Robot Navigation in Constrained Pedestrian Environments using Reinforcement Learning

Claudia Pérez-D’Arpino, Can Liu, Patrick Goebel, Roberto Martín-Martín, Silvio Savarese
Stanford University

Abstract: Navigating fluently around pedestrians is a necessary capability for mobile robots deployed in human environments, such as office buildings and homes. While related literature has addressed the co-navigation problem focused on the scalability with the number of pedestrians in open spaces, typical indoor environments present the additional challenge of constrained spaces such as corridors, doorways and crosswalks that limit maneuverability and influence patterns of pedestrian interaction. We present an approach based on reinforcement learning to learn policies capable of dynamic adaptation to the presence of moving pedestrians while navigating between desired locations in constrained environments. The policy network receives guidance from a motion planner that provides waypoints to follow a globally planned trajectory, whereas the reinforcement component handles the local interactions. We explore a compositional principle for multi-layout training and find that policies trained in a small set of geometrically simple layouts successfully generalize to unseen and more complex layouts that exhibit composition of the simple structural elements available during training. Going beyond wall-world like domains, we show transfer of the learned policy to unseen 3D reconstructions of two real environments (market, home). These results support the applicability of the compositional principle to real-world environments and indicate promising usage of agent simulation within reconstructed environments for tasks that involve interaction.

https://ai.stanford.edu/~cdarpino/socialnavconstrained/

1 Introduction

As mobile robots become more capable of executing a variety of tasks with broader applicability, their increasingly likely deployment in human environments demands seamless and safe integration into spaces with pedestrians such as buildings, homes and offices (Fig. 1c). In the path towards socially compliant mobile robots, research on human-robot co-navigation has made progress on planning and learning approaches for robots that can handle local interactions with moving pedestrians. While some researchers have focused on navigation problems among crowds in open spaces, navigation among pedestrians in indoor environments presents additional challenges due to space constraints that limit maneuverability and create complex patterns of pedestrian interaction. Narrow indoor spaces such as corridors, door exits and crosswalks, populated by humans, are extremely hard problems for any existing navigation solution. For these types of spaces, there is a need for new models that can navigate around pedestrians in tight spaces with constraints imposed by obstacles, objects, walls, doors and other common structural elements, based only on on-board sensor information.

In this paper we present a novel robot learning approach for navigation in human populated environments. Encouraged by the success of previous approaches [1, 2], we propose to combine a motion planner and a reinforcement learning (RL) policy to leverage their complementary strengths: the motion planner provides coarse static strategies that avoid local minima, while the RL policy learns to follow the path given by the planner, deviating from it to adapt the current conditions of the environment (e.g. moving pedestrians) as observed with on-board sensors, even in constrained indoor spaces such as corridors and doors. We instantiate our navigation approach and train it to control a simulated non-holonomic wheeled robot in multiple environments. Our system is trained on iGibson, the interactive Gibson simulator [3, 4] leveraging the Pybullet physics engine [5] providing realistic physics and virtual sensing in our multi-agent setup. The simulator includes multiple simulated pedestrians whose motion is generated based on social forces [6] with the ORCA [7] motion generator.

Preprint, 2020
We observe that typical indoor environments exhibit common geometric elements and diverse composition of these elements. With the aim of generalization to novel spaces, we explore a compositional principle for multi-layout training, in which a policy is trained in a number of simple, canonical, layouts, such corridors, or single door exit, as opposed to composed layouts with multiple and combined navigation challenges. We analyze the generalization capabilities of this approach to layouts not seen during training, and find that policies trained in the small set of geometrically simple layouts are also successful on more complex layouts exhibiting composition of these simple shapes. This insight results particularly relevant for typically built environments that are mainly composed by combinations of basic templates for layouts, such as hallways, rooms, door exits, and crosswalks. We generate these simple environments (denominated by prefix WALLS-) with geometric primitives in simulation. We then demonstrate transfer of the policies learned on WALLS (Fig. 1a) to two unseen 3D reconstructions of two real-world environments, denominated by the prefix MESH- (Fig. 1b). These reconstructions are real-scale 3D textured scans of built environments, such as a supermarket and an apartment. This demonstration raises research questions for agents that train in simplified abstractions of a more complex target model.

In summary, this paper presents the following contributions: (1) a learning approach and a simulation environment for the problem of goal-oriented robot navigation around moving pedestrians, that unifies ideas from reinforcement learning, planning and multi-agent simulation to enable human-robot co-navigation in constrained environments representative of real human indoor spaces; (2) a compositional multi-layout training regime using canonical walls-worlds layouts that generalizes to more complex environments; and (3) a demonstration of transfer of the learned policy on simplified walls-world to unseen 3D reconstructions of two real-world environments represented by scanned 3D textured meshes. These results support the compositional principle applicability to real-world environments and indicate promising frontiers for the emerging topic of agent simulation within reconstructed environments for tasks that involve human-robot interaction. Accompanying the paper, we present a video with demonstrations of the robot behaviors obtained in the experiments.

