

Deep Local Trajectory Replanning and Control for Robot Navigation

Ashwini Pokle¹, Roberto Martín-Martín¹, Patrick Goebel¹, Vincent Chow¹, Hans M. Ewald¹, Junwei Yang¹, Zhenkai Wang¹, Amir Sadeghian¹, Dorsa Sadigh¹, Silvio Savarese¹, Marynel Vázquez²

Abstract—We present a navigation system that combines ideas from hierarchical planning and machine learning. The system uses a traditional global planner to compute optimal paths towards a goal, and a deep local trajectory planner and velocity controller to compute motion commands. The latter components of the system adjust the behavior of the robot through attention mechanisms such that it moves towards the goal, avoids obstacles, and respects the space of nearby pedestrians. Both the structure of the proposed deep models and the use of attention mechanisms make the system’s execution interpretable. Our simulation experiments suggest that the proposed architecture outperforms baselines that try to map global plan information and sensor data directly to velocity commands. In comparison to a hand-designed traditional navigation system, the proposed approach showed more consistent performance.

I. INTRODUCTION

Autonomous robot navigation in known environments encompasses two main problems: 1) finding a safe path for a robot to reach a desired goal location, and 2) following the path while adapting to environmental conditions [1]. While *global planners* can efficiently find optimal motion paths [2], translating these paths into robot commands – which is traditionally the job of a *reactive controller* – can be challenging. Reactive controllers not only need to take into account kinodynamic constraints [3], but also consider plan execution and adaptation to the environment, e.g., to avoid obstacles [4] and respect social conventions [5] (Fig. 1).

At the core of classical approaches to reactive control is a hand-designed objective function that must balance navigation criteria to output motion commands [3, 6, 7]. While successful in simple situations, these approaches can be difficult to tune for dynamic human environments. One reason is that relevant criteria, like social norms, are hard to define mathematically. Even when models exist, complex interactions may emerge between model parameters and the resulting navigation behavior [7]. Some navigation criteria may even be contradictory at times, e.g., reaching a goal in a crowded environment without violating personal space.

In this work, we combine ideas from machine learning [8, 9, 10, 11, 12] and hierarchical planning [13] to improve reactive robot control. Our approach does not require hand-specifying all the parameters of the reactive controller; instead, most parameters are optimized based on example navigation data through imitation learning [14, 15, 8]. Similar to [11], we assume that localization information is available during robot operation, and focus on studying mechanisms to combine high-level planning and learning for low-level motion control. In contrast to [11], though, we use learning not only for controlling robot velocities, but also for predicting a



Fig. 1: Navigation scenario. The (solid) robot needs to reach the back of the foyer. The global planner provides an optimal solution (light green) based on a static environment map. Our navigation approach re-plans locally to adapt to the dynamic environment and control the robot towards the goal (translucent robots).

local motion plan, which guides the velocities output by our approach. Our main insight is that by adding structure to the learning component of our system we can guide the learning process to find an appropriate complex mapping from the high-level plan to motion commands, without incurring in additional annotation costs. As suggested by our experimental evaluation, the added structure can improve overall navigation behavior in comparison to mapping a goal or a global plan directly to commands [10, 11]. Predicting a local plan also facilitates system interpretability upon execution.

The proposed local trajectory planner and velocity controller of our approach adjust the behavior of the robot through simple, deep attention mechanisms. These mechanisms enable the robot to dynamically focus on different tasks: obstacle avoidance, social interaction, or following navigation plans. As a result, the robot is able to account for changing elements in the environment. It behaves in a manner that resembles complex rules that govern human use of space [16, 17].

In summary, this work has several main contributions. First, we introduce a new system for autonomous navigation which combines planning and learning. The system’s learning component predicts a local plan and motion commands. Second, we propose an attention mechanism for multimodal data fusion. Finally, we conduct controlled experiments on a simulated platform to evaluate the proposed system.

II. RELATED WORK

Autonomous navigation has long been studied within robotics. Early navigation methods focused on path planning [18], while more recent approaches tend to leverage machine learning to make navigation systems less brittle to new environmental conditions [19, 20]. In some cases, machine learning is used to create systems that improve as they explore more of the environment, e.g., with reinforcement learning [21] or via knowledge transfer in lifelong learning

¹ Stanford University. ² Yale University.

settings [22]. In other cases, machine learning is used to model preferences for the navigation task with the help of a teacher [23]. For instance, prior approaches have used structured prediction [24] or inverse reinforcement learning [25] to find cost functions that encode preferences for navigating through parts of an environment. Closer to our work, some methods have focused on directly learning motion policies with imitation learning, e.g., [14, 15, 8]. As discussed in [26], directly mapping states to actions can be more efficient than learning a cost function for imitation. Due to limited space, we encourage readers interested in more details of these different approaches to refer to [18, 20, 21, 23, 27].

