Learning to Improve Multi-Robot Hallway Navigation

Jin Soo Park[†] **Brian Y Tsang**[†] **Harel Yedidsion**[†]

Garrett Warnell[‡] **Daehyun Kyoung**[†] **Peter Stone**[†][*]

[†]The University of Texas at Austin [‡]Army Research Laboratory [*]Sony AI

Abstract: As multi-robot applications become more prevalent, it becomes necessary to develop navigation systems which allow autonomous mobile robots to efficiently and safely pass each other in confined spaces. Existing navigation systems, such as the widely used ROS Navigation Stack, usually produce safe, collision free paths in static environments. However, these systems are not perfect, and when multiple mobile robots simultaneously navigate in narrow spaces, collisions and turnarounds are not uncommon. Fine-tuning and enhancing such navigation stacks is not as simple as it looks since they are made up of multiple layers of code, and there exists a tradeoff between optimizing for **efficiency**, i.e. minimizing time to destination (TTD) vs. optimizing for **safety**, i.e. minimizing collisions, with each objective leading to a different combination of parameter values. In this paper we develop a methodology to improve existing navigation stacks with regards to both objectives, without tuning their parameters, while preserving their inherent safety control properties. Our proposed approach is a decentralized learning-based approach that is geared toward real world robotic deployment, by requiring little computing resources. It is agnostic of the underlying navigation stack and can adapt to different types of environmental layouts (i.e., hallway structures).[1]

Keywords: Multi-Robot Collision Avoidance, Blackbox Navigation System, Gaussian Process Regression

1 INTRODUCTION

Coordinating the motion plans of multiple robots is an important and challenging problem. Service robots in office buildings, hotels, hospitals and fulfillment centers share a confined physical space and must navigate past each other safely and efficiently to reach their individual goal destinations. In narrow spaces such as hallways, standard navigation stacks plan a path that avoids obstacles on both sides and otherwise position the robot in the center of the hallway.

When two uncoordinated robots approach each other from opposite sides, they perceive each other as obstacles and either turn around or try to maneuver in the small gap between the other robot and the wall. This behavior is inefficient and even risky as it may cause collisions with either the wall or the other robot.

Tuning the internal parameters of the navigation stack, such as the costs of obstacles or the inflation of the base footprint, can change the stack's behavior to optimize for different objectives. However, it is not clear what the perfect balance between safety and efficiency is. When optimizing for safety, inflating these parameters results in 100% turnarounds and no collisions, but the paths the robots take become highly inefficient. When optimizing for efficiency, these parameter values are decreased, which allows the robots to attempt passing each other, but this results in some collisions. The inability of the navigation stack to effectively deal with dynamic obstacles, regardless of their internal parameter configuration, encourages developing coordination mechanisms for multi-robot navigation, ideally ones that do not require fine-tuning internal parameters and can generalize to different environments.

[1]A video illustrating our approach: `https://youtu.be/7IQPOQrJuSI`

A straightforward coordination solution would be to always keep to one side; this solution might work for a straight hallway with no obstacles, but it will fail in less structured environments where benches, trash cans, and other objects contains the locations where passing is.

Our approach modifies the local motion plan of one robot when detecting an interaction; the approach chooses a waypoint for the robot to move to and wait at while the other robot passes. The challenge is to decide which waypoint will produce the best results in given hallway based on the robot positions. To better estimate the usefulness of potential waypoints, we frame the problem as a Contextual Bandit learning problem [1], a variation of the general reinforcement learning problem that involves learning a policy but where each action affects only the immediate reward.

The contributions and benefits of our approach include:

- It is designed to be a lightweight, decentralized multi-robot coordination method that relies on a low-dimensional state representation, which enables both fast learning with a small number of examples and fast computation and decision making in real world deployment.
- It can be used to enhance any navigation stack with better maneuverability in narrow hallways, without tuning its internal parameters, or losing any of its inherent safety features.
- It is designed to adapt to any environment without having to manually engineer a solution for every hallway structure.
- The empirical results indicate that the learned method leads to more successful and efficient interactions, shorter times to destination, fewer collisions, and fewer planning failures compared to three other navigation control methods.

2 Related Work

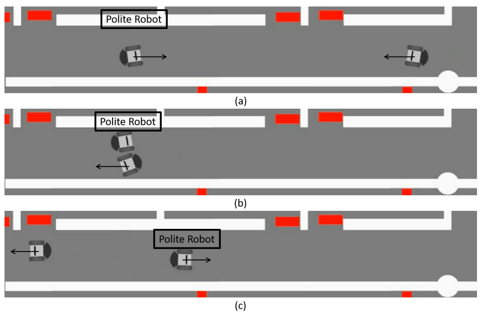
Coordinating the motion plans of multiple robots to avoid collisions is addressed by different AI approaches. Deliberative approaches try to avoid conflicts by coordinating global plans [2, 3].