2 Related Work

Navigating safely among humans is a prerequisite for the deployment of mobile robots. This has motivated researchers in autonomous navigation and human-robot interaction to study different aspects of the navigation problem, such as navigation within crowds [8], communicating implicit and explicit navigation information with haptic devices [9], co-navigation [10] using anticipatory signals [11], navigation cues in human gaze [12], and the effects of distinct robot navigation strategies on pedestrians [13]. Despite this large tradition, safe navigation among humans is still an active research area.

Early solutions for navigation in human environments combine a motion planner [14] with a controller that executes the plan [15]. These approaches suffer from the so-called “freezing robot problem” [16, 17], in which the planner find all solutions to be below certain threshold of safety or feasibility, causing the robot to stop. This problem is more severe in domains with complex geometry and large number of pedestrians, where the multiple objectives balanced by the controller (e.g. collision avoidance and social norms) lead to contradicting robot commands. To overcome these limitations, researchers have resort to machine learning approaches [18]. Multiple approaches have proposed to use imitation learning to match human navigation patterns [19, 20, 21, 1]. The problem
of imitation learning is that the agent does not generalize beyond what has been seen in the dataset, requiring a large amount of human demonstrations to train successfully.

The dependency on large amount of training data to learn to navigate among humans can be solved within a self-exploratory loop, for example with reinforcement learning (RL) [22]. Previous approaches successfully applied deep RL to navigation and developed multi-agent collision avoidance systems focused on optimizing path efficiency while finding a feasible trajectory around a potentially large number of pedestrians [23, 24, 8]. This line of work showed that is possible to understand and generate dynamic navigation behaviors with reinforcement learning. However, these systems do not explicitly consider the presence of geometric constraints in the layout and are rather tested on spacious environments with no sensor input that reflects environmental constraints. This paper proposes a lidar-based framework that performs in indoor environments.

Previous solutions have explored how to combine motion planning and learning for navigation around dynamic obstacles. The idea is to leverage the motion planner to provide guidance to a learned controller, which adapts the plan to the current surrounding conditions as observed with the robot’s sensors. Pokle et al. [1] and Gao et al. [25] presented similar versions of this structure and showed that it outperforms a single monolithic end-to-end architecture with no planning guidance.

We build on this idea in but replace their IL mechanism to learn the reactive controllers with a RL solution in order to overcome the generalization limitation of IL to a limited coverage of expert demonstrations.

Finally, our a learning procedure trains on simple instances of a task with the goal of generalizing to task instances with the true complexity. Similar idea is behind methods on curriculum learning [26] that create a series of increasingly complex task instances to facilitate learning [27, 28, 29]. Differently, we assume that the challenges of our original navigation task cannot be learned progressively; they are the result of combining a finite set of complex patterns of layout-motion of dynamic agents (pedestrians). Therefore, we propose to learn to overcome these complex patterns separately in independent but intertwined training episodes, which will enables us to transfer eventually to a navigation task with the full combinatorial complexity.

3 Learning to Navigate in Pedestrian Environments

We propose to address the navigation problem in pedestrian environments with a combination of a sampling-base motion planner [14] and a reactive low level sensorimotor policy learned with RL and implemented as a deep neural network. We assume that our solution can make use of a 2D map approximating the layout of static elements in the environment (e.g. furniture and walls) and that it can localize on it. Each episode the robot will start at a different location and is queried to navigate to a new final goal defined as a location in the map. We query the motion planner once at the beginning of the episode for a shortest path to the goal location. This path information is then used to define the low-level sensorimotor task that we describe next, followed by a description of the simulation environments we create to train our low-level policy.

3.1 Reactive Navigation to Follow a Path

We model the problem of navigating safely among humans following a given path as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $\mathcal{M} = (S, A, O, T, R, \gamma)$. Here, $S$ is the state space; $A$ is the action space; $O$ is the observation space; $T(s'|s,a), s \in S, a \in A$, is the state transition model defining a probability over the next state $s'$ after taking action $a$ in state $s$; $R(s) \in \mathbb{R}$ is the reward at state $s$; $\gamma \in [0, 1)$ is the discount factor. We assume that the state is not directly observable and learn a policy $\pi(a|o)$ conditioned on observations $o \in O$ instead of latent states $s \in S$. The agent following the policy $\pi$ obtains an observation $o_t$ at time $t$ and performs an action $a_t$, receiving from the environment an immediate reward $r_t$ and a new observation $o_{t+1}$. Assuming the policy is parameterized by $\theta$, a policy gradient algorithm optimizes $\theta$ to maximize that the expected future return:

$$\theta^* = \arg\max_{\theta} J(\theta, \rho) = \arg\max_{\theta} E \left[ \sum_{t} \gamma^t r_t \right] \tag{1}$$

