Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations

Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto

***Abstract*—This letter presents a case study of a learning-based approach for target-driven mapless navigation. The underlying navigation model is an end-to-end neural network, which is trained using a combination of expert demonstrations, imitation learning (IL) and reinforcement learning (RL). While RL and IL suffer from a large sample complexity and the distribution mismatch problem, respectively, we show that leveraging prior expert demonstrations for pretraining can reduce the training time to reach at least the same level of the performance compared to plain RL by a factor of 5. We present a thorough evaluation of different combinations of expert demonstrations, different RL algorithms, and reward functions, both in simulation and on a real robotic platform. Our results show that the final model outperforms both standalone approaches in the amount of successful navigation tasks. In addition, the RL reward function can be significantly simplified when using pretraining, e.g., by using a sparse reward only. The learned navigation policy is able to generalize to unseen and real-world environments.**

***Index Terms*—Navigation, deep reinforcement learning, end-to-end planning.**

I. Introduction

AUTONOMOUS navigation in environments where global knowledge of the map is available is nowadays well understood [1]. Optimization objectives like, e.g., minimum path length, travel time or safe distance to obstacles can be used to find the optimal path connecting the start and goal position of a robot. However, full knowledge of the map is not always available in practice, e.g., in search and rescue applications or rapidly changing environments. If no reliable environment map can be used for navigation, classical path planning approaches [1] might fail. Given only local perception of the robot and a relative target position, robust map-less navigation strategies are required. In recent years, machine learning techniques—with neural networks leading the way [2]–[4]—have gained importance allowing for the application of end-to-end motion planning approaches. Instead of splitting the navigation task into multiple sub-modules like, e.g., sensor fusion, obstacle detection, global and local motion planning, end-to-end approaches use a direct mapping from sensor data to robot motion commands which can reduce the complexity during deployment significantly.

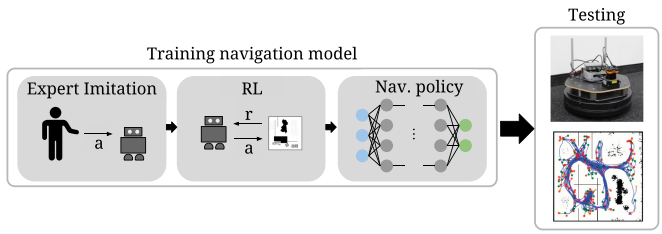

Fig. 1. An end-to-end navigation policy is learned from a combination of imitation and reinforcement learning. The resulting policy is tested thoroughly in simulation and on a real robotic platform.

Current state-of-the-art end-to-end planning approaches can be split in two major groups: (i) imitation learning (IL) based ones use supervised learning techniques to imitate expert demonstrations as close as possible,[1] and (ii) approaches based on Reinforcement learning (RL) where the agents learn their navigation policy by trial and error exploration combined with reward signals. imitation learning (IL) is sample efficient and can achieve accurate imitation of the expert demonstrations. Given the training data, satisfactory navigation models can be found within a few hours of training [2]. However, it is likely to overfit to the environment and situations presented at training time. This limits the potential for generalization and the robustness of the policy (distribution mismatch). Reinforcement learning (RL) is conceptually more robust—also in unseen scenarios—as the agent learns from its own mistakes during training [3]. The disadvantage of RL is its sample inefficiency and missing safety during training, limiting the current utilization to applications where training can be conducted using extremely fast simulators [5]. As for RL training, episodes need to be forward simulated (on- or off-policy), training iterations are significantly more time consuming than in IL, which reduces the number of training iterations in a given time. However, RL allows to encode desired behavior—such as reaching the target

Manuscript received May 3, 2018; accepted August 23, 2018. Date of publication September 10, 2018; date of current version September 24, 2018. This letter was recommended for publication by Associate Editor J. Kober and Editor T. Asfour upon evaluation of the reviewers' comments. This work was supported by the European Union Horizon 2020 project CROWDBOT under Grant 779942. *(Mark Pfeiffer and Samarth Shukla contributed equally to this work.) (Corresponding author: Mark Pfeiffer.)*

The authors are with the Autonomous Systems Lab, Computer Vision Lab, Learning and Adaptive Systems Group, and Max Planck ETH Center for Learning Systems, ETH Zurich, Zurich 8057, Switzerland (e-mail: pfmark@ethz.ch; shuklas@ethz.ch; matteotu@ethz.ch; cesarc@ethz.ch; krausea@ethz.ch; rsiegwart@ethz.ch; nietoj@ethz.ch).

Digital Object Identifier 10.1109/LRA.2018.2869644

[1] Also known as behavioral cloning.

and avoiding collisions — specifically in a reward function and does not only rely on suitable expert demonstrations. In addition, RL maximizes the overall expected return on a full trajectory, while IL treats every observation independently [6], which conceptually makes RL superior to IL.

In this work, we present and analyze an approach that combines the advantages of both IL and RL. It is inspired by human learning, which typically combines the observation of other people and self-exploration [7]. Our approach, in the following called *Reinforced imitation learning (R-IL)*, combines supervised IL based on expert demonstrations to pre-train the navigation policy with subsequent RL (see Fig. 1). For RL, we use Constrained Policy Optimization (CPO) [8] due to its ability to incorporate constraints during training. This allows for safer training and navigation, which is especially important for real-world mobile robotics.