Similar to [8, 9, 10, 12], we use deep learning [28] to parameterize a motion policy. This approach allows us to forgo hand-engineered features for sensor data. In contrast to most of these efforts, though, we do not aim to solve the whole navigation problem with a single function trained in an end-to-end fashion. Instead, we leverage planning in conjunction with deep learning for autonomous navigation.

Inspired by IntentionNet [11], we use a global planner to solve for the general direction that a robot should follow to reach a desired destination in a known environment, as well as use deep learning for motion control. But different to [11], (1) our approach explicitly considers the presence of people nearby the robot, (2) processes raw lidar measurements instead of RGB images, (3) represents global plans via trajectories, and (4) enforces additional structure on the learning component of the navigation system. Our rationale behind these considerations are as follows. First, providing information about people’s motion directly to our navigation system facilitates interactions in human environments. Second, providing raw lidar measurements to our system reduces the complexity of the input space and facilitates system development through simulation in comparison to using raw images. Lidar can also help with obstacle avoidance, as in [10], because it measures depth directly and typically has a wider field of view than cameras. Third, representing global plans via trajectories, instead of rendering them on maps, reduces even further the dimensionality of the input space. Fourth, adding structure to the learning component of our navigation system offers an opportunity to add supervision and facilitate interpretability. Our approach does not aim to build an environment map as the robot navigates [29], but assumes that a map is given for high-level, global planning.

The study of how people use and share space, or proxemics [16], is relevant for social robot navigation, e.g., see [30] for a recent review. Generally, most methods for social navigation explicitly model interactions among agents or social conventions [5, 31, 7, 32, 33, 34, 35, 36, 37, 38, 39]. Other methods, like ours, implicitly model these aspects and let navigation strategies emerge through imitation [11, 40, 41].

We leverage simulations to facilitate training motion policies and conduct system evaluations. Similar to work by Tai et al. [41] and Müller et al. [42], we avoid providing complex sensory input, like images, to our method. Instead, our deep model takes as input lidar and information derived from complex signals, like people’s position nearby the robot.

III. APPROACH

Building on a long tradition in robotics [43, 44, 45, 46, 11], we use a hierarchical approach for locomotion. Our particular hierarchy is composed of three levels: Global Planning, Local Planning, and Velocity Control. As shown in Fig. 2, the *Global Planner* takes as input a 2D map of the environment with static obstacles, e.g., walls, the current location of the robot, and the coordinates of a desired destination (or goal) in the map. Based on these inputs, the global planner computes a path in the map for the robot to reach the desired goal location. The *Local Planner* then focuses on a shorter time horizon: what are the immediate navigation steps that the robot should take to follow the global plan in an acceptable manner given observed environmental conditions? These conditions include dynamic elements, such as people, as well as static obstacles sensed in the vicinity of the robot. Finally, the *Velocity Controller* commands motions to the robot based on the navigation plans.

Explicitly or manually defining socially appropriate navigation behavior is hard. Human behavior is malleable and social norms often change from one social context to another [16, 17]. Thus, we resort to imitation learning to model appropriate behavior. Our main assumption in this work is that expert demonstrations are available to learn what constitutes good navigation patterns for a robot.

We implement the Local Planner and the Velocity Controller of our navigation approach as Neural Networks and optimize their parameters based on expert motion data. While it would have been possible to also implement the Global Planner as a differentiable function [47] and optimize for the whole system in a complete end-to-end fashion, we opt for combining machine learning with traditional planning in this work. This combination aims to take advantage of the benefits of both approaches, while keeping the implementation of our system practical. We use planning because it is fast and reliable for computing obstacle-free routes in static environments, and we use deep networks because they have a great capacity for reasoning about raw sensor data and modeling complex phenomena like social behavior [48, 49]. Note that the learning component of our hierarchical navigation approach has more structure than similar prior work [10, 11]. We use a differentiable Local Planner to provide additional supervision to the learning component of our system and facilitate interpreting its execution.

A. Global Planning: Finding a Route to the Goal

As a first step when navigation starts, our system computes a collision free path from the initial location of the robot x_0 to the desired goal location g in a static map of the environment. The path is computed with Dijkstra’s shortest path algorithm [50] and saved as a reference during navigation.