Reactive approaches on the other hand resolve conflicts as needed when local plans conflict at runtime. The most popular reactive solution for multi-robot collision avoidance is a family of velocity obstacle algorithms. Fiorini et al. [4] show that by choosing a reactive velocity outside of the velocity obstacle of other robots within a small time frame, the robot can avoid collisions. Van den Berg et al. [5] propose a reciprocal behavior to solve the oscillating motion problem of the velocity obstacle algorithm. Van den Berg, [6] presents the optimal reciprocal collision avoidance (ORCA) method which was shown to be successful for navigating in a crowded environment. However, all of these methods assume a holonomic robot with perfect sensing. Some research papers try to relax the constraint of velocity obstacle methods. Alonso-Mora et al. [7] tries to relax the holonomic robot constraint by limiting the velocity space, and Hennes et al. [8] relax the perfect sensing constraint of ORCA by inflating the radius of the robot with a bounded localization error. In general, ORCA-based approaches require hyper-parameter tuning and direct control of the velocity of the robot. This method is not applicable when the navigation system of the robot cannot be directly controlled by giving velocity commands to the robot.

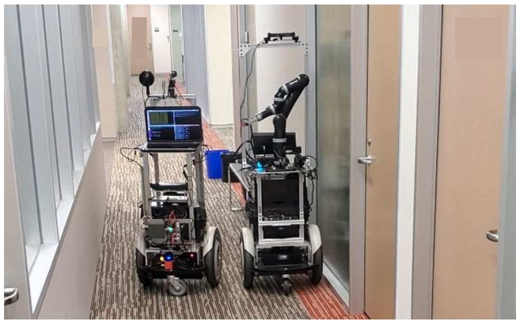
Recent papers propose the use of machine learning to learn a collision avoidance policy. Long et al. [9] uses supervised learning to learn the mapping from lidar readings and preferred velocity to the probability distribution of safe velocity commands, aiming to mimic ORCA without hyper-parameter tuning. Chen et al. [10] use reinforcement learning to further improve the network that learns the trajectory of ORCA. Long et al. [11] directly learn velocity commands from the raw lidar information with a two stage reinforcement learning scheme. Tan et al. [12] further improves the work by Long et al. [11] by incorporating global information into the network. Lin et al. [13] improve the performance of a reinforcement learning policy by introducing a centralized learning and decentralized executing framework.

One common feature of these approaches is that they try to build the entire local navigation system from scratch. In addition, they rely on a high-dimensional state space, which requires a large number of training examples, and significant computational resources when deployed.

Recently, efforts have been made to combine the advantages of existing rule-based systems with reinforcement learning policies. Ding et al.[14] combine differential rule-based outputs with a reinforcement learning policy, but still require access to low-level velocity commands. Fan et al. [15, 16]



(a)



(b)

Figure 1: (a) The robots move toward each other in a simulated hallway. For safe passage, the "polite" robot waits in a safe position until the other robot has completely passed, as the width of the corridor is barely enough for the two robots to pass through. (b) Picture of the real robot deployment.

use heuristics to determine when to completely switch control between a rule-based system and a learned policy that directly controls velocity commands.

Despite the great success of reinforcement-learning-based collision-avoidance policy, they mostly aim to map a state to a steering command, thus removing the advantage of safety guarantees that classical rule-based controllers provide. In this paper, we focus on developing a computationally light collision-avoidance policy on top of any black-box navigation system which provides high-level actions (such as "move to location") but not direct velocity commands.

3 Methodology

In this section we formulate the hallway passing problem, provide details of our solution method - the adaptive policy, and briefly describe the alternative methods we compared against. The alternative methods consist of the vanilla ROS navigation stack, the ROS Social Navigation stack [17], and a hand-coded solution we call the fixed strategy.

3.1 Problem Definition

We formulate the hallway passing problem as an interaction between two robots moving in opposite directions in a narrow space, each moving toward a destination that is on the opposite side of the other robot. The goal is to minimize the time to destination (TTD) of both robots while also minimizing the collision and turnaround rate.

$$min\,\mathbb{E}[\sum(w_j \cdot TTD_j) + c_1 \cdot \mathbf{1}_{collision} + c_2 \cdot \mathbf{1}_{turnaround}] \tag{1}$$

where w_j denotes the priority of robot j, TTD_j denotes the TTD of robot j, c_1 denotes the penalty for collision and c_2 denotes the penalty for turnaround. $\mathbf{1}_{event}$ is an indicator function which outputs a value of 1 if there was an event and 0 otherwise.

We describe the approach for two robots, but this can be extended to multiple-robot scenarios by grouping the agents with similar direction and poses and having the members of the group follow the behavior of their leader similar to the study done by He et al.[18].

3.2 Adaptive Policy

In this section, we describe our solution to the hallway passing problem, "**the adaptive policy**".

We train the navigation control policy in Gazebo [19] simulations where virtual robots pass each other in different hallway structures. In each training episode, the learning robot chooses a waypoint to move to, receives a numerical reward according to the outcome of that choice, and updates its policy based on that reward for future interactions. We model the policy using Gaussian Process Regression (GPR) [20]. Once the policy is learned and tested in simulation, it is transferred to the real robots and tested in real-world passing scenarios.