We hypothesize that the combination of the two learning approaches yields a more robust policy than pure IL, and that it is also easier and faster to train than pure RL. In addition, by enforcing the collision avoidance by constraint instead of a fixed penalty in the reward function, the amount of collisions during training and testing should be decreased. To the best of our knowledge, this is the first work to explore this combination for robot navigation and also to apply constraint-based RL to map-less navigation. We provide an extensive evaluation of the training and navigation performance in simulation and on a robotic platform. Our main contributions are:

- a case study for combining IL and RL[2] for map-less navigation
- a model for map-less end-to-end motion planning that generalizes to unseen environments
- an extensive evaluation of training and generalization performance to unseen environments

II. Related Work

A. Learning by Demonstration

Learning by demonstration can be split in two main areas: (i) Inverse reinforcement learning (IRL), where a reward function is inferred from expert demonstrations and a policy is derived by optimizing this reward with optimal control techniques and (ii) IL, where expert demonstrations are used to directly infer a policy. Abbeel *et al.* [9] present an IRL-based approach where they teach an autonomous car to navigate in parking lots by observing human demonstrations. Similarly, Pfeiffer *et al.* [10] and Kretzschmar *et al.* [11] present approaches for navigation in dynamic environments based on IRL. By observing pedestrian motion, a probability distribution over pedestrian trajectories is found. For path planning, the trajectory with the highest probability according to the learned model is chosen with the goal of a close imitation of pedestrian motion. Wulfmeier *et al.* [12] present a similar approach using deep IRL instead of a combination of classical features in order to learn how to drive an autonomous car through static environments.

In the following, we give an overview of the literature on map-less navigation using IL. Muller *et al.* [4] present an image-based approach for end-to-end collision avoidance using imitation learning. In their work, the focus is on feature extraction and on generalization to new situations. The overall navigation performance of such approaches is not analyzed. Another approach focused on perception is presented by Chen *et al.* [13]. They combine learning-based feature extraction using Convolutional neural networks (CNNs) with a classical driving controller for an autonomous car. However, they focus on a lane-following application and do not deal with target-driven navigation. Kim *et al.* [14] present an IL approach for hallway navigation and collision avoidance for an unmanned aerial vehicle (UAV). They show a working model on a real-world platform, yet the environmental setup is relatively easy and no real navigation capabilities are required. Sergeant *et al.* [15] present an end-to-end approach for laser-based collision avoidance for ground vehicles demonstrated in simulation and real-world tests. However, the approach is limited to collision avoidance and cannot be used for target-driven navigation. Ross *et al.* [16] present the Dataset Aggregation (DAGGER) method which collects demonstrations according to the currently best policy but can also query additional expert demonstrations in order to alleviate the distribution mismatch problem. One application of the DAGGER algorithm is presented in [17], where directional commands for forest navigation and collision avoidance are learned from expert demonstrations. In addition, Kuefler *et al.* [6] presented an approach based on Generative Adversarial Imitation Learning (GAIL) [18], where they learn driver models for an autonomous car based on expert demonstrations. Tai *et al.* [19] recently applied GAIL to model interaction-aware navigation behavior. Although conceptually GAIL generalizes better than standard behavioral cloning techniques, it is still constrained by the provided expert demonstrations.

The method we introduce builds upon prior work presented in [2], where a global planner is used to generate expert demonstrations in simulation. Given demonstrations, an end-to-end navigation policy mapping from 2D laser measurements and a relative goal position to motion commands is found. The main drawbacks of this approach are the generalization to new environments—also due to the specific CNN model structure—and the behavior in situations which were not covered in the training data.

B. Reinforcement Learning

Bischoff *et al.* [20] use ideas from hierarchical RL to decompose the navigation task in motion planning and movement execution and thus are able to improve the sample efficiency of plain RL. Yet global map information is always assumed to be known. Zuo *et al.* [21] use a popular model-free RL algorithm, Q-learning, to teach a robot a policy to navigate through a simple spiral maze from sonar inputs only. Mirowski *et al.* [22] use auxiliary tasks such as depth prediction and loop closure assessment to improve the learning rate of A3C [5] for simulated maze navigation from RGB images. Bruce *et al.* [23] use interactive experience replay to learn how to navigate in a known environment to a fixed goal from images by traversing it only once. The method presented in [24] focuses on efficient knowledge transfer across maps and conditions for an autonomous navigation task. To this end, it uses a particular parametrization of the Q-function, known as successor representation, that

[2]Our source code is available here: https://github.com/ethz-asl/rl-navigation

decouples task specific knowledge from transferable knowledge. Zhu *et al.* [25] present an end-to-end vision-based navigation algorithm that uses the target as an additional input to the policy to learn to achieve proper target-driven navigation.

Chen *et al.* [26] presented a RL approach for collision avoidance in dynamic environments. Similar to our work, prior demonstrations are used for pre-training, yet their focus lies on learning interactions between multiple agents and the algorithm is not designed for navigation scenarios. The method presented by Tai *et al.* [3] is the most closely related to ours. In their work, the Asynchronous Deep Deterministic Policy Gradients (ADDPG) algorithm is used to learn a policy from range findings to continuous steering commands for both simulated and real-world map-less navigation tasks. However, using ADDPG, no collision constraints can be enforced and the models are trained from scratch. When moving towards real world applications and eventually RL training on real platforms, safety and training speed become decisive factors. Therefore, compared to [3], we use prior demonstrations for pre-training and CPO during RL training, targeting the real-world applicability of RL approaches.

As experiments in robotics usually require large amounts of time, the problem of reducing the sample complexity of RL based approaches has received increasing attention recently. Using a combination of IL and RL to obtain a sample efficient and robust learning algorithm has previously been explored in robotics in the context of manipulation tasks [27], [28]. In this context, the main challenge consists in using human demonstrations that may not be replicable by the robot due its dynamics. In the case of navigation, this is usually not a concern. However, navigation tasks present challenges in terms of safety. Even small deviations from the expert policy may lead to a crash. To the best of our knowledge, our method is the first to use expert demonstrations to boost RL learning performance in the context of map-less autonomous navigation.