More concretely, the spaces that build the aforementioned POMDP are defined as follows. The observation space, $O$, includes elements $o = \{goal, lidar, waypoints\}$, where goal is the episodic
navigation goal, represented by the 2D coordinates in robot’s reference frame, lidar contains the 128 range measurements from a 1D LiDAR sensor in robot sensor reference frame, and waypoints contains a $n = 6$ waypoints computed by a global planner with access to a map of the environment. The action space, $A$, defines actions $a = \{ v_x, v_y, \omega \}$, where $(v_x, v_y)$ is the commanded linear velocity, and $\omega$ is the commanded angular velocity for the mobile robot.

The reward function is compose of multiple terms that encourage the policy to learn to reach the final goal in the shortest time possible following the given path but deviating from it to avoid collisions:

$$R = R_{\text{goal}} + R_{\text{timestep}} + R_{\text{collision}} + R_{\text{potential}} + R_{\text{waypoint}}$$

where $R_{\text{goal}} = +1$ is a sparse reward assigned upon reaching the goal within a distance $d_g = 0.5m$, $R_{\text{timestep}} = -0.001$ is a small negative reward with value per time step to encourage minimizing the task time, $R_{\text{collision}} = -1$ is assigned upon collision and terminates the episode, $R_{\text{potential}}$ is a dense reward assigned per time step as a function of the distance to the next desired waypoint.

We use the Soft Actor-Critic (SAC) algorithm [30, 31] to learn both a value function (critic) and a policy (actor) approximated by deep neural networks. The policy network maps from observation space to action space, providing velocity references to a low level velocity controller. The model architecture we propose for the policy network is depicted in Fig.2. The critic network presents a similar architecture with a different implementation after the shared featurizing head. We tested a number of other variants for the observation space by including other inputs to the policy network, such as observation stacking with consecutive lidar frames, robot odometry, pedestrian detection (coordinates of pedestrians) and time to collision. These variants did not offer significant performance advantages in our domains and we opted for the simplest model that solves the task. However, these other inputs are expected to become become relevant when considering explicit aspects of social interactions, such as pedestrian prediction, preferences and safety guaranties. We believe this is an interesting avenue for future research that can build on the presented model.

### 3.2 Simulated Environments

We created a multi-agent simulation environment for the general problem of navigation around pedestrians, in which a simulated robot agent can collect experience and train according to the described POMDP. We use the Interactive Gibson Simulator (iGibson) [4] that runs on top of the pyBullet physics engine [5]. Figure 3 shows a top-down view of the environments used for training and testing. These layouts were designed to capture the essential geometric properties of many indoor environments such as corridors, crossing hallways and office spaces with doorways. Many indoor spaces can be viewed as a composition of these simpler components.

Pedestrian behaviors are simulated using the Optimal Reciprocal Collision Avoidance (ORCA) model [7]. This model is often used to simulate pedestrian motion for crowd simulation and robot social navigation research [8]. In particular, ORCA guarantees that pedestrians will not collide with each other or any other objects added to a list of known obstacles. ORCA is based on a joint optimization that requires all agents to coordinate in order to guarantee no collisions. Note that while this is an useful property for driving the simulated pedestrians and provide a training environment for the robot, this assumption of a centralized controller for humans and robots, as well as the need
for full observability of pedestrians’ positions and velocities (a challenging open problem in perception), forbids using ORCA and its variants as the solution for the posed problem.

Based on a standard RL episodic training loop, a collision between the robot and any object results in termination of the episode and environment reset, which generates new random samples for the start and goal location of the robot and all pedestrians. Success is recorded if the robot gets to within 0.5 meters of the goal. A timeout is recorded if it takes the robot longer than 125 seconds to get to the goal. The personal space radius for each pedestrians was set to 10cm.