The Global Planner of our system provides a down-sampled version of the full global plan to the Local Planner. Down-sampling preserves the desired motion direction and highlights the relevant part of the global plan for successive processing. Let the output of the Global Planner be a down-sampled path $G = \{(x_i, y_i) \mid 1 \leq i \leq 10\}$, with each (x_i, y_i)

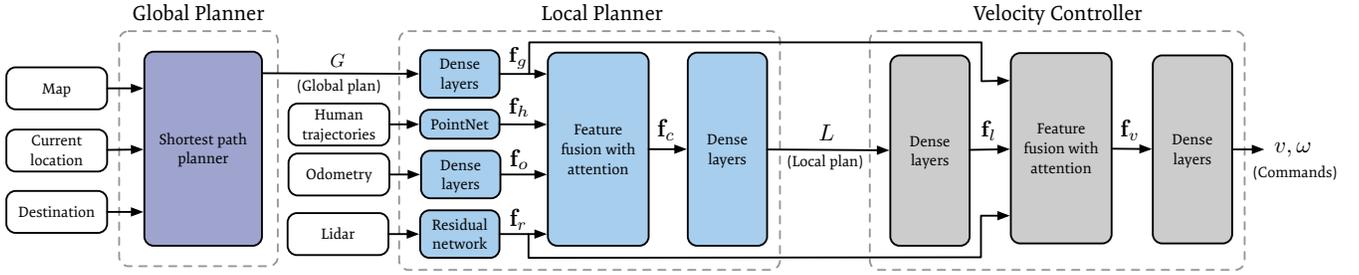


Fig. 2: Proposed approach. The white boxes denote inputs to our system. The purple box, blue boxes, and gray boxes belong to the global planner, the local planner, and the velocity controller, respectively. See the text for more details.

a waypoint belonging to the original full plan. We enforce that the first waypoint (x_1, y_1) in G is the closest waypoint in the full plan to the current location of the robot, and that the remaining waypoints are at least 0.5 meters from the previous one in G . If not enough waypoints that satisfy these constraints remain in the full plan, then the goal is used instead.

B. Local Planning: Predicting Navigation Subgoals

Let the inputs to the local planner be the down-sampled global plan G , the lidar range measurements $R = \{r_i \mid 1 \leq i \leq M\}$, the robot's linear and angular velocities $O = (\hat{v}, \hat{\omega})$ from odometry, and a set of observed human trajectories $H = \{\mathcal{T}_i \mid 1 \leq i \leq P\}$ with $\mathcal{T}_i = \{(x_j, y_j) \mid t - k \leq j \leq t\}$ the 2D coordinates of person i for the last k time-steps relative to the current pose of the robot. Features for these inputs are computed by:

Global Plan Features. The model flattens G into $g \in \mathbb{R}^{20}$, and projects g into a higher dimensional space through two fully connected ReLU layers with 32 and 512 units, respectively. The result is a feature vector $\mathbf{f}_g \in \mathbb{R}^{512}$.

Lidar Features. The model uses the residual network from [10] to compute the lidar features $\mathbf{f}_r \in \mathbb{R}^{512}$. The residual network is composed of 5 convolutional layers and two residual connections. See Fig. 2 in [10] for more details.

Odometry Features. The model projects O into a higher dimensional space through two fully connected ReLU layers, similar to the Global Plan features. The result is a feature vector $\mathbf{f}_o \in \mathbb{R}^{512}$.

Human Trajectory Features. In the spirit of [51], our model uses a PointNet network [52] to compute features for a set of past human trajectories. This choice frees us from needing to specify an order in which human coordinates should be input to our model. More specifically, the local planner organizes observed human trajectories H into a matrix $H' \in \mathbb{R}^{P \times 2k}$, where each row corresponds to the data of one person. The rows are independently projected into a higher dimensional space of 1024 dimensions through a three layer perceptron with ReLU activations. The resulting matrix in $\mathbb{R}^{P \times 1024}$ is then applied a max pooling operation and flattened to convert it into a vector in \mathbb{R}^{1024} that encodes information about the motion of the people nearby the robot. This data is finally transformed by a fully connected layer with ReLU activation into the feature vector $\mathbf{f}_h \in \mathbb{R}^{512}$, which has the same dimensionality as the other features.

In initial experiments, we considered combining the above features through concatenation [36, 11] or outer product operations [53, 54]. We found, however, that it was hard for the robot to pay attention to multiple sources of information with concatenation. For example, the robot would follow the global plan and ignore obstacles observed through its lidar in some cases. We also found that, as in [53], the outer product led to very high-dimensional features that made training prohibitively expensive or prone to over-fitting.

Fusion of Input Features. Inspired by prior work in text translation [55], we propose to fuse input features with a simple concatenation-based attention mechanism:

$$\mathbf{f}_c = [a_g \mathbf{f}_g \ a_r \mathbf{f}_r \ a_h \mathbf{f}_h \ a_o \mathbf{f}_o] \quad (1)$$

with $\mathbf{a} = [a_g \ a_r \ a_h \ a_o]^T$ coefficients that reflect the relative importance of the features. We compute the attention coefficients \mathbf{a} by first concatenating the input features and transforming them into a higher dimensional space, $\mathbf{u} = \max(0, W_1 \text{concat}(\mathbf{f}_g, \mathbf{f}_r, \mathbf{f}_h, \mathbf{f}_o))$ with $W_1 \in \mathbb{R}^{128 \times 2048}$ a parameter matrix. Then, the desired coefficients are given by $\mathbf{a} = \text{softmax}(W_2 \mathbf{u})$ with $W_2 \in \mathbb{R}^{4 \times 128}$ trainable parameters.