III. Approach

A. Problem Formulation

Classical path planning techniques [1] require prior knowledge of the environment for navigation. In case of unknown or constantly changing and dynamic environments, obtaining and maintaining an accurate map representation becomes increasingly difficult or even unfeasible. Therefore, map-less navigation skills based solely on local information available to the robot through its sensors are required.

Given the sensor measurements $\mathbf{y}$ and a relative target position $\mathbf{g}$, we want to find a policy $\pi_{\boldsymbol{\theta}}$ parametrized by $\boldsymbol{\theta}$ which maps the inputs to suitable control commands, $\mathbf{u}$, i.e.

$$\mathbf{u} = \pi_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{g}). \tag{1}$$

The required control commands are comprised of the translational and rotational velocity. As the mapping from local sensor and target data to control commands can be arbitrarily complex, learning how to plan from experience in an end-to-end fashion using powerful non-linear function approximators, such as neural networks, has become more prominent.

In this work, we aim at combining IL and RL to obtain a sample efficient and robust learning based navigation algorithm.

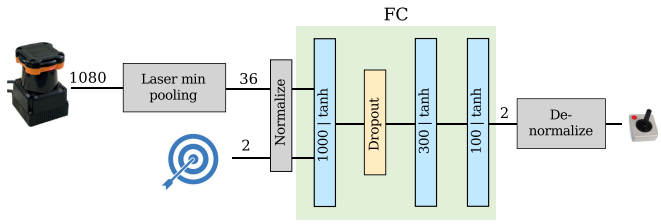

Fig. 2. The neural network model for $\pi_{\boldsymbol{\theta}}$. The normalized input data is fed through three fully connected layers with tanh activation functions. Between layer one and two, dropout is added during IL training. The outputs are de-normalized to obtain physical control commands from the neural network.

We do this in a sequential fashion by using the result from IL to initialize our RL method. In the remainder of this section we introduce separately the underlying neural network model, the IL and RL components of our method.

B. Neural Network Model

The neural network model which represents $\pi_{\boldsymbol{\theta}}$, is shown in Fig. 2. In this work, the inputs to the model are 2D laser range findings and a relative target position in polar coordinates w.r.t. the local robot coordinate frame. In contrast to [2], where a CNN was used to extract environmental features, this model is simplified and only relies on three fully connected layers. While the CNN allows to find relevant environmental features, we found that it tends to overfit to the shapes of the obstacles presented during training. Instead, we use minimum pooling of the laser data and compress the full range of 1080 measurements into 36 values, where each pooled value $\mathbf{y}_{p,i}$ is computed as:

$$\mathbf{y}_{p,i} = \min\left(\mathbf{y}_{i\cdot k}, \ldots, \mathbf{y}_{(i+1)\cdot k-1}\right), \tag{2}$$

where i is the value index and k is the kernel size for 1D pooling. In our case, we chose $k = 30$. Using min-pooling, safety can be assured, yet detailed environmental features may get lost. The resulting simplified neural network model can be trained more efficiently and is less likely to overfit to specific obstacle shapes. Furthermore, the inputs are normalized before being fed to the neural network model. The pooled laser measurements are cropped and then mapped to lie in the interval $[-1, 1]$ by applying the normalization $2 \cdot \left(1 - \frac{\min(\mathbf{y}_{p,i}, r_{\max})}{r_{\max}}\right) - 1$, where $r_{\max}$ is the maximum laser range. The same normalization is applied to the relative target position. The outputs of the neural network, which also lie in the interval $[-1, 1]$, are de-normalized and mapped to translational and rotational velocities.

C. Supervised Pre-Training Via Behavior Cloning

In order to improve the performance and sample complexity of the succeeding RL, the policy is pre-trained using supervised IL based on expert demonstrations similar to [2]. The goal is to imitate the expert as closely as possible, given the representation limitations of the neural network model. Compared to plain IL, where the performance of the final model is limited by the performance of the expert demonstrations, R-IL can overcome this limitation through self-improvement.

D. Reinforcement Learning

1) Background Information: Given a Markov Decision Process (MDP), $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(\cdot|s_t, a_t) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} :\rightarrow \mathbb{R}_+$ is the transition probability distribution, $\mathcal{R}(\cdot, \cdot) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function and $\gamma \in [0, 1]$ is the discount factor, RL aims to find a policy π_θ, mapping states to actions and parametrized by $\boldsymbol{\theta}$, that maximizes the expected sum of discounted rewards,

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \cdot r(s_t, \pi_\theta(s_t))\right], \tag{3}$$

where T is the time horizon of a navigation episode. In our case, s_t consists of laser measurements and the target information, a_t of the control commands.

Policy gradient methods [29] are model-free RL algorithms that use modifications of stochastic gradient descent to optimize $J(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$. However, they suffer from a high variance in gradients, resulting in undesirably large updates to the policy. A popular technique to reduce model variance and ensure stability between updates is Trust Region Policy Optimization (TRPO) [30]. To this end, it restricts the change in policy at each update by imposing a constraint on the average Kullback-Leibler (KL) divergence between the new and old policy.