In the scenario where two robots need to pass each other in a narrow hallway, we hypothesize and design our solution method according to the notion that having one robot wait in a safe place until the other robot has completely passed it provides the best tradeoff between efficiency and safety. Therefore, the adaptive policy, when faced with the hallway passing scenario, modifies the navigation control of one robot to move it to a waypoint until the other robot passes by. We call the robot that modifies its plan the "polite robot", and the other robot that uses the regular navigation stack the "regular robot".

To enable the estimation of multiple waypoints very quickly while maintaining the ability to generalize, we chose a minimal state representation of four measurements to evaluate the usefulness of a waypoint. The features that define the context of a waypoint consist of these four measurements:

1. d_1 -The distance between the waypoint and polite robot ($Robot_1$ in Figure 2).
2. d_2 -The distance between the waypoint and regular robot ($Robot_2$ in Figure 2).
3. d_3 -The distance between the waypoint and right wall relative to $Polite\ robot$'s motion.
4. d_4 -The distance between the waypoint and left wall relative to $Polite\ robot$'s motion.

Intuitively, d_1 and d_2 provide information about efficiency and the usefulness of the waypoint in minimizing the TTD, while d_3 and d_4 indicate how safe the point is in terms of allowing the other robot to pass.

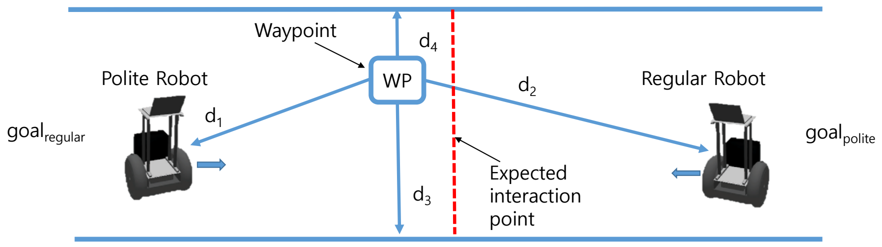


Figure 2: The four features that represent the waypoint. d_1, d_2 represent the distance from the waypoint to each robot. d_3, d_4 represent the distance between the waypoint and the closest obstacle on each side perpendicular to the line connecting the two robots.

Finding the best waypoint to move to is hard since we are assuming that the navigation system of the robot is unknown. We solve this problem by using a GPR model [21] to model the reward from the characteristics of the waypoint and pick the best we have.

The GPR model is a kernel-based Bayesian method that defines the posterior distribution of a function from a given set of data, X.

Let $r_j(t)$ be a reward of waypoint j at episode t and $\mathbf{x}_j$ be a feature vector representing the waypoint, j. Then, we can define the posterior distributions , $f(x_j)$, at waypoint j with a GPR model.

A GPR model is defined as follows [22]:

$$f(x) \sim GP(\mu(x), k(x, x'))$$

where $\mu(x)$ denote the mean function and $k(x, x')$ denotes the kernel function.

For our model, we chose the popular radial basis function (RBF) kernel.

$$K(x, x') = exp(-\frac{(x - x')^2}{r^2})$$

3.2.1 The training Process

Our training data for the GPR model was collected by repeatedly running scenarios of two robots attempting to pass each other in a hallway. The two robots start at different locations 10m apart, which is slightly more than the robot detection range of approximately 8m.

The robots face each other in the hallway and are given goal points 16m ahead of their starting locations. One of the robots, the regular robot, uses the basic ROS navigation stack throughout all of the training episodes. The other, polite robot, starts with the basic ROS navigation policy and uses online learning to train the GPR model; at each episode, it picks a waypoint according to the model it has learned up to that point, and it uses the reward from each episode to update its policy.

The polite robot samples waypoints uniformly at random from the free space around it and selects the best ranked waypoint according to the GPR model. The polite robot moves to that point and waits as the regular robot continues to navigate to its goal point. If the regular robot successfully passes the polite robot (i.e., without collisions or turnarounds), the polite robot resumes navigation towards its original goal.

The features of our training data contain the four measurements of the waypoint picked by the polite robot. The labels of our training data are calculated using a reward function that quantifies the success of that waypoint.

$$reward = -TTD_{polite} - TTD_{regular} - 1000 \cdot \mathbf{1}_{collision} - 100 \cdot \mathbf{1}_{turnaround}$$

TTD_{polite} and $TTD_{regular}$ denotes the time for $robot_n$ to reach its destination, in seconds. We set w_j=1 for both robots to indicate that they have equal priority. This reward function heavily penalizes collisions and mildly penalizes navigation failures (i.e. turnarounds). The GPR model is trained to predict this reward using the four measurements.

Under our modified navigation controls, the robot uniformly samples 200 waypoints from a 4m square box region centered around the polite robot. The set of waypoints are evaluated by the trained GPR model, and the robot selects the one which maximizes the expected posterior reward.

We used a multi-robot Gazebo simulation to train the GPR model. The GPR model was trained on 1000 episodes in the straight hallway, scenario 1, using an ϵ-greedy algorithm with $\epsilon = 0.05$. A graphical representation of each scenario is shown in Figure 3.