The Local Planner finally generates a plan L by transforming the combined features \mathbf{f}_c through 3 fully connected layers with 512, 256 and 64 units and ReLU activations, and one final linear transformation. The plan is an ordered set of poses $L = \{(x_i, y_i, \cos(\theta_i), \sin(\theta_i)) \mid 1 \leq i \leq 5\}$ sampled at 1Hz that represent a time-dependent trajectory that the robot could follow to adapt to the dynamic environment within the next 5 seconds. L guides the robot in the direction of the global plan, and encodes temporal information to enable variations in speed as necessary while navigating.

C. Velocity Control: Predicting Low-Level Commands

We provide the Velocity Controller three inputs: L , the lidar features \mathbf{f}_r , and the global plan features \mathbf{f}_g computed during local planning. While L might suffice in cases where the prediction of the local plan is appropriate for the current environmental conditions, there might be cases when L is problematic, e.g., bringing the robot close to collision due to a prediction error, or getting the robot stuck due to local minima. In these cases, range information and the global plan can help the robot avoid obstacles in close proximity and navigate towards the goal.

The Velocity Controller first computes features for the local plan L by projecting it to a higher dimensional space

through two fully connected ReLU layers with 32 and 512 units, respectively. The result are local plan features $\mathbf{f}_l \in \mathbb{R}^{512}$ with the same dimensionality of features \mathbf{f}_r and \mathbf{f}_g .

The Controller combines features with attention, as in eq. (1), except that only three attention coefficients are needed in this case. Let these coefficients be denoted by b_l, b_r, b_g . Then, the combined features $\mathbf{f}_v = [b_l \mathbf{f}_l \ b_r \mathbf{f}_r \ b_g \mathbf{f}_g]$.

Finally, the Velocity Controller projects the combined features \mathbf{f}_v to a 2D space through 2 fully connected ReLU layers with 512 and 128 units, followed by a fully connected ReLU layer and a linear activation function. The two outputs are the linear velocity (v) and angular velocity (ω) that command the robot to move as desired, as shown in Fig. 2.

D. Learning a Local Planner and Velocity Controller

The joint network of the Local Planner and the Velocity Controller can be seen as a policy that maps states into actions based on examples from an expert. The state is composed of the inputs to the Local Planner; the actions are the low-level, velocity commands that are output by our navigation system. Under this formulation, our objective is to compute a policy $\hat{\pi}$ that minimizes the expected loss ℓ with respect to the expert policy π^* under the distribution of states $s \sim d_{\pi^*}$ induced by the expert:

$$\hat{\pi} = \arg \min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}} [\ell(s, \pi)] \quad (2)$$

This particular view of the imitation problem follows close related work [10, 11] and is typically known as Behavioral Cloning [14, 8, 40].¹ Our main assumption is that expert motion trajectories are available for supervised learning.

We learn the parameters of the Neural Networks of our system through back-propagation with three Adam optimization procedures [59] aimed at providing an easier path to learning, similar to curriculum learning [60] or shaping [61]. Unless otherwise noted, we use L2 loss with a batch size of 300 samples. We pick the best parameters after 100 epochs.

Training Procedure. First, we compute lidar features \mathbf{f}_r with the help of the navigation model proposed by Pfeiffer et al. [10]. We train their model by minimizing the loss on robot velocity. We then load the pre-trained weights for \mathbf{f}_r into the Residual Network of our Local Planner (Fig. 2) and optimize for the rest of this component. Our insight is that we can use expert motion trajectories to supervise the Local Planner without collecting additional data. Our ground truth for the local plan L at a given time-step t is the pose of the robot at $t+1s, t+2s$, etc., in the future, which is readily available after collecting expert trajectories. Finally, we load the Local Planner into a network with all of the learning components of our navigation system, and train the Velocity Controller.

IV. EXPERIMENTAL SETUP

Experimental Platform. We use a simulated JackRabbit 2 robot for data collection and evaluation. The robot is a

¹It is possible to frame the objective as finding a policy that minimizes the loss ℓ under its distribution of states $s \sim d_{\pi}$ [56]. Even though this is computationally more expensive, prior work suggests that it can improve imitation [57, 58]. This alternative formulation is interesting future work.

TABLE I: Configuration of simulation environments. ‘‘G’’ denotes Geometric obstacles, ‘‘SP’’ is Static People, and ‘‘MP’’ is Moving Pedestrians. ‘‘# Train’’ indicates the number of scenarios from the corresponding env. configuration that were used for training models. ‘‘# Test’’ indicates the number of (new) scenarios used for testing.