4 Experimental Evaluation and Results

We conduct a series of experiments in simulation using the environments in Fig.3. We design the experiments to study the following questions: (1) Does the proposed approach enable a robot agent to learn to navigate around pedestrians in constrained environments? (Sec. 4.1.1) (2) Does multi-layout training on a small set of canonical environments (WALLS-ABCDF) generalize to navigation on more complex layouts (WALLS-GHI) that exhibit composition of the basic geometric elements? (Sec. 4.1.1) (3) How does the proposed method compare to a planning-only approach (Sec. 4.1.2) (4) Can a policy trained in the simplified environments (WALLS-ABCDF) transfer to more complex 3D reconstructions of indoor environments (MESH)? (Sec. 4.1.3)

The evaluation and comparison is conducted using a set of eight independently-trained policy networks. Two policies are trained using a single layout ($\Pi_{WALLS-B}$, $\Pi_{WALLS-F}$). Four policies are training using the proposed multi-layout training regime, in which the agent is randomly located in one of the layouts for each episode. The multi-layout policies are $\Pi_{T1}$ trained on WALLS-ABF, $\Pi_{T2}$ trained on WALLS-ABD, $\Pi_{T3}$ trained on WALLS-ADF, and $\Pi_{T4}$ trained on WALLS-ADE. Note that $\Pi_{T1}$ has been exposed to a corridor (A), door exit (B) and crosswalk (F), whereas $\Pi_{T2}$ doesn’t see the crosswalk, $\Pi_{T3}$ doesn’t see the door exit, and $\Pi_{T4}$ doesn’t experience the crosswalk or door configurations during training. Finally, the MESH environments are used to train the policies $\Pi_{MESH-MARKET}$ and $\Pi_{MESH-HOME}$, separately using the supermarket and apartment 3D reconstructions, respectively.

For each experimental session, we train the policy for 200,000 training steps using 4 parallel simulation environments. We use the SAC implementation available on TFagents [32] and the available software integration with iGibson [4] and openAI Gym [33].

**Metrics:** The following metrics are used to evaluate the objective performance of the system. **Success rate (S):** percentage (%) of episodes where the robot reaches its goal without colliding or
Table 1: Single-layout training

| Test | # ped. | S  | C  | PC | OC | TO | PSO | S  | C  | PC | OC | TO | PSO |
|------|--------|----|----|----|----|----|-----|----|----|----|----|----|-----|
| WALLS-I | 4 | 96.00 | 4.00 | 4.00 | 0.00 | 0.00 | 2.07 | 86.00 | 14.00 | 8.00 | 6.00 | 0.00 | 1.95 |
| WALLS-H | 6 | 70.00 | 30.00 | 2.00 | 28.00 | 0.00 | 0.66 | 80.00 | 20.00 | 2.00 | 18.00 | 0.00 | 1.90 |
| WALLS-G | 3 | 92.00 | 8.00 | 6.00 | 2.00 | 0.00 | 1.09 | 68.00 | 32.00 | 12.00 | 1.96 | 0.00 | 2.80 |
| WALLS-F | 6 | 16.00 | 84.00 | 2.00 | 80.00 | 0.00 | 0.00 | 1.09 | 32.00 | 16.00 | 1.96 | 0.00 | 2.80 |
| WALLS-R | 4 | 56.00 | 46.00 | 4.00 | 5.00 | 1.00 | 1.22 | 92.00 | 8.00 | 4.00 | 4.00 | 0.00 | 1.72 |
| WALLS-RO | 4 | 56.00 | 46.00 | 4.00 | 5.00 | 1.00 | 1.22 | 92.00 | 8.00 | 4.00 | 4.00 | 0.00 | 1.72 |
| WALLS-OS | 4 | 56.00 | 46.00 | 4.00 | 5.00 | 1.00 | 1.22 | 92.00 | 8.00 | 4.00 | 4.00 | 0.00 | 1.72 |

Table 2: Evaluation results using multi-layout training

| Test | # ped. | S  | C  | PC | OC | TO | PSO | S  | C  | PC | OC | TO | PSO |
|------|--------|----|----|----|----|----|-----|----|----|----|----|----|-----|
| WALLS-I | 4 | 96.00 | 4.00 | 4.00 | 0.00 | 0.00 | 2.07 | 86.00 | 14.00 | 8.00 | 6.00 | 0.00 | 1.95 |
| WALLS-H | 6 | 70.00 | 30.00 | 2.00 | 28.00 | 0.00 | 0.66 | 80.00 | 20.00 | 2.00 | 18.00 | 0.00 | 1.90 |
| WALLS-G | 3 | 92.00 | 8.00 | 6.00 | 2.00 | 0.00 | 1.09 | 68.00 | 32.00 | 12.00 | 1.96 | 0.00 | 2.80 |
| WALLS-F | 6 | 16.00 | 84.00 | 2.00 | 80.00 | 0.00 | 0.00 | 1.09 | 32.00 | 16.00 | 1.96 | 0.00 | 2.80 |
| WALLS-R | 4 | 56.00 | 46.00 | 4.00 | 5.00 | 1.00 | 1.22 | 92.00 | 8.00 | 4.00 | 4.00 | 0.00 | 1.72 |
| WALLS-RO | 4 | 56.00 | 46.00 | 4.00 | 5.00 | 1.00 | 1.22 | 92.00 | 8.00 | 4.00 | 4.00 | 0.00 | 1.72 |
| WALLS-OS | 4 | 56.00 | 46.00 | 4.00 | 5.00 | 1.00 | 1.22 | 92.00 | 8.00 | 4.00 | 4.00 | 0.00 | 1.72 |