3.3 Baseline and Alternative Methods

The proposed adaptive policy is compared against three other navigation control methods:

- The *Vanilla ROS navigation stack*, or simply *Vanilla navigation*, is a policy in which both robots only utilize the basic ROS navigation stack, arguably one of the most commonly used robot navigation control systems. The navigation stack uses hyper-parameter values which are tuned to enable a single robot to reliably navigate in a building environment and avoid static obstacles. Since this method is just the basic ROS navigation stack, it serves as our baseline.
- The *social navigation stack* augments the *vanilla ROS navigation stack* with a repulsive potential field to avoid collisions. The social navigation stack adds extra costs to the local cost map along the estimated trajectory of detected moving objects. We supply the ROS social navigation stack [23] with additional communication between agents to accurately indicate their relative locations.
- The *fixed strategy* is a hand-crafted policy that focuses on safety over TTD. This navigation method works similarly to the adaptive method, but instead of having a trained policy pick the polite robot's parking spot, the polite robot always picks the closest waypoint out of a set of waypoints predefined by the user. Since the same set of waypoints is used for any situation, we only chose parking spots outside of the narrow hallways as they are safe for almost any scenario. This strategy prevents collisions at the expense of longer TTD since the polite robot has a very limited number of waypoints to choose from. Additionally, the fact that this method requires a user-made list of safe waypoints means that constant human assistance is required if robots are in a new environment or if the environment is dynamic.

Additionally, the following extension of the adaptive policy is also tested:

- The *Adaptive policy with negotiation* is a policy in which both robots evaluate waypoints in their vicinity using the adaptive policy. The robot closer to the waypoint with the highest expected reward becomes the polite robot, and the other robot takes the role of the regular robot. Determining the waypoint with the highest expected reward is done by communicating each robot's best expected reward to each other.

Table 1: Overall statistics of each navigation method in simulation.

	Vanilla Navigation	Social Navigation	Fixed Strategy	Adaptive Policy	Adaptive with Negotiation
Efficiency	0.270	0.293	0.624	0.776	0.803
% collisions	7.5%	41.5%	0%	3%	0%
% turnarounds	92.0%	35.75%	0%	0%	0%

4 Empirical Evaluation

First, we carefully designed four different scenarios with unique properties. In scenario 1, the hallway near the polite robot is straight without any free spaces to park at or obstacles. Scenario 2 has an alcove and a small obstacle near the polite robot. Scenario 3 has an obstruction near the expected interaction point. Finally, scenario 4 has both an obstruction and an open space near the expected interaction point. Figure 3 illustrates these four scenarios.

To show that the adaptive policy can be generalized to other indoor environments, our results sections will display results from an adaptive policy trained on scenario 1 and tested on the rest of the scenarios. See Appendix A for experimental results of models trained on the other scenarios.

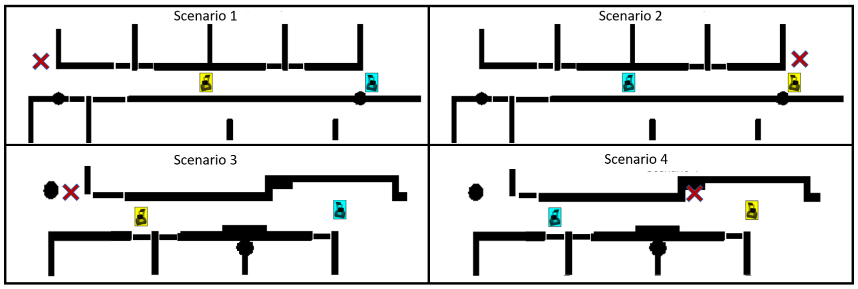


Figure 3: Graphical representation of each scenario. The yellow represents the polite robot, and the blue represents the regular robot. The red cross shows the predefined waypoints of the fixed strategy.

The adaptive policy is compared to the baseline, the vanilla navigation, and three other alternative approaches: fixed strategy, social navigation stack, and adaptive policy with negotiation. We compared the performance of each policy based on three objectives:

1. Efficiency: The ratio of (a bound on) the best possible reward to the average reward of the two robots with policy, p. As a best-case bound for this purpose, we use the average TTD when a single robot traverses the hallway without any other robot present, using the vanilla navigation, which we denote as t_{lb}. Thus, the reported efficiency is $t_{lb}/reward_p$. The (generally unachievable) upper bound on efficiency is 1.
2. Safety: (1 - Rate of collisions).
3. Planning failure: Rate of turnarounds. Robots turn around when they can not find a feasible path to pass the other robot. Thus, the turnaround rate represents the ratio of planning failure of each policy.

4.1 Simulation Results

The adaptive policy was tested for 100 episodes in each scenario. The relative position of each robot between the walls and initial orientations were sampled from [-0.3, 0.3]m, [-15, 15]$^\circ$ respectively.