Env.	Map	# G	# SP	# MP	# Train	# Test
E1	Foyer	8	0	0	42	0
E2	Foyer	12	0	0	17	0
E3	Foyer	6	3	2	23	0
E4	Foyer	3	2	3	1364	100
E5	Foyer	1	1	1	0	100
E6	Foyer	8	3	3	0	50
E7	Lab.	8	3	3	0	50

differential-drive mobile manipulator with a forward-facing 2D SICK lidar and an Occam 360° stereo camera, among other sensors. JackRabbit’s software stack uses the Robot Operating System (ROS) [62] for inter-process communication and logging, OpenSlam’s Gmapping for creating environment maps [63], ROS’s Adaptive Monte Carlo Localization algorithm for pose estimation [64], and YOLO [65] and DeepSort [66] for visually detecting and tracking pedestrians all around. Pedestrian tracks are converted from 2D to 3D with calibrated cameras and lidars.

Simulation Environments. We use the Gazebo simulator for our experiments. To test navigation algorithms on the simulated robot, we use the robot’s true pose from Gazebo instead of its localization algorithm, the true pedestrian motion relative to the robot instead of its vision pipeline, and a simulated forward-facing lidar. These changes allowed us to systematically study robot behavior without potential confounds due to perception errors.

To evaluate navigation in varied scenarios, we added obstacles to the simulation, e.g., 0.5-0.75 m³ boxes and cylinders, and implemented a Social Forces model [67] for the pedestrians in Gazebo. The Social Forces model enables each pedestrian to adapt their path to reach navigation sub-goals. Overall, we consider 7 different environmental conditions in our experiments (Table I). Six conditions (E1-E6) use the same map of the foyer of a university building, which is depicted in Fig. 1. The other condition uses the map of a laboratory environment. We use four different environmental conditions for training or tuning parameters (E1-E4) and four for testing (E4-E7). In particular, we use E5 and E6 for testing generalization to new environmental conditions in a known map, and E7 for testing generalization to a new environment. Each *scenario* indicated under the ‘‘# Train’’ and ‘‘# Test’’ columns of Table I corresponds to a different pair of start-end locations for the robot, and navigation sub-goals for the pedestrians. The scenarios used for testing in E4 are different than those used for training or tuning methods.

Data Collection. We collect a dataset of 1446 expert motion trajectories – each corresponding to a different scenario – for the robot with Gazebo. The robot was tele-operated with a gamepad controller for 600 scenarios in E1-E4. The remaining 764 expert motion trajectories were selected from a set of runs of the ROS’ Navigation Stack with social costs enabled [68] in E4, as tele-operation is time consuming and

deep learning benefits from big amounts of data. We tried various approaches for combining tele-operated and auto-generated data, e.g., transfer learning, but found that using all of it for training at once led to best performance in general.

Baselines. We compare our system with three baselines:

- *Nav. Stack.* ROS' layered Navigation Stack with social costs [68], Dijkstra's algorithm for global planning, and an Elastic Band local controller [69].

- *Goal Controller (GC).* Deep controller that maps lidar range measurements and a 2D navigation sub-goal to velocity commands [10]. The sub-goal moves along the global plan. It is generally 2m ahead of the robot until JackRabbit approaches the destination and the sub-goal becomes the goal. Input features are fused with concatenation as in [10].

- *Trajectory Controller (TC).* Deep controller that maps lidar measurements and the down-sampled global plan G to velocity commands. This model is based on [11] but takes lower-dimensional inputs, which facilitates learning. Input features are combined with concatenation as in [11].

The GC and TC baselines can be considered ablation models of the proposed deep local trajectory planner and velocity controller. These baselines use the same lidar features f_r and global plan features f_g as the proposed system.

Objective Metrics. For a given set of test scenarios, we consider the following metrics:

- *Running Time.* Total running time for all the scenarios.

- *Distance.* Total distance that the robot traversed considering all the scenarios.

- *Linear Vel.* Average linear velocity that the robot reported considering all the scenarios.

- *Reached Goal (RG).* Percentage of the scenarios in which the robot reached the goal.

- *Failure (F).* Percentage of the scenarios in which the robot failed catastrophically and tipped over, e.g., after a collision.

- *Collisions or Near Collisions (C).* Number of events in which the robot's lidar sensed an obstacle closer than 0.3m. The count considers all scenarios; it is not averaged.

- *Pedestrian Collisions (PC).* Number of events in which the robot collided with a pedestrian (their distance was less than 0.5 m). The count considers all scenarios.

- *Violations of Personal Space (PS).* Number of events in which the robot violated personal space [16] and was less than 1.2 m from a person. The count considers all scenarios.

Subjective Metrics. We conduct a survey to gather qualitative perceptions of navigation performance. Participants rate navigation behavior on three 5-point Likert scales: *Aggressive*, *Natural*, and *Efficient* navigation.