timing out: Collision rate (C): percentage (%) of episodes during which the robot collides with either a pedestrian or static object such as a wall; Collision with pedestrians (PC): percentage of episodes involving collisions with pedestrians; Collision with obstacles (OC): percentage (%) of episodes involving collisions with static obstacles such as walls; Timeout rate (TO): percentage (%) of episodes where the robot runs out of time before reaching its goal; Personal Space Overlap (PSO): distance (cm) between the robot and pedestrians for which the robot is closest than the specified personal space threshold, aggregated across all pedestrians and averaged over length of the episode.

4.1 Experiments

4.1.1 Single- and Multi-Layout WALLS- Training

The single-layout and multi-layout policies are deployed in unseen layouts WALLS-G, WALLS-H and WALLS-I. Layout WALLS-H represents the generalization target with most compositional elements on it (corridors, doors, obstacles, intersections). The results are summarized in Tables 1 and 2, respectively. All presented Tables contain the average results over three experimental runs of 100 episodes each. Success rate below 90% are highlighted in orange.

First, we note that policies from single-layout training, which achieve $S > 90\%$ when tested in the same training environment, have a performance drop when tested on more complex layouts. Both $\Pi_{WALLS-B}$ and $\Pi_{WALLS-F}$ had lower performance in layout WALLS-H. This indicates the need for a larger body of experience during training, considering other arrangements of constraints and the patterns of pedestrian motion that arise in these configurations. Unlike single-layout training, using multiple canonical environments for training policy $\Pi_{T1}$ resulted in consistent generalization performance to all three test environments with average success rate of 93.5% (WALLS-I, 96% WALLS-H and 96% WALLS-G) (Table 2).

However, not all multi-layout policies generalize equally well to the compositional test environments. While $\Pi_{T1}$ and $\Pi_{T3}$ produce consistent transfer performance (on WALLS-G, WALLS-H, and WALLS-I), $\Pi_{T2}$ and $\Pi_{T4}$, which haven’t experience the crosswalk or door configurations, have significant performance drop in WALLS-H which challenges the agent the most. $\Pi_{T4}$ also has a performance cost on WALLS-I.

This results supports the importance of relevant geometric configurations in enabling transfer to a compositional domain, as a key element on learning beyond simply large collection of varied experience, and give a positive answer to the posed questions (1) and (2).

The multi-layout approach results in consistent training curves as shown in Figure 4 for $\Pi_{T1}$ training with 4 random seeds. The resulting behavior of the policy can be observed in a video accompanying this paper linked here. Figure 5 shows some illustrative examples of navigation episodes produced by $\Pi_{T1}$. The first two cases test the agent in an intersection pattern. The robot lower speeds while pedestrians cross in front and later advances with increased speed. The policy allows linear velocities in the range $(-0.2,1)m/\sec$, which enable stopping and backwards motions to accommodate pedestrians. The third example shows an episode of $\Pi_{T1}$ in WALLS-H with a number of pedestrians present in the corridor between doors.
4.1.2 Comparison with a planning-based approach

Planning approaches are suitable for application in our problem set up as they can also perform in arbitrary layouts (maps) without increasing complexity as a function of the layout geometry or size in an intractable manner. We compare the performance of the learned policy $\Pi_{T_1}$ with the planner available with the ROS Navigation framework (ROS-NAV-STACK). The planner in the ROS-NAV-STACK is a layered costmap method [34] that also uses two components: a global planner using Dijkstra’s algorithm and a local controller that uses the dynamic window approach [35]. The ROS navigation parameters were tuned to optimize the performance of the planner in the WALLS layouts. The tests are conducted on WALLS-I with an increasing number of pedestrians in comparison with $\Pi_{T_1}$. Results are summarized on Table 3 and the success rate as a function of the number of pedestrians is shown in Fig. 6a. The planner resulted in lower success rate across all tests caused mainly due to timeouts. This robot-freezing problem [16] becomes more frequent as the pedestrian density increases and the planner fails to find feasible plans in the dynamic scene. While the combination of planning and learning in $\Pi_{T_1}$ has a performance that also degrades with the number of pedestrians, the failure cases are due to collisions and not timeouts. It is worth to highlight that we have conducted the tests we no enforced safety stop in order to obtain a bound for the performance of the policy itself. However, real deployment of this system can incorporate momentary safety stops based on sensor readings for safe deployment.