The performance of the adaptive policy, baseline, and alternative approaches is shown in Table 1 and Figure 4. The "training scenario" represents the result of scenario 1, and the "test scenario" represents the aggregated results of the other three scenarios. Table 1 shows the percentage of each possible outcome that was observed over 400 simulation episodes. The box and whisker plots in Figure 4 show the efficiency of each policy, ignoring trials with collisions. We used Student's T-test to test for statistical significance between the different methods in terms of efficiency.

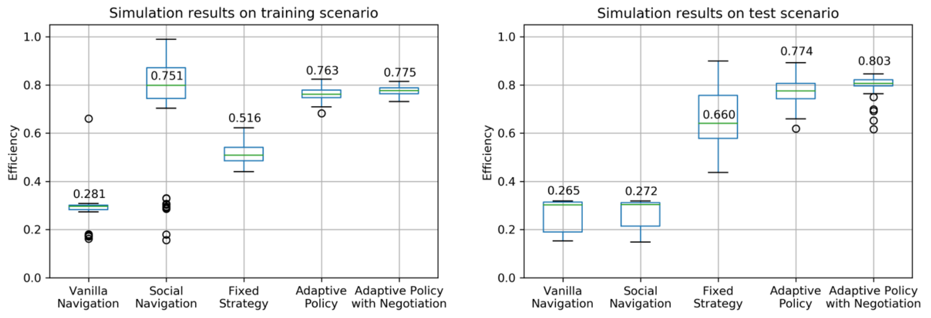


Figure 4: Box and whisker plot showing the efficiency of each policy across the trials without collision. The numbers represent the average efficiency of each policy.

The simulation results indicate that the social navigation stack exhibits better efficiency than the vanilla navigation with a trade-off of safety. The average efficiency on the training scenario is 0.751, meaning that the robots, on average, move at a speed that is 75.1% the speed they would travel if the other robot weren't present. However, the average efficiency on the test scenarios is 0.272, which is not significantly different from the vanilla navigation (p-value: 0.172). Due to its inability to generalize well, it is not an effective method for multi-robot navigation in a very narrow hallway.

The average efficiency of the fixed strategy is 0.516 in the training scenario and 0.660 in the test scenarios. The wide inter-quartile range of fixed strategy indicates that its efficiency is heavily dependant on the quality of the predefined waypoints and the location of free spaces in the building.

The adaptive policy provides a good balance between efficiency and safety. The efficiency of the adaptive policy is 0.763 in the training scenario and 0.774 in test scenarios while maintaining a relatively low collision rate of 3%. It outperforms the other alternative policies except for its extension, the adaptive policy with negotiation, with statistical significance (p-value less than 0.001).

The adaptive policy with negotiation achieves the best results in terms of both efficiency and safety. It achieves the best efficiency (0.775 / 0.803) while maintaining collision and turnaround rates at 0%. Since this policy considers waypoints from both robots' vicinities, the policy's overall performance is strictly better than the adaptive policy; the simulation results reflect this.

4.2 Robot Experiments

The physical robots used for deployment have a Segway base and a 2D Hokuyo Lidar, used for localization. The software architecture used is built on top of the Robot Operating System (ROS) [24]. The system provides an ability to autonomously navigate to a certain Cartesian coordinate given a current position and a 2D grid map.

We directly transfer the adaptive policy from simulation to real robots without any modification. The policies were tested over 40 real episodes, 10 from each scenarios. The results show that our method does not suffer from the sim-to-real problem presumably because the features used to represent waypoints in simulation are an accurate reflection of reality, the robot motion simulation is sufficiently representative to model the collisions. And the separation between high-level planning and low-level control helps our models to overcome the sim-to-real problem. [25] The results of testing the four navigation methods on the robots are displayed in Figure 5, and in Table 2.

Table 2: Overall statistics of each navigation method in real-robot deployment.

	Vanilla Navigation	Social Navigation	Fixed Strategy	Adaptive Policy
Efficiency	0.329	0.587	0.673	0.765
% collisions	37.5%	55.0%	0%	3%
% turnarounds	62.5%	25.0%	0%	0%

Social navigation on the real robots achieved the best average efficiency, 0.867, in the training scenario but also the highest overall collision rate, 55%, making it still unsuitable for solving the hall-

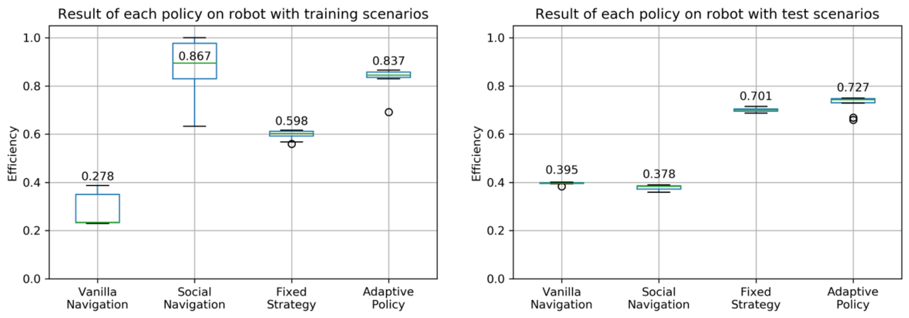


Figure 5: Box and whisker plot showing the efficiency of each policy without collision episodes in robot deployment. The numbers represent the average efficiency of each policy.

way passing problem. We have extensively tuned the hyper-parameters of the Social Navigation Stack such as radius of robot, footprint padding of both global and local planner, and xy tolerances, but we could not find any better configurations.

