collision by 0.76% on average ($\approx$ 70% improvement, p-value < 0.05, t-test). However, this is achieved by the trading of motion smoothness, causing a

TABLE II: Comparison of different proactive methods.

Method	Type	Proactive information	Convergence (#trials)	Update rate (Hz)	Reported improvement		
					Smoothness	Collision	Others
[5]	model-based	obstacle prediction	n/a	1.67	n/a	✓	n/a
[6]	model-based	obstacle prediction	n/a	10	✓	✓	n/a
[7, 8]	model-based	obstacle prediction	20k	5–8	n/a	✓	path length, time
[10]	RL	environment reward	>200†	n/a	n/a	✓	robustness
[11]	RL	environment reward	4k	20–40	n/a	✓	time
[12]	SL	human demonstration	1	n/a	✓	n/a	n/a
[20]	UL	direct sensory feedback	≈ 7	n/a	n/a	n/a	input association
[21, 22]	UL	direct sensory feedback	n/a	n/a	n/a	✓	n/a
[19]	UL	direct sensory feedback	10–50	n/a	✓	n/a	path length
[18]	UL	direct sensory feedback	n/a	n/a	✓	n/a	time
[17]	UL	processed sensory feedback	10	30	n/a	✓	n/a
[16]	UL	processed sensory feedback	3–5	10	n/a	✓	n/a
MPL	**UL**	**local planner command** (adaptable proactive feedback)	**5–35**	**30–50**	✓	✓	**robustness**

† 200 runs were used to optimized with 100 parallel agents, each agent training converged after 1 million learning steps.

50% increase in RMS jerk (p-value < 0.05, t-test). In this case, to enhance collision avoidance in the testing dynamic environment, using the MPL was found to be more effective than increasing the desired planning rate and using simple obstacle predictions. Increasing the desired planning rate did not provide any significant improvement as the maximum actual planning rates were limited to around 5 Hz. Similarly, using a simple obstacle prediction model did not show a significant improvement due to the unpredictable nature of the setup environment (e.g., different random speed and curved walking paths). Therefore, using the MPL is an effective solution to improve collision avoidance in complex environments, where increasing the planning rate is impossible and the moving obstacles are unpredictable.

For further analysis, the average converged weights are normalized and plotted in Fig. 8b. The figure shows that, because the dynamic obstacles can be presented in all directions, the robot develops the weights in all directions equally to proact to those cues. An additional test, presented in https://youtu.be/yF7n7kNFumo, demonstrates that, in this testing environment, simply tuning fixed equal weight values is also prone to instability if the value is too high. For example, when all the weights were over 1.4 ($\approx$ 1.5 times the average value obtained by the MPL), the system became unstable as the robot merely reacted to avoid obstacles and rarely moved toward goals, resulting in 50% fewer target reached under the same testing time. Thus, the MPL can also be employed as a fast online autonomous weight adaptation mechanism in the environment with moving obstacles, where the weights converged in $\approx$ 2 mins as depicted in Fig. 8c. Note that after the weights converged, they were continually and slightly adapted to deal with the specific presence of different states of the crowd. Given that there were various feedback combinations in the free space with the dynamic obstacles than those in the narrow corridor and the learning mechanism also depended on the environment, the weights in Fig. 8c oscillated more than those in Fig. 7c.

V. Discussion and Conclusion

To address the limitations of local motion planners, which are often constrained by update rates of 1-10 Hz [5–8], this study presents the multiple proactive behavior learning (MPL) module. The MPL maps sensory feedback to outputs at higher update rates[1] (30-50 Hz), training autonomously and unsupervisedly with correlation-based learning. This allows rapid development of proactive actions using obstacle feedback and balances different proactive behaviors, such as improving smoothness in narrow spaces and enhancing obstacle avoidance in dynamic environments. Having an access to merely the planner command, the MPL improves obstacle avoidance using the obstacle avoidance cost included in Eq. 5 as well as improves motion smoothness even when the smoothness cost is excluded from Eq. 5. The MPL has been also successfully transferred to the physical robot, improving system robustness as depicted in Figs. 1 and 5, and https://youtu.be/yF7n7kNFumo.

Compared with the other UL-based state-of-the-arts [16–22], as summarized in Table I and Table II, the MPL is the only approach that achieves multiple proactive behaviors/multi-objective behaviors for smoothness and obstacle avoidance improvement. The MPL utilizes an adaptable local planner command as an adaptable proactive information to learn and switch between multiple proactive behaviors (reduced motion jerk and/or improved obstacle avoidance). In contrast, other UL-based approaches employ direct/processed sensory feedback as their proactive information, leading to the development of uni-purpose behaviors.

Compared with the other state-of-the-art methods summarized in Table II, which aim to obtain a set of near-global optimal parameters using additional obstacle prediction model [5–9], true targets/reward function [10–12], or numerous training samples/long training time [10, 11], the MPL aims to, without those requirements, quickly learn and continuously adapt local optimal parameters using simpler control to deal with different local conditions.

Unlike model-based approaches [5–9] that use assumptions to derive obstacle prediction models, the MPL learns unsupervisedly without any additional model to achieve and balance multiple proactive behaviors (i.e., improved smooth navigation and avoiding obstacles). As a plug-and-play UL-based approach, the MPL does not introduce any additional objectives/rewards/cost functions, unlike SL and RL approaches that depend on human demonstrations (SL) [12] or predefined reward functions (RL) [10, 11]. Additionally, the MPL converges quickly within minutes, whereas other RL-based approaches [10, 11] require many training samples and significantly longer training times. This allows the MPLto continuously develop and switch between multiple proactive strategies in a lifelong learning and adaptation fashion.