Enforcing safety is crucial when dealing with mobile robotics applications. Often, safety in RL is encouraged by imposing high cost on unsafe states. However, this requires tuning such cost. If it is too low, the agent may decide to experience unsafe states for short amounts of time as this will not severely impact the overall performance (Eq. 3). Conversely, if the cost is too high, the agent may avoid exploring entire portions of the state space to avoid the risk of experiencing unsafe states. A more elegant and increasingly popular way of ensuring safety in RL is to treat it as a constraint [8], [31]. In particular, in this work, we use a safety constrained extension of TRPO known as Constrained Policy Optimization (CPO) [8] to ensure safety. Given a cost function $C : \mathcal{S} \times \mathcal{A} :\rightarrow \mathbb{R}$, let $J_C(\theta)$ indicate the expected discounted return of π_θ with respect to this cost

$$J_C(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \cdot C(s_t, \pi_\theta(s_t))\right]. \tag{4}$$

CPO finds an approximate solution to the following problem,

$$\boldsymbol{\theta}^* = \arg\max J(\boldsymbol{\theta}), \text{ s.t. } J_C(\boldsymbol{\theta}) \leq \alpha. \tag{5}$$

2) Training Process: For training, the neural network model is first initialized either randomly (pure RL) or using IL (R-IL). We use a stochastic policy where the actions are sampled from a 2D Gaussian distribution having the de-normalized values of the output of the neural network as mean, and a 2D standard deviation which is a separate learn-able parameter. Using a supervised IL model thus only influences the initialization of the RL policy. During training we randomly select a start and target position and collect robot experience samples by running an episode using the current policy $\pi_{\boldsymbol{\theta}}$ for a fixed number of time steps or until the robot reaches the target. At each policy update, we use a batch of samples collected from multiple episodes.

The agent's objective is to learn to reach the target in the shortest possible number of time-steps while avoiding collisions with surrounding obstacles. The reward function provides the required feedback to the robot during the learning process. In this work, we investigate different choices for the reward function encoding various degree of information about the task. These rewards can be expressed by:

$$r(s_t) = \begin{cases} 10, & \text{if success} \\ -(d(s_t) - d(s_{t-1})), & \text{otherwise.} \end{cases}$$

Setting $d(s) = 0$, $\forall s \in \mathcal{S}$ we encode the minimum information required to carry out the task. This *sparse reward* makes the learning process difficult due to the credit assignment problem, i.e. the fact that all the actions taken in an episode get credit for its outcome regardless of whether they contributed to it or not. An alternative to such choice is to set $d(s)$ to the *Euclidean distance* between s and the target. This reward provides continuous feedback for each action by rewarding/penalizing the agent for getting closer/further to/from the goal in Euclidean space. However, it does not consider the placement of obstacles in the environment. The last option we investigate consists in setting $d(s)$ to the distance between s and the goal along the *shortest feasible path* that can be computed using the Dijkstra algorithm. Note, the agent does not have any knowledge about $d(\cdot)$. This distance is only used to compute the reward which the agent receives from the environment during training.

Using a negative reward for collisions makes the policy highly sensitive to this reward's magnitude, resulting in a delicate trade-off between two different objectives — reaching the target and avoiding crashes. However, in constrained MDPs, we can encode collision avoidance through a constraint on the expected number of crashes allowed per episode. Let $\mathcal{S}_c \subset \mathcal{S}$ denote the set of states that correspond to a crash. We define a state dependend cost function as follows:

$$c(s_t) = \mathcal{I}(s_t \in \mathcal{S}_c), \tag{6}$$

where $\mathcal{I}$ is the indicator function. In our experiments, we noticed the robot stays in a crash state for four consecutive timesteps on average. By setting the discount factor for the cost — which does not have to be equal to the one for the reward — close to 1 and introducing the constraint value α, we can constrain the total number of expected crashes per episode to be approximately less or equal to $\frac{\alpha}{4}$. In our model we set $\alpha = 0.4$. This value was found empirically by testing values between 0.0 and 0.6 in a simple environment. While training, we allow for multiple crashes in each episode. This leads to more crash samples in the training set and makes it easier to reach the target, thus making the training process more efficient.

IV. Experiments

This section presents the experiments conducted in simulation and on the real robotic platform. The goal of the experiments is to investigate the influence of pre-training the RL policy, to compare constraint-based to fixed penalty methods and analyze the influence of the reward functions presented in Section III. We also compare to models presented in prior work [3]. Furthermore, we investigate the generalization performance of the navigation policies to unseen scenarios and the real world,

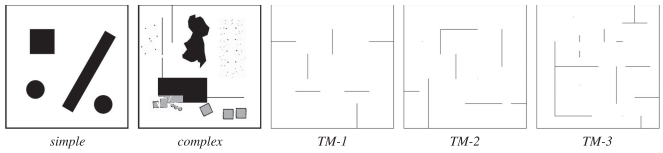

Fig. 3. Training maps for IL and RL. The *TM* vary significantly in difficulty. Maps can be better viewed by zooming in on a computer screen.

which is also shown in our video.[3] Our work does not intend to show that we can outperform a global graph-based planner in known environments, where graph-based solutions are fast and can achieve optimal behavior. The goal of our experiments is to investigate the limits of motion planning with local information only.

A. Experimental Setup

The models are purely trained in simulation since it is a safe, fast and efficient way of training and evaluating the model. Additionally, there are no physical space constraints and the environment structure can be changed almost arbitrarily. Models trained in simulation have previously been shown to successfully transfer to the real-world [2], [3], [25].

The experiments are based on a differential drive Kobuki TurtleBot2[4] platform equipped with a front-facing Hokuyo UTM laser range finder with a field of view of 270°, maximum range of 30 m and 1080 range measurements per revolution. For on-board computations we resort to an Intel NUC with an i7-5557U processor and without any GPU, running Ubuntu 14.04 and ROS [32] as a middleware. The motion commands are published with a frequency of 5 Hz.