Other Details. We consider that the robot reached the goal if its distance to the destination is less than 0.5 m. For all the scenarios, we limit the time that the robot can take to reach the destination to 1.5 min. Finally, the robot has a safe, maximum limit of 0.6 m/s and 1.2 rad/s on its absolute linear and angular speeds.

V. EXPERIMENTAL EVALUATION

A. Evaluation Based on Objective Metrics

Experiments in Previously-Seen Maps. We first conducted an evaluation on the Foyer map, which was used for training or tuning of algorithms, to evaluate the capacity of the models on a previously-seen environmental condition (E4) and in new conditions (E5 and E6). As indicated in Table I, we considered 100 test scenarios according to each of E4 and E5, and 50 test scenarios according to E6.

Table II presents quantitative results for this experiment. Overall, all methods increased speed as the environments were simpler, suggesting that they adjusted navigation behavior based on environmental conditions. Unfortunately, though, collisions were observed throughout the experiment. In terms of relative performance on the familiar condition E4, the proposed approach reached the goal the most. For new environmental conditions, the Navigation Stack was the best model in the simplest condition (E5), while the proposed approach was the best model in the more complex condition (E6). In E5, the performance of our system is worse than in the other two environments. We observed that in simulation, our model paid too much attention to the single pedestrian, and halted multiple times to let the pedestrian pass and as a result, could not reach the goal within 90 seconds. Remarkably, the performance of our approach varied only by $\pm 3\%$ when reaching the goal in the foyer, while the performance of the Nav. Stack varied by about $\pm 8\%$. It is difficult to manually find a fixed set of model parameters for all environmental conditions.

The GC and TC baselines showed their best performance in the simple condition E5, even in comparison to E4. This result suggests that the controllers were able to follow the global plan, but had trouble adapting to dynamic environments, which were more common in E4 and E6.

Generalization to a New Map. We tested generalization capabilities in a new map of a university laboratory (environmental condition E7). The results are presented in the last four rows of Table II. Our proposed model reached the goal the most often, followed by the navigation stack. There is a marked difference in the goal reach percentages of the proposed system and the navigation stack. This result again highlights the difficulty in manually tuning static model parameters for navigation with ROS' navigation stack.

B. Evaluation Based on Subjective Metrics

We surveyed 35 people through Amazon Mechanical Turk in regards to their perception of the motion of the robot in 48 videos of scenarios from environmental conditions E4, E5, E6, and E7 (Table I). We then ran REstricted Maximum Likelihood (REML) [70] analyses on how Aggressive, Natural, and Efficient the motion looked with *Method* (Nav. Stack, GC, TC, Proposed) as main effect and participant as random effect. The results for Efficiency were significant, $F[3, 1642] = 19.26$, $p < 0.01$. A Tukey post-hoc test [71] showed that the proposed navigation approach was perceived as significantly more efficient ($M=3.69$, $SE=0.06$) than the

TABLE II: Objective results. \uparrow : higher is better; \downarrow : lower is better. Please see Sec. IV and V for more details.

Model	Env.	Running Time	Distance	Linear Vel.	RG \uparrow	F \downarrow	C \downarrow	PC \downarrow	PS \downarrow
Nav. Stack	E4	60.1 min	802.7 m	0.22 m/s	91%	0/100	64	1	82
Goal Cont. (GC)	E4	55.4 min	901.9 m	0.27 m/s	89%	2/100	167	12	108
Traj. Cont. (TC)	E4	49.8 min	993 m	0.33 m/s	85%	7/100	129	7	112
Proposed System	E4	54.2 min	923.3 m	0.28 m/s	95%	0/100	102	1	76
Nav. Stack	E5	56.8 min	892.2 m	0.26 m/s	98%	0/100	0	1	28
Goal Cont. (GC)	E5	46.4 min	930.2 m	0.33 m/s	98%	0/100	52	1	27
Traj. Cont. (TC)	E5	45.3 min	961.5 m	0.35 m/s	97%	1/100	61	0	43
Proposed System	E5	59.6 min	941.7 m	0.26 m/s	93%	0/100	27	1	34
Nav. Stack	E6	38.7 min	424.8 m	0.18 m/s	82%	0/50	67	0	80
Goal Cont. (GC)	E6	34.0 min	333.0 m	0.16 m/s	74%	1/50	178	1	65
Traj. Cont. (TC)	E6	39.4 min	472.2 m	0.2 m/s	66%	1/50	152	4	65
Proposed System	E6	34.4 min	484.4 m	0.23 m/s	98%	0/50	112	1	62
Nav. Stack	E7	25.2 min	276.26 m	0.18 m/s	66%	0/50	12	0	17
Goal Cont. (GC)	E7	45.9 min	540.01 m	0.19 m/s	62%	0/50	21	2	28
Traj. Cont. (TC)	E7	46.2 min	416.81 m	0.15 m/s	64%	1/50	18	1	28
Proposed System	E7	39.4 min	503.95 m	0.21 m/s	84%	0/50	16	0	35