The fixed strategy remains a safe policy in real-robot deployment. The robots had no collisions or turnarounds, but the strategy also did not perform the fastest in any of the scenarios.

The safety of the adaptive policy increased in real experiments. The robots successfully avoided collision in all 40 episodes. In addition, the adaptive policy maintained its good efficiency with an average of 0.837 in the training scenario and 0.727 in test scenarios, making it the best overall policy in the real-robot hallway passing experiments. The adaptive policy with negotiation was not deployed with robots due to unreliable communication channels in initial trials and temporary loss of access to the hardware due to the COVID-19 pandemic.

5 Conclusion and Future Work

This study shows the potential of improving an existing robot navigation stack with a learnable, high-level control policy. We show that our proposed learning-based adaptive policy can be combined with the generic ROS navigation stack to improve its efficiency and safety without tuning its internal parameters and while preserving its safety properties. The adaptive policy is also designed to take in a low-dimensional state space, which allows for fast training and fast reaction times even when deployed on robots with limited computational resources. We compared the adaptive policy to three other methods: the vanilla ROS navigation stack, a social navigation stack, and a fixed strategy using predefined parking spots. Empirical evaluation both in simulation and on real robots indicates that our adaptive policy is more efficient than the other methods. Additionally, the adaptive policy maintains a very small collision ratio (3%) in simulation and even completely avoids collisions in our real-robot deployment. Furthermore, when the adaptive policy is implemented on both robots as in the adaptive policy with negotiation, it can obtain 0% collision rate in simulation. Finally, the adaptive policy, unlike the fixed strategy, does not need predefined parking zones and hence can easily generalize to new environments. This advantage was demonstrated in our experimental results as the adaptive policy, trained only on one scenario, performed well in three other unseen scenarios in both simulation and on real robots.

This paper opens up some interesting directions for future works. The adaptive policy may be able to be further improved by learned during deployment. Another interesting future direction is to explore how to effectively apply multi-agent learning for the hallway passing problem.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARO (W911NF-19-2-0333), DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Y. Jiang, H. Yedidsion, S. Zhang, G. Sharon, and P. Stone. Multi-robot planning with conflicts and synergies. *Autonomous Robots*, 43(8):2011–2032, 2019.

[3] K. W. Wong and H. Kress-Gazit. Need-based coordination for decentralized high-level robot control. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2209–2216. IEEE, 2016.

[4] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[5] 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*, pages 1928–1935. IEEE, 2008.

[6] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[7] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed autonomous robotic systems*, pages 203–216. Springer, 2013.

[8] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls. Multi-robot collision avoidance with localization uncertainty. In *AAMAS*, pages 147–154, 2012.

[9] P. Long, W. Liu, and J. Pan. Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics and Automation Letters*, 2(2):656–663, 2017.

[10] Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[11] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[12] Q. Tan, T. Fan, J. Pan, and D. Manocha. Deepmnavigate: Deep reinforced multi-robot navigation unifying local & global collision avoidance. *arXiv preprint arXiv:1910.09441*, 2019.

[13] J. Lin, X. Yang, P. Zheng, and H. Cheng. End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning. In *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 2493–2500. IEEE, 2019.

[14] W. Ding, S. Li, H. Qian, and Y. Chen. Hierarchical reinforcement learning framework towards multi-agent navigation. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 237–242. IEEE, 2018.

[15] T. Fan, P. Long, W. Liu, and J. Pan. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *arXiv preprint arXiv:1808.03841*, 2018.

[16] T. Fan, P. Long, W. Liu, and J. Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, page 0278364920916531, 2020.

[17] D. V. Lu, D. Hershberger, and W. D. Smart. Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–715. IEEE, 2014.

[18] L. He, J. Pan, W. Wang, and D. Manocha. Proxemic group behaviors using reciprocal multi-agent navigation. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 292–297. IEEE, 2016.

[19] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[20] M. P. Deisenroth. *Efficient reinforcement learning using Gaussian processes*, volume 9. KIT Scientific Publishing, 2010.

[21] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

[22] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[23] D. V. Lu. Ros social navigation layers. URL `http://wiki.ros.org/social_navigation_layers`.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[25] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

Learning to Improve Multi-Robot Hallway Navigation: Appendices

A Additional Experimental Results

In this section, we investigate the potential of the adaptive policy by comparing its performances under different conditions. The conditions are the following.