B. Model Training

Different procedures for model training are applied: (i) pure IL, (ii) pure RL and (iii) R-IL, which is a combination of both. In order to test the influence of the complexity and the diversity of the training environments on test performance, we train the models on five maps (or subsets of them) as shown in Fig. 3. The pure IL models are trained in the *simple* and *complex* maps, the RL part is conducted on all three *TM* maps. Similarly, for R-IL, IL is conducted on the *simple* and *complex* maps and the RL part takes place on the *TM* maps. We do this separation in order to investigate how demonstrations from a different environment can be transferred to the RL training.

The expert demonstrations used for IL are generated using the ROS `move_base`[5] navigation stack to navigate between random start and target positions, as presented in [2]. We use an expert planner instead of a human to make the demonstrations more consistent and time efficient. We note that the demonstrations are suboptimal for RL, as they are generated based on a different cost function and also in a different environment. After recording the demonstrations, one IL training iteration takes around 7 ms on an Intel i7-7700K processor and a Nvidia GeForce GTX 1070 GPU. Therefore, IL model training takes between one hour (s_{10}, 500 k iterations) and around 2.5 hours (c_{1000}, 1.5 M iterations).

[3]https://youtu.be/uc386uZCgEU
[4]http://kobuki.yujinrobot.com/about2
[5]http://wiki.ros.org/move_base

TABLE I
MODEL DETAILS, INCLUDING THE MAPS AND NUMBER OF TRAJECTORIES USED FOR IL AND THE REWARD SIGNAL USED FOR RL. BESIDES CPO1, ALL MODELS ARE TRAINED ON ALL THREE TM MAPS. CPO AND TRPO IN THE MODEL NAME SPECIFY THE RL TRAINING PROCEDURE, THE SUBSCRIPT OF TRPO INDICATES THE FIXED PENALTY WEIGHT FOR COLLISIONS

	model name	IL-map(s)	#IL traj.	RL reward
R-IL	s_{10}+CPO$_{sparse}$	simple	10	sparse
	s_{10}+CPO$_{Eucl.}$	simple	10	Euclidean
	s_{10}+CPO	simple	10	short
	s_{1000}+CPO	simple	1000	short
	123_{1500}+CPO	1+2+3	500 each	short
	c_{1000}+CPO	complex	1000	short
	s_{1000}+TRPO$_{c0.1}$	simple	1000	short
	s_{1000}+TRPO$_{c1.0}$	simple	1000	short
IL	s_{10}	simple	10	—
	c_{1000}	complex	1000	—
RL	CPO1	—	0	short
	CPO123	—	0	short
	CPO123$_{sparse}$	—	0	sparse
	TRPO123$_{c0.1}$	—	0	short

Table I summarizes all the models we trained. Our case study presents constraint based R-IL yet compares to a broad range of different models: We vary the number of demonstrations (from 10 to 1000), the RL training procedure (CPO, TRPO) and reward signals (sparse, Euclidean and shortest distance) in order to provide insights into how those factors influence map-less navigation. The TRPO training procedure is the fixed collision penalty version of CPO with a collision constraint, as described in Section III.

During RL, the training environment is uniformly sampled among the three *TM* maps (see Fig. 3). One training iteration—for which we consider a batch consisting of 60 k time steps—takes around 180 s using the accelerated Stage [33] simulation. Therefore, 1000 iterations require around 50 hours of training time using the simulation, which is a real-time equivalent of around 100 days. This further motivates the need to find a good policy initialization by IL in order to reduce the training time significantly.

Fig. 4 shows the success and crash rates of a broad range of models during RL training alongside the performance of pure IL trained on all *TM* maps. This IL model only serves as a baseline to evaluate the progress of the RL and R-IL methods during training. CPO1 differs from all the other models during training as it is exclusively trained on the simplest *TM* map (*TM*1). However, it will be shown that this model does not generalize well to more complex test environments. From Fig. 4 the following can be shown:

1) Difference Between the Models Which Were Pre-Trained Using IL and the Ones Based on Pure RL Using CPO / TRPO: While the pre-trained models already start at a certain success rate (depending on the performance of the IL model), it takes a significant amount of iterations for the RL models to reach the target in the majority of the cases. Comparing the TRPO and CPO versions of the different models also shows the potential problems of constraint-based methods. Initially, the cost that defines the safety constraint (Eq. 6), used in CPO has very high values and the agent learns to satisfy it. This also explains the drop in success rate early during training, which all R-IL models

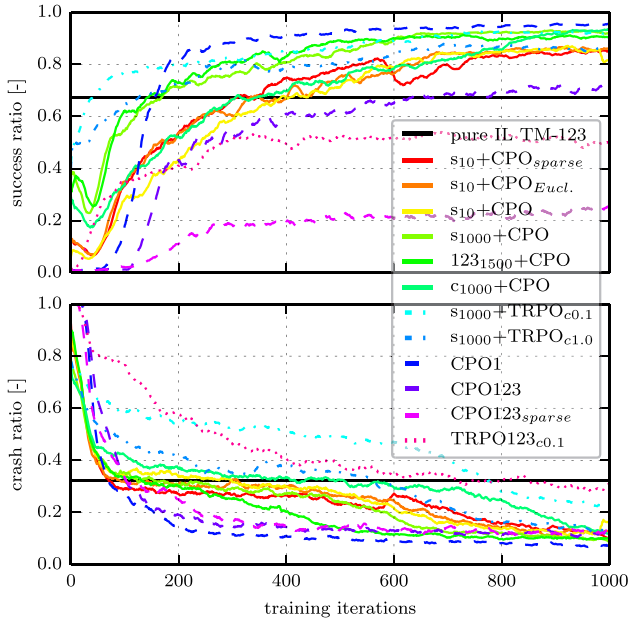

Fig. 4. The evolution of navigation success and crash rates throughout the RL training process of various models. The curves indicate the rolling mean of success and crash rates over 20 steps. The models contain pure RL models, pure IL models and R-IL which differ in the amount and complexitiy of pre-training, the reward structure and the RL training procedure. The black line indicates the performance of IL on the training maps (*TM-123*) as a reference. For the RL training, where multiple runs were conducted, only the best one is shown.