Currently, the MPL module uses solely obstacle-related information, limiting it to the development of proactive behaviors based on detected obstacles. This restricts the robot from achieving certain behaviors that require more input information. For example, learning smooth acceleration toward new goals in a narrow corridor or better collision avoidance in dynamic environments requires additional sensory feedback, such as the state of the robot, the environment, or goals. Introducing more sensory feedback is therefore expected to improve results and allow the study of other metrics beyond obstacle avoidance and smoothness.

Despite this limitation, the study demonstrates that the MPL can autonomously and quickly learn neural control weights (input-output mapping) and continuously adapt them to different environments throughout its operating lifetime. It enhances an arbitrary motion planning by compensating for limited planning rates and model mismatches. Since the MPL operates in the obstacle detection-velocity command space, it can potentially be applied to other types of robots [2, 16, 17] by incorporating specific kinematic models to map velocity commands to robot actuator commands. In the end, this study highlights another application of unsupervised learning in training a model-free neural control to cooperate with existing model-based control approaches, such as DWA, MPC, and those with obstacle prediction.

ACKNOWLEDGMENTS

This work was funded by Odense Robotics under the LOFT project (grant no. 1022-00014B). We thank Capra Robotics ApS, NCC ApS, Hecto Drone ApS, Buildcode ApS, Desupervised ApS, Lorenz Technology ApS, Meili Robots ApS, Robstruct ApS for the overall concept and discussion. This work was also supported by the fellowship from Vidyasirimedhi Institute of Science and Technology.

REFERENCES

[1] "Capra robotics," https://capra.ooo/, accessed: 28-02-2024.

[2] A. Chakrabarty, V. Stepanyan, K. S. Krishnakumar, and C. A. Ippolito, "Real-time path planning for multi-copters flying in utm-tcl4," in *AIAA Scitech 2019 Forum*, 2019, p. 0958.

[3] Z. He, J. Wang, and C. Song, "A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures," *arXiv preprint arXiv:2108.13619*, 2021.

[4] 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.

[5] K. Majd, S. Yaghoubi, T. Yamaguchi, B. Hoxha, D. Prokhorov, and G. Fainekos, "Safe navigation in human occupied environments using sampling and control barrier functions," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5794–5800.

[6] Z. Jian, Z. Yan, X. Lei, Z. Lu, B. Lan, X. Wang, and B. Liang, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3679–3685.

[7] P.-T. Singamaneni and R. Alami, "Hateb-2: Reactive planning and decision making in human-robot co-navigation," in *International Conference on Robot & Human Interactive Communication, 2020*, 2020.

[8] G. Ferrer and A. Sanfeliu, "Anticipative kinodynamic planning: multi-objective robot navigation in urban and dynamic environments," *Autonomous Robots*, vol. 43, no. 6, pp. 1473–1488, 2019.

[9] T. Fraichard and V. Levesy, "From crowd simulation to robot navigation in crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 729–735, 2020.

[10] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[11] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, and Z. Zhou, "Robot navigation in a crowd by integrating deep reinforcement learning and online planning," *Applied Intelligence*, pp. 1–17, 2022.

[12] X. Xiao, B. Liu, G. Warnell, J. Fink, and P. Stone, "Appld: Adaptive planner parameter learning from demonstration," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4541–4547, 2020.

[13] R. S. Sutton and A. G. Barto, "Toward a modern theory of adaptive networks: expectation and prediction." *Psychological review*, vol. 88, no. 2, p. 135, 1981.

[14] S. Dasgupta, F. Wörgötter, and P. Manoonpong, "Neuromodulatory adaptive combination of correlation-based learning in cerebellum and reward-based learning in basal ganglia for goal-directed behavior control," *Frontiers in neural circuits*, vol. 8, p. 126, 2014.

[15] B. Porr and F. Wörgötter, "Strongly improved stability and faster convergence of temporal sequence learning by using input correlations only," *Neural computation*, vol. 18, no. 6, pp. 1380–1412, 2006.

[16] V. Jaiton, K. Rothomphiwat, E. Ebeid, and P. Manoonpong, "Neural control and online learning for speed adaptation of unmanned aerial vehicles," *Frontiers in neural circuits*, vol. 16, 2022.

[17] J. Homchanthanakul and P. Manoonpong, "Continuous online adaptation of bioinspired adaptive neuroendocrine control for autonomous walking robots," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 1833–1845, 2021.

[18] F. Haarslev, D. Docherty, S.-D. Suvei, W. K. Juel, L. Bodenhagen, D. Shaikh, N. Krüger, and P. Manoonpong, "Towards crossmodal learning for smooth multimodal attention orientation," in *International Conference on Social Robotics*. Springer, 2018, pp. 318–328.

[19] D. Shaikh and P. Manoonpong, "A neural circuit for acoustic navigation combining heterosynaptic and non-synaptic plasticity that learns stable trajectories," in *International Conference on Engineering Applications of Neural Networks*. Springer, 2017, pp. 544–555.

[20] B. D. Northcutt, J. P. Dyhr, and C. M. Higgins, "An insect-inspired model for visual binding i: learning objects and their characteristics," *Biological cybernetics*, vol. 111, no. 2, pp. 185–206, 2017.

[21] B. Porr and F. Wörgötter, "Fast heterosynaptic learning in a robot food retrieval task inspired by the limbic system," *Biosystems*, vol. 89, no. 1-3, pp. 294–299, 2007.

[22] T. Kulvicius, B. Porr, and F. Wörgötter, "Chained learning architectures in a simple closed-loop behavioural context," *Biological Cybernetics*, vol. 97, no. 5, pp. 363–378, 2007.

[23] R. L. Guimarães, A. S. de Oliveira, J. A. Fabro, T. Becker, and V. A. Brenner, "Ros navigation: Concepts and tutorial," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 121–160, 2016.