The comparison with a motion planner (no learning) is a proxy to quantify the benefit added by the RL components (Fig. 6a). The planner is less successful at accomplishing the goal because it suffers frequently from the robot freezing problem (large timeouts (TO) in Table 3), which means that it continuously fails to find a feasible path and keeps re-planning while pedestrians move. In contrast, our method resulted in no timeouts. Note that the large timeouts for the planner cause it to exhibit less collisions than our method in some experiments, but this result does not indicate superior performance, it is instead a consequence of the robot not moving for extended periods. This results answers the third posed question with a quantitative assessments.

4.1.3 Cross-domain generalization with WALLS and MESH environments

We investigate if an agent trained in this simplified representation WALLS- is capable of navigating around pedestrians in realistic 3D scanned scenes (MESH), such as a supermarket (MESH-MARKET) and an apartment (MESH-HOME), illustrated in Fig. 6b and shown in the accompa-
Figure 5: Example test episodes showing the trajectory of the robot and pedestrians for $\Pi_{T1}$.

Figure 6: (a) Performance of T1 and ROS-NAV tested on WALLS-I with increasing number of pedestrians. (b) Robot deployed in MESH-HOME. The red signals in front of the robot show the waypoints computed by the planner. Six pedestrians are present in the scene and are conducted by the ORCA policy. The robot must navigate around pedestrians in the corridor and door to enter the room at the bottom left.

nying video. First we measure the performance of training and testing on MESH-MARKET and MESH-HOME (i.e. MESH to MESH), and compare to that of $\Pi_{T1}$ in the mesh environments (i.e. multi-layout training on WALLS to MESH), and viceversa. The results are presented on Table 4.

The average success rate obtained by training and testing on the same mesh results in 79.5% (MESH-MARKET =75%; MESH-HOME=84%), superior to the average transfer from WALLS to MESH of 71.5% ($\Pi_{T1}$ in MESH-MARKET =77%; $\Pi_{T1}$ in MESH-HOME=66%). This is consistent with the expected performance of training in the target environment vs. policy transfer. However, transfer from MESH training to any other environments exhibits a significant performance drop compared to $\Pi_{T1}$. This indicates that the MESH performance is the product of overfitting to the environment, while WALLS training performs more consistently across domains. This result supports both the layout compositional argument and the approach of training on abstracted blocks-like worlds and gives a positive answer to the fourth posed question.

5 Conclusion

We present an approach based on reinforcement learning to learn robot policies for navigation around pedestrians in constrained environments. The proposed model receives guidance from a motion planner that provides waypoints to follow a globally planned trajectory, whereas the reinforcement component handles the local interactions needed for on-line adaptation to pedestrians. The learned policy naturally exhibits interesting behaviors such as slowing down, maneuvering around pedestrians and going backwards to accommodate different common social interactions that emerge as a consequence of the constrained layouts.

The analysis of the use of a compositionality principle for the proposed multi-layout training regime showed (1) the ability to train on simplified model abstractions (walls worlds with straight walls and simple layouts such as corridors and crosswalks) and deploy on a more complex unseen environment with composition of the basic layouts; (2) that policies trained on relevant geometric configurations enable better generalization than those trained on layouts that don’t exhibit the geometry and inherent pedestrians interactions present in the target layout; and (3) policies trained on walls-worlds are able to generalize to unseen 3D reconstructions from real environments (such as an apartment and a supermarket). These results offer a practical advantage by removing the need to train on target environments with high reconstruction fidelity.
References

[1] A. Pokle, R. Martín-Martín, P. Goebel, V. Chow, H. M. Ewald, J. Yang, Z. Wang, A. Sadeghian, D. Sadigh, S. Savarese, et al. Deep local trajectory replanning and control for robot navigation. In 2019 International Conference on Robotics and Automation (ICRA), pages 5815–5822. IEEE, 2019.

[2] J. Choi, C. Dance, J. eun Kim, K. sik Park, J. Han, J. Seo, and M. Kim. Fast adaptation of deep reinforcement learning-based navigation. In 2020 International Conference on Robotics and Automation (ICRA). IEEE, 2020.

[3] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson env: real-world perception for embodied agents. In Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on. IEEE, 2018.

[4] F. Xia, C. E. Li, W. Shen, P. Kasimbeg, R. Martin-Martin, A. Toshev, and S. Savarese. Interactive gibson: Benchmark for interactive navigation in cluttered environments. RA-L, 2020.

[5] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[6] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. Physical review E, 51(5):4282, 1995.