trained with CPO have in common. Therefore, in this phase, the agent learns to avoid crashes and unlearns the behavior of reaching the target, which is also supported by the crash rate curves. Both models (with high and low cost of collision) trained with TRPO do not show this behavior as no constraint needs to be satisfied. Therefore TRPO allows for more "risky" exploration initially. This further motivates to use pre-training when using constraint-based RL, as it provides enough intuition to reach the target while the agent can learn how to satisfy the safety constraint. This would be hard otherwise, as exploration through Gaussian perturbation of a nominal motion command is inherently local in the policy space. The difference becomes even more pronounced for the simpler reward structures, such as sparse target reward. While the agent is stuck with a low success rate for CPO123$_{sparse}$ and mostly learns collision avoidance, pre-training with only 10 demonstrations allows the agent to successfully reach the goal in the vast majority of the cases (s_{10}+CPO$_{sparse}$). With pre-training, sparse and full (shortest path) reward reach about the same final performance.

2) Problem of Fixed Penalty Methods: While, e.g., s_{1000}+CPO reaches a high final success rate and a low crash rate, s_{1000}+TRPO$_{c0.1}$ reaches similar success rates, yet struggles with significantly more crashes. On the other side, s_{1000}+TRPO$_{c1.0}$ reaches a similar crash rate yet does not achieve the same final success rate. This difficulty of fixed penalty parameter tuning was already raised in [8].

3) Final Performance is Affected by the Initial Starting State: Models initialized using more complex maps and/or more trajectories not only perform better but also learn faster. Even a very small amount of demonstrations can significantly improve the overall performance. The R-IL models reach the final performance of CPO123 after less than one fifth of the iterations ($\approx$200) as pre-training provides a good initial policy and makes the stochastic exploration more target-aimed. This confirms our initial hypothesis that the prior IL can significantly reduce the training time in RL applications.

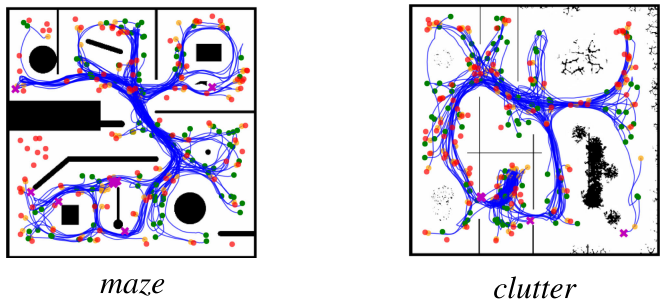

Fig. 5. Evaluation runs between 100 randomly sampled start and target positions on the two unknown test maps (both 10 m $\times$ 10 m). The model used for visualization is c_{1000}+CPO. The trajectories are shown in *blue*, the starting positions in *green*, the set targets in *red*, the trajectory end points in *yellow* and crashes as *magenta* crosses.

C. Simulation Results

In the following, the performance of the navigation policies is analyzed when deployed in unseen environments in simulation. We constructed two 10 m $\times$ 10 m evaluation maps as shown in Fig. 5: (i) A test *maze* and (ii) an environment with thin walls and *clutter*. Then, we conducted the following experiment: 100 random start and target positions were sampled for each of the two environments and consistently used for the evaluation of all models. Possible outcomes for each run are a *success*, a *timeout* or a *crash*. The timeout is triggered, if the target cannot be reached within 5 min. This time would allow the robot to travel 60 m with an average speed of 0.2 ms^{-1} and should suffice to reach the target on a 10 m $\times$ 10 m map. Each episode is aborted after a collision. The resulting trajectories of the evaluation with model c_{1000}+CPO on both maps are visualized in Fig. 5.

Based on the 200 evaluation trajectories per model, Fig. 6 presents the resulting statistics. For comparison, first, we trained the model presented in [3] in our environments, which in the following will be referred to as the V2R (virtual-to-real) model. Second, we used their policy architecture to train our R-IL policy (pretrained in c_{1000}) in order to test the generalization to other model structures (c_{1000}+CPO$_{V2R}$). The robot's velocity was removed from the inputs (resulting in 12 inputs) as supervised learning approaches (as for pre-training) tend to predict the prior velocity values instead of focussing on the perception [4].