- General: The policy presented in the paper. This policy is trained on scenario 1 for a 1,000 episodes and tested on all scenarios to show how the adaptive policy can be generalized.
- Specialized: This policy consists of four adaptive policies trained on each specific scenario for 1,000 episodes and tested only on the **same** scenario. Therefore, in scenario 1, the "General" adaptive policy is equivalent to "Specialized". The performance of this policy can be seen as the best possible performance of the adaptive policy for a specific hallway.
- Complete: This policy is trained on all four scenarios for 4,000 episodes. This can be seen as the performance of training the adaptive policy on an entire floor.

Efficiency is measured without collision episodes. The resulting performance under each condition is shown in Figure 6 and Table 3.

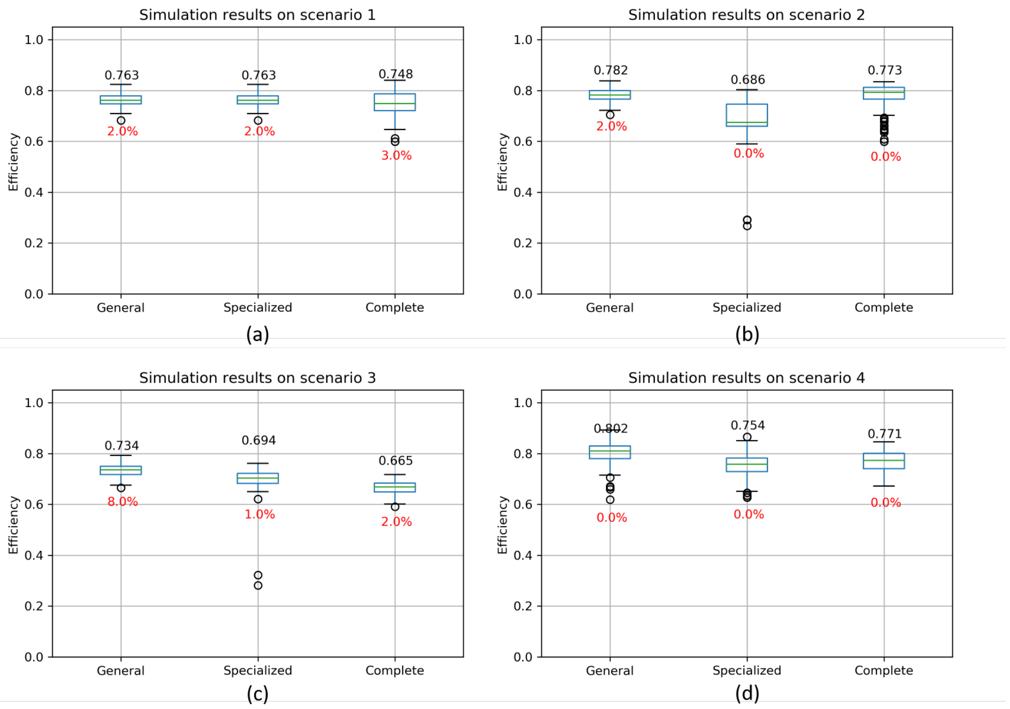


Figure 6: Box and whiskers plots showing the efficiency of general, specialized, complete adaptive policy on each scenario. The black number represents the average efficiency and the red number indicates the collision rate of each policy. Note that in (a) the results of General and Specialized are the same.

Both Specialized and Complete adaptive policy are safer than General. However, efficiency is lower for both since both choose more conservative waypoints for safer passage. This makes sense considering that in Section 3.2.1, we heavily penalize collisions (-1,000). Table 3 shows that more data (specialized and complete) does improve the model in terms of rewards.

Table 3: Overall statistics of adaptive policy with different conditions in simulation.

	General	Specialized	Complete
average efficiency	0.770	0.724	0.739
% collision	3.0 %	0.75 %	1.25 %
% turnaround	0.0 %	1.25 %	0.0 %

B Implementation Details

Hyper-parameter tuning: The Gaussian Process Regression (GPR) model requires hyper-parameters such as noise in likelihood, constant mean value, output scale and length scale of covariance module. which need to be tuned. We tune these parameters using the "gpytorch" [21] module with 100 data points from independent episodes.

Sampling waypoints: The adaptive policy samples 200 waypoints from the 4m x 4m square surrounding the polite robot and evaluates the expected rewards using the GPR model (See Section 3.2.1 for more context). However, the 4m x 4m square could contain unreachable areas such as a room beyond the walls. Therefore, we exclude these "unreachable" areas and only sample waypoints that the robot can get to. For the ease of computation, we created a map highlighting all areas of the floor that the robots can reach from the hallways; this was only done once before the robot was deployed.

C Model Visualization

In this section, we will show the ability of the adaptive policy to generalize through the heatmap of the Gaussian Process Regression (GPR) model.

Figure 7 shows the expected reward of **general** adaptive policy in four scenarios. Note that the waypoints are sampled from reachable points indicated by the regular ROS navigation stack. The general adaptive policy was only trained on Scenario 1, nevertheless the heatmap of waypoints in scenarios 2 and 3 indicate that it does learn to avoid waypoints near the obstacles such as the pillar (orange circle) or the bench (green box). In scenario 4, the expected rewards of waypoints are higher than other scenarios because this scenario contains a wide open space around the hallway.