[7] 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.

[8] C. Chen, Y. Liu, S. Kreiss, and A. Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In 2019 International Conference on Robotics and Automation (ICRA), pages 6015–6022. IEEE, 2019.

[9] Y. Che, A. M. Okamura, and D. Sadigh. Efficient and trustworthy social navigation via explicit and implicit robot–human communication. IEEE Transactions on Robotics, 36(3):692–707, 2020.

[10] H. Khambhaita and R. Alami. Viewing robot navigation in human environment as a cooperative activity. In Robotics Research, pages 285–300. 2020.

[11] V. V. Unhelkar, C. Pérez-D’Arpino, L. Stirling, and J. A. Shah. Human-robot co-navigation using anticipatory indicators of human walking motion. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 6183–6190. IEEE, 2015.

[12] Y. Chen, C. Liu, B. E. Shi, and M. Liu. Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. IEEE Robotics and Automation Letters, 5(2):2754–2761, 2020.

[13] C. Mavrogiannis, A. M. Hutchinson, J. Macdonald, P. Alves-Oliveira, and R. A. Knepper. Effects of distinct robot navigation strategies on human behavior in a crowded environment. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 421–430. IEEE, 2019.

[14] S. M. LaValle. Planning algorithms. Cambridge university press, 2006.

[15] V. Kunchev, L. Jain, V. Ivancevic, and A. Finn. Path planning and obstacle avoidance for autonomous mobile robots: A review. In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pages 537–544. Springer, 2006.

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

[17] P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. The International Journal of Robotics Research, 34(3):335–356, 2015.
[18] M. W. Otte. A survey of machine learning approaches to robotic path-planning. *Cs. Colorado. Edu*, 2015.

[19] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[20] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746, 2006.

[21] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

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

[23] 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.

[24] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[25] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 185–194. PMLR, 13–15 Nov 2017.

[26] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML ’09*, 2009.

[27] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2019.

[28] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320, 2017.

[29] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495, 2017.

[30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1861–1870, 2018.

[31] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[32] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gongina, N. Wu, C. Harris, et al. Tf-agents: A library for reinforcement learning in tensorflow, 2018.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[34] 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.

[35] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
Supplementary Materials

1 Simulation Environment Implementation

Simulation: The robot collects experience in the simulated environment while being tasked to navigate from a set of initial to target positions. A policy network is learned using the off-policy Soft Actor Critic reinforcement algorithm (SAC) in continuous action space of robot velocity. The policy network receives guidance from a motion planner that provides waypoints to follow a globally planned trajectory, whereas the reinforcement component handles the balance with the local interactions. The RL agent is rewarded with a balance between following the waypoints but also deviating to accommodate for the presence of pedestrians. This framework results in policies that can adapt to arbitrary environments by generating a navigation motion plan, and leaves the local reactive interaction with nearby pedestrians to be learned by reinforcement. Robot perception is based on a lidar sensor, which raw data resulted in enough information for obstacles and pedestrians avoidance in the tested environments.

The simulation environment is built based on iGibson and the PyBullet physics engine. The learning agent is a differential drive mobile robot. These non-holonomic constraints are included in the simulation. The policy network outputs the desired values of linear and angular velocity for the robot \((V, w)\), which are passed to a low level velocity controller in wheel speed. This software architecture in simulation follows the architecture on the real robot in terms of input commands.

Robot Behavior: The policy network receives as input a set of observations \(o = \{\text{goal}, \text{lidar}, \text{waypoints}\}\), where \(\text{goal}\) is the episodic navigation goal (2D polar coordinates of the goal in robot frame), \(\text{lidar}\) contains 1D LiDAR sensor measurements in robot frame, and \(\text{waypoints}\) contains a number of reference waypoints computed by a global planner with access to a map of the environment. The waypoints are selected from the collision-free shortest path between the start and goal points of the episode. The reference waypoints are updated as the robot makes progress along the desired path. A waypoint resolution is used to define the distance between consecutive waypoints, and a waypoint tolerance value is used to define the minimum proximity from the robot to a waypoint to receive the waypoint reward and update the next desired waypoint. Parameters for the robot behavior are summarized on Table 5.