Fig. 6 shows that more reward information during training and more pre-training samples not only benefit the training but also the generalization performance. c_{1000}+CPO, the model with shortest distance reward and *complex* pre-training, shows the best generalization performance to unseen environments (using the model structure shown in Figure 2), with a success rate of 79%. Interestingly, even the model with only sparse reward and 10 demonstration trajectories in the *simple* environment shows similar performance to the fixed collision penalty TRPO methods, which were pre-trained with 1000 samples and use the full reward. Both R-IL TRPO methods show a lower success rate

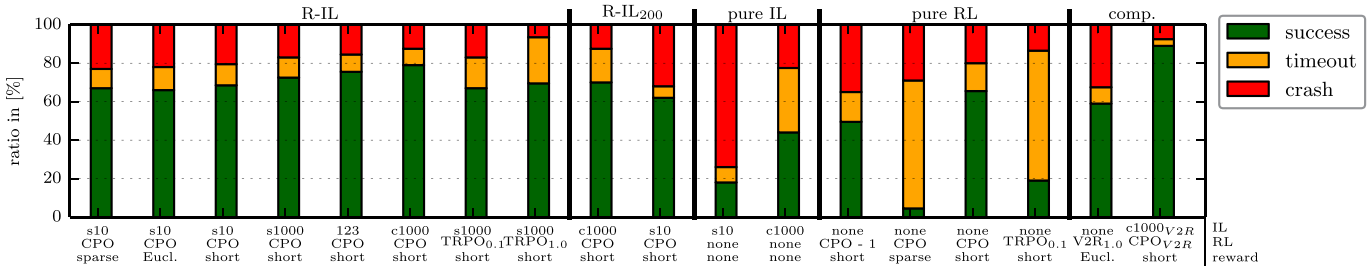

Fig. 6. Evaluation results of 200 trajectories on the previously unseen test maps (100 each) as shown in Fig. 5. The outcome of each trajectory can be a success, a timeout (not reaching the target after 5 min), or a crash. The models are split in five categories: R-IL, where IL is combined with 1000 RL iterations; R-IL$_{200}$; with 200 RL iterations only; pure IL; pure RL and the *comp.* approaches for comparison, comprising the method presented in [3] (V2R) and our method based on the model presented in [3] (c_{1000}+CPO$_{V2R}$). More details of the analyzed models can be found in Table I.

than the corresponding CPO model (s_{1000}+CPO), which also shows that encoding both collision avoidance and reaching the target in one reward is inferior to encoding the collision avoidance as a constraint. Furthermore, the R-IL$_{200}$ models show that early stopping of the training (at 200 RL iterations) still leads to similar performance as training pure RL from scratch. Therefore, pre-training allows for a RL training time reduction of around 80% in order to achieve the same performance. CPO1 model, which reached a high success rate during training, does not generalize properly to unseen and more complex environments.

The V2R method [3] (second-to-right bar) shows a similar success rate as the CPO123 model, while the crash rate is about 50% higher although a collision penalty of 1.0 was used. However, it uses the Euclidean distance reward which is a slight disadvantage compared to CPO123. With V2R, the same problems as with other fixed collision penalty methods can be observed, which is the difficult tuning between exploration and collision avoidance. Our approach also generalizes well to other model structures as the one presented in [3], as shown by the rightmost bar of Fig. 6. Using this simpler architecture, the success rate can even be further improved in our test scenarios, which leaves more room for further graph optimization, which is not covered in this letter.

D. Real-World Experiments

Moving to the real world scenarios further shows the generalization capabilities of the models and also their robustness against sensor noise and actuation delays. The models are purely trained in simulation and the real-world test environment (see Fig. 7) is unknown to the agents.

A quantitative analysis of the trajectories is provided in Table II, where the number of crashes, the amount of manual joystick interference and the comparison of the learning-based trajectories compared to the ones taken by the grid-based `move_base` planning module (which uses global map information) are listed. Table II both lists the average and maximum values observed during five runs per model. The human joystick interference was triggered, if no motion command was sent by the autonomous agent for 10 seconds.

The pure RL model tends to be more cautious, which results in a larger factor λt_{MB}, which is the relative time compared to a global planner. The pure IL model collides more often as there is no collision constraint or penalty during training. Also the R-IL

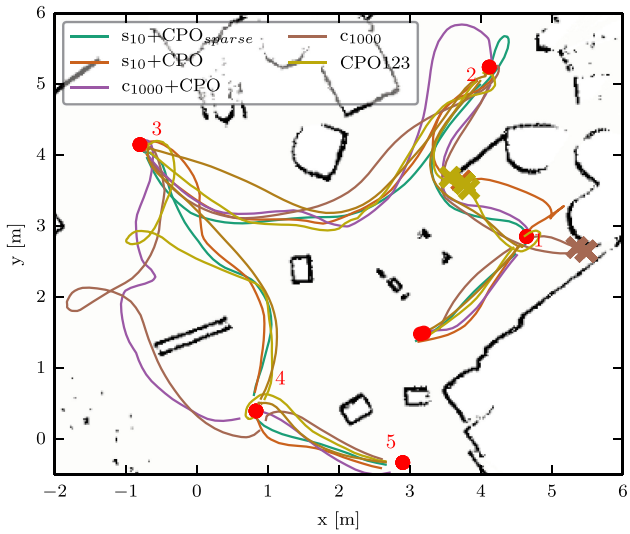

Fig. 7. Trajectories driven with the real robotic platform for a subset of the models analyzed in Fig. 6. *Red dots* depict the numbered target positions, *crosses* in trajectory colors show crashes of the corresponding agents. For clarity reasons, only the first out of 5 runs with each model is shown.