D Multi-Robot Results

To show that our approach can generalize to situations with more than two robots, we ran some experiments with three robots passing in the hallway. In narrow hallways, situations that involve three robots having to pass each other will generally have one robot on one side and two robots on the other. To minimize time-to-destination, the single robot, while using the adaptive policy, should park while the other two pass. Table 4 shows the simulation results from a three-robot experiments using this methodology.

Future work includes generalizing to even more robots, tackling complex scenarios such as T-intersections, and fleshing out the communication between robots for deciding on which robots should park.

Table 4: Results of the three-robot scenario in simulation.

	Vanilla Navigation	Adaptive Policy
average efficiency	0.273	0.637
% collision	50 %	5 %
% turnaround	50 %	2 %

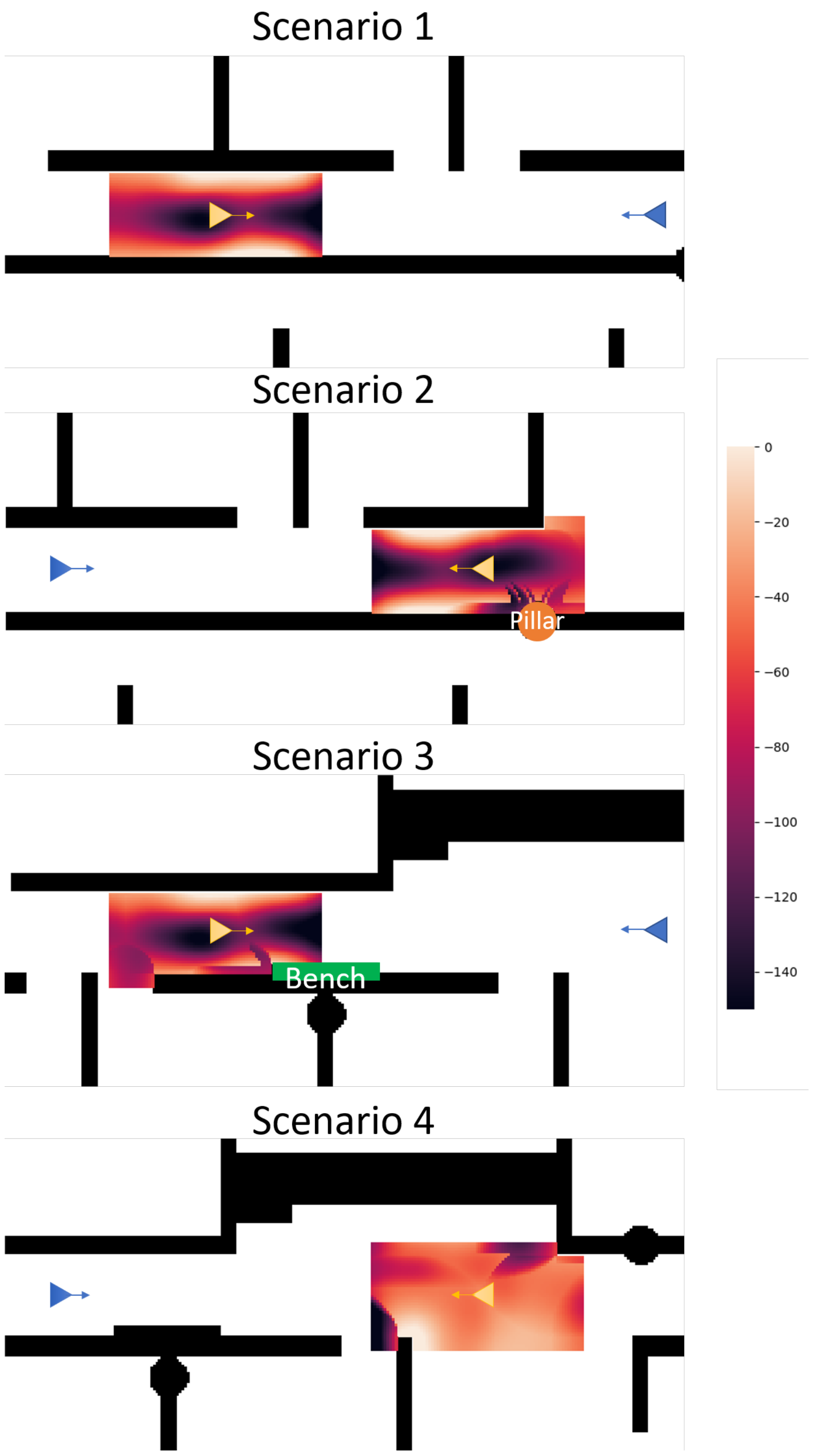


Figure 7: Heatmaps showing the expected reward of waypoints near the polite robot. The position of the polite robot (yellow) and the regular robot (blue) is represented as a triangle. The arrow on the triangle indicates the robot's orientation. The brighter colors in the heatmaps indicate the safer and more efficient waypoints.