Simulated Pedestrians: The simulated pedestrians follow the ORCA social forces model. The used parameters are summarized on Table 5. The ORCA model jointly optimizes the trajectories of all agents (pedestrians) to collaboratively avoid collisions. This property is guaranteed if all agents on the scene participate in the optimization. The ORCA model that drives all pedestrians has access to the state of the robot and considers it a participant agent. This produces the behaviors of the pedestrians to be collaborative with the robot under the expectation that the robot will be reciprocally collaborative in avoiding collisions. However, the robot follows a policy different from ORCA (either the learned policy or the ROS-NAV-STACK planner). This simulation strategy results in an approximation to mutual collaboration between the pedestrians and the robot. While ORCA offers an efficient model to simulate multiple pedestrians, it is limited in the types of trajectories

| Observations and Robot Behavior |  |
|---------------------------------|--|
| Angular Velocity \([-0.5, 0.5]\) rad/sec |  |
| Linear Velocity \([-0.2, 1]\) m/sec |  |
| Number of Reference Waypoints | 6 |
| Waypoint Resolution | 1.00m |
| Waypoint Tolerance | 0.50m |

| Pedestrians Behavior (ORCA) |  |
|-------------------------------|--|
| Personal Space | 10cm |
| Pedestrian Radius | 30cm |
| Prefered Velocity | 2m/sec |
| Maximum Speed | 2m/sec |
| Robot Visible to Pedestrians | TRUE |
the pedestrians can follow and does not adapt well to trajectory that require turns, concave obstacles or trajectories with lack of visibility to the goal. To address this limitation, each layout includes a definition of allowed initial and end areas (2D locations and allowed radius shown in Fig. 7 as shaded areas) to require underlying pedestrian trajectories that are feasible for the ORCA model to generate. For each episode, locations are randomly sampled from this allowed set. Pedestrians are represented in the simulator interchangeable as either a cylinder shape or a static mesh. We observe that improvements on the behavioral simulation of pedestrians at the macro level would potentially unlock greater adaptability of the learned behaviors to the real world. For example, in the simulation of MESH- worlds, which follows the same principles as the WALLS- environments, the simulation of pedestrians becomes more challenging as the specific context becomes relevant for sampling the start and end points for the ORCA-driven pedestrians: (e.g. walking from the kitchen to the bedroom). We expect that improvement on pedestrian simulation alone will result on better robot performance overall, without additional modifications to the method, as it would produce more realistic pedestrians trajectories. We leave these improvements as subject of future work.

**Multi-Layout Training:** We train a set of policies on selected combinations of simple layouts WALLS- These combinations are denominated \{T1, T2, T3, T4\}. For each Ti, the robot is randomly located in one of the included WALLS- layouts at the beginning of each training episode.

**Simulation Design:** Using the proposed approach and generally on simulation-based robot learning, the engineering effort centers on designing a simulation environment that properly exhibits the dynamics and properties of the target domain. In robotics, this effort differs from planning techniques, in which the effort is rather centered in explicitly defining algorithmic solutions within some problem properties. Both approaches require significant design and engineering, while the difference lies in specifying the environment (the problem) or the algorithm (an explicit solution).

![Figure 7: Locations on WALLS- domains. Each layout has dimensions 20mx20m.](image)

### 2 Hyperparameters

Table 6 reports the hyper-parameters used for the reinforcement learning algorithm SAC. All the reported policies were learned using the same set of hyper-parameters.
Table 6: SAC Hyperparameters

| SAC Hyperparameters                                      |   |
|----------------------------------------------------------|---|
| Hardware Configuration                                   | NVIDIA® V100 |
| Optimizer                                               | Adam |
| Discount Ratio Gamma                                     | 0.99 |
| Critic Learning Rate                                     | 0.0003 |
| Actor Learning Rate                                      | 0.0003 |
| Alpha Learning Rate                                      | 0.0003 |
| Maximum Episode Length (Step)                            | 500 |
| Replay Buffer Capacity                                   | 150000 |
| Action Time Step                                         | 0.25 sec |

We tuned the parameters of the ROS-NAV-STACK planner to optimize its performance in the tested environments. Adjusting these values reduces planner timeouts and increases likelihood of finding feasible solutions in the footprint of the tested layouts. Table 7 reports the used parameters.

Table 7: Optimized parameter for ROS-NAV planner

| DWA Trajectory Scoring Parameters                        |   |
|----------------------------------------------------------|---|
| path_distance_bias                                       | 48 |
| goal_distance_bias                                       | 24.00 |
| occdist_scale                                            | 0.01 |

| Global and local costmaps                                |   |
|----------------------------------------------------------|---|
| footprint_padding                                        | 0.05 |
| inflation_layer: cost_scaling_factor                     | 2.58 |
| inflation_radius                                         | 2.50 |

| move_base Parameters                                      |   |
|----------------------------------------------------------|---|
| planner_frequency                                        | 0.5 Hz |
| controller_frequency                                     | 15 Hz |
| planner_patience                                         | 200 sec |
| controller_patience                                      | 15 sec |
| recovery_behavior_enabled                                | FALSE |
| clearing_rotation_allowed                                | FALSE |
| max_planning_retries                                     | -1 |