TABLE II
AVERAGE RESULTS (5 RUNS) FROM THE REAL-WORLD EXPERIMENTS, AS SHOWN IN FIG. 7. THE CORRESPONDING MAXIMUM VALUES ARE LISTED IN PARENTHESIS. d_{RC} STANDS FOR THE REMOTE CONTROLLED (JOYSTICK) DISTANCE, λd_{MB} FOR THE RELATIVE DISTANCE COMPARED TO `Move_base` AND λt_{MB} FOR THE RELATIVE TIME COMPARED TO `move_base`

model	#crash	d_{RC} [m]	λd_{MB}	λt_{MB}
s_{10}+CPO	0.8 (2)	0.15 (0.28)	1.17 (1.2)	1.86 (1.95)
c_{1000}+CPO	0.0 (0)	0.0 (0)	1.19 (1.22)	1.04 (1.24)
s_{10}+CPO$_{sparse}$	0.0 (0)	0.01 (0.03)	1.15 (1.19)	1.38 (2.00)
c_{1000}	1.6 (4)	0.05 (0.12)	1.29 (1.39)	1.75 (2.52)
CPO123	0.6 (2.0)	0.08 (0.15)	1.26 (1.29)	2.13 (2.18)

models generalize well to the unseen real-world environment and show similar performance. As expected, c_{1000}+CPO shows the best performance. However, s_{10}+CPO$_{sparse}$ performs surprisingly well. This can be explained by the fact that the sparse reward structure allows for the best generalization performance to unseen environments, since no information about the shortest path to the goal has to be inferred. This is a promising result, as for this model no environment information and reward shaping is required. By combining sparse reward with pre-training and constraint-based RL, even real-world training might be feasible.

V. Conclusion

In this work, we presented a case study for a learning-based approach for map-less target driven navigation. It is based on an end-to-end neural network model which maps from raw sensor measurements and a relative target location to motion commands of a robotic platform and is trained using a combination of imitation (IL) and reinforcement learning (RL). We compare different combinations of prior demonstrations for IL, different RL algorithms and analyze the influence of different reward structures.

Our simulation and real-world experiments show that target-driven demonstrations through IL significantly improve the exploration during RL. The RL training time in R-IL can be reduced by around 80% while still achieving similar final performance in terms of success rate and collision avoidance. While pure RL does achieve the same collision avoidance capabilities as R-IL, there are significant differences in the target reaching success. Pre-training with supervised IL provides a good intuition for more efficient exploration during RL, even if only 10 demonstrations are provided. This becomes even more pronounced when using low information reward structures, like sparse target reward.

Furthermore, our experiments show that constraint-based methods focus on enforcing the collision constraint early during training. This makes exploration harder yet allows for safer training and deployment which becomes important when moving towards real-world applications. Therefore, especially in combination with IL, to achieve safe navigation capabilities, we recommend to enforce collision avoidance by constraint instead of a fixed penalty in the reward signal.

Our trained navigation models are able to reliably navigate in unseen environments, both in simulation and the real world. We do not recommend to replace global planning if a map is available, yet this work shows the current state of what is possible using only local information for navigation scenarios, where no environment map is available.

While in this work, training was purely conducted in simulation, in future work we will investigate how real-world human demonstrations can be leveraged and how this navigation method can be extended to dynamic environments.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[2] 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 *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1527–1533.

[3] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 31–36.

[4] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 739–746.

[5] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[6] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Proc. Intell. Veh. Symp.*, 2017, pp. 204–211.

[7] D. B. Grimes and R. P. Rao, "Learning actions through imitation and exploration: Towards humanoid robots that learn from humans," in *Creating Brain-Like Intelligence*. New York, NY, USA: Springer, 2009, pp. 103–138.

[8] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.

[9] P. Abbeel, D. Dolgov, A. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, Sep. 2008, pp. 1083–1090.

[10] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 2096–2101.

[11] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *Int. J. Robot. Res.*, vol. 35, no. 11, pp. 1289–1307, 2016.

[12] M. Wulfmeier, D. Z. Wang, and I. Posner, "Watch this: Scalable cost-function learning for path planning in urban environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2089–2095.

[13] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2722–2730.

[14] D. K. Kim and T. Chen, "Deep neural network for real-time autonomous indoor navigation," arXiv:1511.04668, 2015.

[15] J. Sergeant, N. Sünderhauf, M. Milford, and B. Upcroft, "Multimodal deep autoencoders for control of a mobile robot," in *Proc. Australas. Conf. Robot. Automat.*, 2015, pp. 1–10.

[16] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[17] S. Ross *et al.*, "Learning monocular reactive UAV control in cluttered natural environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 1765–1772.

[18] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4565–4573.

[19] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially-compliant navigation through raw depth inputs with generative adversarial imitation learning," presented at ICRA, 2018.

[20] B. Bischoff, D. Nguyen-Tuong, I.-H. Lee, F. Streichert, and A. Knoll, "Hierarchical reinforcement learning for robot navigation," in *Proc. 11th Int. Conf. Intell. Eng. Syst.*, 2013, pp. 149–153.

[21] B. Zuo, J. Chen, L. Wang, and Y. Wang, "A reinforcement learning based robotic navigation system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, pp. 3452–3457, 2014.

[22] P. Mirowski *et al.*, "Learning to navigate in complex environments," arXiv preprint arXiv:1611.03673, 2016.

[23] J. Bruce, N. Sünderhauf, P. W. Mirowski, R. Hadsell, and M. Milford, "One-shot reinforcement learning for robot navigation with interactive replay," *CoRR*, vol. abs/1711.10137, 2017.

[24] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2371–2378.

[25] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3357–3364.

[26] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.

[27] B. Balaguer and S. Carpin, "Combining imitation and reinforcement learning to fold deformable planar objects," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 1405–1412.

[28] Y. Zhu *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," in *Proc. of Robotics: Science and Syst.*, 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p09.pdf

[29] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*. New York, NY, USA: Springer, 1992, pp. 5–32.

[30] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[31] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 908–918.

[32] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, 2009.

[33] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intell.*, vol. 2, no. 2-4, pp. 189–208, 2008.