Nav. Stack and TC ($M=3.46$, $SE=0.07$; $M=3.05$, $SE=0.07$). The results for Aggressiveness and Naturalness were significant as well with $p < 0.01$. The post-hoc showed that the TC baseline led to the most aggressive ($M=2.7$, $SE=0.06$) and least natural ($M=3.14$, $SE=0.06$) navigation behavior in comparison to all other methods. In particular, the ratings for our approach in terms of aggressiveness and naturalness were $M=2.30$, $SE=0.06$, and $M=3.62$, $SE=0.06$, respectively.

C. Attention Visualization

We inspected the attention coefficients from the learned components of the proposed navigation system and found that they correlate consistently with specific task conditions, providing a mechanism to interpret the execution of the model. For instance, as the robot navigated nearby obstacles, the attention coefficient b_r for the lidar features of the Velocity Controller tended to grow, while the attention coefficient b_g for the global plan features tended to get lower (left image of Fig. 3). Meanwhile, the opposite result was observed when the robot approached the goal (right image of Fig. 3). These results suggest that the proposed attention mechanisms helped the robot extract the right information from the many inputs to the components of our deep model. More visualizations can be seen in the supplementary video.

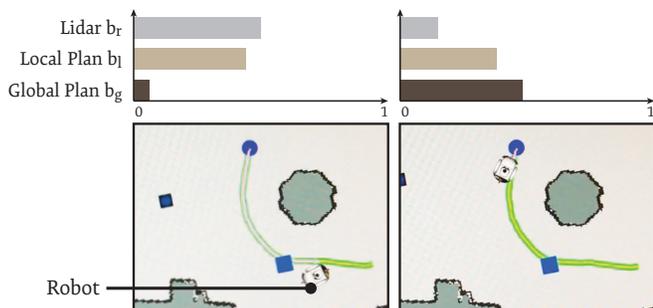


Fig. 3: Attention visualization for the Velocity Controller. Left: The robot navigates near an obstacle (blue square). Right: the robot approaches the goal (blue circle).

VI. LIMITATIONS

We conducted experiments in relatively open simulated environments, but future work should also evaluate the performance of the proposed approach in more narrow spaces, like corridors. Further, it is important to conduct a systematic evaluation of our model in real-world scenarios. While we have attempted such preliminary tests and our system has worked effectively several times, it has also failed due to errors in people tracking and noise in real lidar measurements, e.g., due to material reflections, that were not typical of our simulation. We continue to work to address these issues.

VII. CONCLUSION

We proposed a new navigation system that combines traditional planning with modern deep learning techniques. The learning components of the system were structured as a local planner and a velocity controller. This structure made our model interpretable. First, by visualizing the local plan, one could get a sense of how the robot would move and react to dynamic elements of the environment. Second, by inspecting the attention coefficients of our model, one could see the type of information that the robot was considering during navigation. Overall, our evaluation of the proposed model suggested that it was effective for navigation in complex dynamic environments. The performance of the proposed model was more consistent than the performance of a traditional two-layer navigation system with social costs [68]. Moreover, the learning component of our approach was more successful at reaching the goal than other deep controllers that mapped sensor data and a global plan directly to velocity commands, e.g., [10]. Finally, our results reinforce the idea that imitation learning can facilitate modeling socially appropriate behavior from example navigation data [40, 41].

ACKNOWLEDGMENTS

The Toyota Research Institute provided funds to assist with this research, but this paper solely reflects the opinions and conclusions of its authors and not of any Toyota entity.

REFERENCES

- [1] D. Nakhaeina, S. H. Tang, S. M. Noor, and O. Motlagh, "A review of control architectures for autonomous navigation of mobile robots," *International Journal of Physical Sciences*, vol. 6, no. 2, pp. 169–174, 2011.
- [2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
- [3] 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.
- [4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.
- [5] R. Kirby, "Social robot navigation," Ph.D. dissertation, Carnegie Mellon University, The Robotics Institute, 2010.
- [6] B. P. Gerkey and K. Konolige, "Planning and control in unstructured terrain," in *ICRA Workshop on Path Planning on Costmaps*, 2008.
- [7] D. V. Lu, D. B. Allan, and W. D. Smart, "Tuning cost functions for social navigation," in *International Conference on Social Robotics*. Springer, 2013, pp. 442–451.
- [8] 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.
- [9] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.
- [10] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1527–1533.
- [11] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intentionet: Integrating planning and deep learning for goal-directed autonomous navigation," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 185–194.
- [12] G. Sepulveda, J. C. Niebles, and A. Soto, "A deep learning based behavioral approach to indoor autonomous navigation," *arXiv preprint arXiv:1803.04119*, 2018.
- [13] A. Kelly, *Mobile robotics: mathematics, models, and methods*. Cambridge University Press, 2013.
- [14] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.
- [15] 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*, 2006, pp. 739–746.
- [16] E. T. Hall, *The hidden dimension*. Garden City, NY: Doubleday, 1966, vol. 609.
- [17] A. Kendon, *Conducting interaction: Patterns of behavior in focused encounters*. CUP Archive, 1990, vol. 7.
- [18] 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*. Springer, 2006, pp. 537–544.
- [19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
- [20] M. W. Otte, "A survey of machine learning approaches to robotic path-planning," *Cs. Colorado. Edu*, 2015.
- [21] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [22] S. Thrun and T. M. Mitchell, "Lifelong robot learning," in *The biology and technology of intelligent autonomous agents*. Springer, 1995, pp. 165–196.
- [23] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [24] J. Bagnell, J. Chestnutt, D. M. Bradley, and N. D. Ratliff, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems*, 2007, pp. 1153–1160.
- [25] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.
- [26] J. Ho, J. Gupta, and S. Ermon, "Model-free imitation learning with policy optimization," in *International Conference on Machine Learning*, 2016, pp. 2760–2769.
- [27] J. A. Bagnell, "An invitation to imitation," Carnegie Mellon University, Robotics Institute, Tech. Rep., 2015.
- [28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [29] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," *arXiv preprint arXiv:1702.03920*, vol. 3, 2017.
- [30] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 137–153, 2015.
- [31] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 981–986.
- [32] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard, "Time dependent planning on a layered social cost map for human-aware robot navigation," in *Mobile Robots (ECMR), 2015 European Conference on*. IEEE, 2015, pp. 1–6.
- [33] M. Luber, L. Spinello, J. Silva, and K. O. Arras, "Socially-aware robot navigation: A learning approach," in *Intelligent robots and systems (IROS), 2012 IEEE/RSJ international conference on*. IEEE, 2012, pp. 902–907.
- [34] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2153–2160.
- [35] H. Kretschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.
- [36] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2096–2101.
- [37] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *CoRR*, vol. abs/1703.08862, 2017. [Online]. Available: <http://arxiv.org/abs/1703.08862>
- [38] B. Okal and K. O. Arras, "Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2889–2895.
- [39] C. Johnson and B. Kuipers, "Socially-aware navigation using topological maps and social norm learning," in *AAAI/ACM Conf. on Artificial Intelligence, Ethics, and Society (AIES)*, 2018.
- [40] K. Shiarlis, J. Messias, and S. Whiteson, "Acquiring social interaction behaviours for telepresence robots via deep learning from demonstration," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 37–42.
- [41] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially-compliant navigation through raw depth inputs with generative adversarial imitation learning," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, May 2018.
- [42] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *CoRR*, vol. abs/1804.09364, 2018. [Online]. Available: <http://arxiv.org/abs/1804.09364>
- [43] Z. Shiller and Y.-R. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, 1991.
- [44] J.-P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 577–593, 1994.
- [45] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 811–818.
- [46] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 41, 2017.
- [47] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2154–2162.

- [48] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," *arXiv preprint arXiv:1701.01909*, vol. 4, no. 5, p. 6, 2017.
- [49] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018*, May 2018.
- [50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [51] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. CONF, 2018.
- [52] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.
- [53] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," *arXiv preprint arXiv:1606.01847*, 2016.
- [54] H. Ben-Younes, R. Cadene, M. Cord, and N. Thome, "Mutan: Multimodal tucker fusion for visual question answering," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 3, 2017.
- [55] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
- [56] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.
- [57] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving via end-to-end deep imitation learning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.
- [58] A. Venkatraman, R. Capobianco, L. Pinto, M. Hebert, D. Nardi, and J. A. Bagnell, "Improved learning of dynamics models for control," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 703–713.
- [59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [60] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.
- [61] K. A. Krueger and P. Dayan, "Flexible shaping: How learning in small steps helps," *Cognition*, vol. 110, no. 3, pp. 380–394, 2009.
- [62] 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*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
- [63] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2432–2437.
- [64] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, 2002, pp. 713–720.
- [65] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint*, 2017.
- [66] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.
- [67] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.
- [68] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 709–715.
- [69] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 802–807.
- [70] R. R. Corbeil and S. R. Searle, "Restricted maximum likelihood (reml) estimation of variance components in the mixed model," *Technometrics*, vol. 18, no. 1, pp. 31–38, 1976.
- [71] H. Abdi and L. J. Williams, "Tukeys honestly significant difference (hsd) test," *Encyclopedia of Research Design*. Thousand Oaks, CA: Sage, pp. 1–5, 2010.