Learning Socially Normative Robot Navigation Behaviors with Bayesian Inverse Reinforcement Learning

Billy Okal Kai O. Arras

Abstract— **Mobile robots that navigate in populated environments require the capacity to move efficiently, safely and in human-friendly ways. In this paper, we address this task using a learning approach that enables a mobile robot to acquire navigation behaviors from demonstrations of socially normative human behavior. In the past, such approaches have been typically used to learn only simple behaviors under relatively controlled conditions using rigid representations or with methods that scale poorly to large domains. We thus develop a flexible graph-based representation able to capture relevant task structure and extend Bayesian inverse reinforcement learning to use sampled trajectories from this representation. In experiments with a real robot and a large-scale pedestrian simulator, we are able to show that the approach enables a robot to learn complex navigation behaviors of varying degrees of social normativeness using the same set of simple features.**

I. Introduction

Mobile robots are increasingly being deployed in human populated spaces such as airports, shopping malls, or warehouses for use-cases such as guidance, information provision, or transportation. Such robots are required to be socially normative with respect to the human occupants [1], for example by respecting personal spaces, people's arrangements in groups or by generating legible motion behavior (see Fig. 1 for an example). This is in addition to optimizing standard navigation metrics such as shortest paths, minimal energy or time to goal. Thus, in the design of robot behavior, we need to unify the objectives of being socially compliant, optimizing subjective human comfort metrics, and being efficient by optimizing objective task metrics. "Social navigation" is an increasingly active research area which deals with this combination by considering people as more than dynamic obstacles, but rather as rational agents with intentions, attributes, preferences and social relations. This comprises advanced perception capabilities of the robot in terms of human detection, tracking and analysis but also the questions of how to represent and acquire such behaviors. Manual design approaches using models like Proxemics [2] have proven difficult and generalize poorly due to model specificity as demonstrated by [3], [4]. Alternatively, the problem can be posed as being data-driven using learning techniques. In particular Markov decision process (MDP) representations in which part(s) of the model such as the reward function is learned from data using Inverse Reinforcement Learning (IRL) is a promising approach as shown in [5], [6], [7], [8] among others.

Both authors are with the Social Robotics Lab, Univ. of Freiburg. Kai Arras is also with Bosch Corporate Research, Germany, {okal, arras}@cs.uni-freiburg.de. This work has been partly supported by the European Commission under contract number FP7-ICT-600877 (SPENCER).

Fig. 1. Socially normative behavior example. Three persons are engaged into a photography activity and the robot should not drive through the group even though there is enough space and the path would be shorter and smoother. The corresponding costmap from the learned model is shown on the right, with darker cells having higher costs. The strength of the social relations between individuals are indicated in percent.

In this work, we take the latter approach and present a new efficient way to model socially normative behavior using MDPs and Bayesian IRL. In doing so, we focus on spatial robot motion behaviors (as opposed to the large variety of possible human behaviors) and assume that the relevant factors that govern motion behavior can be distinguished by features of the robot environment and occupants therein such as distances or relative headings or velocities. We also assume that the resulting robot behaviors can be assessed by examining paths executed by a robot in relations to the aforementioned features. However, by using MDPs, we face the challenges of inflexible representations in large domains, and consequently, poor scalability of behavior learning algorithms. We thus develop a graph-based representation which is flexible and allows integration of task-specific constraints directly into the MDP representation. We then extend an existing Bayesian IRL algorithm to realise a scalable variant that takes advantage of the new representation. The resulting setup enables us to address the questions of how a robot's efficiency is affected by social normativeness and how a robot can learn different complex navigation behaviors of varying degrees of social normativeness with the same set of basic features. To study this, we consider three scenarios:

1) Navigation in medium density crowds typical for e.g. lobbies, shopping malls or transit waiting areas.
2) Navigation in high density crowds with distinct and persistent movement directions such as in narrow hallways with opposing flows.
3) Navigation in densly crowded and chaotic scenes such

as large hallway intersections or meeting points.

The rest of the paper is organised as follows: in II we discuss related work, in III where we present the details of our approach. In IV and V we describe our experiments and discuss the results. Finally, conclusions are given in VI.

II. Related Work

Existing work that strives to achieve socially normative navigation can be divided into two main categories depending on whether the behaviors are learned or manually designed. Those using learned behaviors include [5], [7], [8], in which Henry *et al.* [5] and Vasquez *et al.* [7] address the problem of learning to navigate efficiently in crowds of simulated pedestrians. Kim and Pineau [8] consider small groups of people and carry out experiments with a robotic wheelchair. These approaches also use an MDP formalism for learning and discretize state and action spaces, in particular using "grid world" like representation of state and action spaces, which are popular in reinforcement learning literature [9]. However, such discretisation is not efficient for robots operating in large spaces because; (i) the number of states gets very large, (ii) actions remain highly constrained to either 4- or 8-neighborhoods which ignore typical constraints on mobile robots such as non-holonomy.

Other works using manually designed behavior or not based on IRL, include Sisbot *et al.* [10] who developed a human-aware motion planner by using Gaussians to model the personal space of individuals. Svenstrup *et al.* [11] used dynamic potential fields and RRT to plan trajectories around multiple people but without considering social relations between them. Lu *et al.* [12] developed a method to tune layered cost-maps for social navigation, we however want an automatic discovery of such cost-maps in continuous spaces.

On scalable learning algorithms, Neumann *et al.* [13] extended the work of Guestrin and Ormoneit [14] to build a graph-based representation for continuous reinforcement learning tasks. We build upon this work by adding improved exploration and then leverage the resulting flexible representation to also improve upon Bayesian IRL algorithm of Ramachandran and Amir [15]. To achieve this, we also draw from earlier work of Ng and Russell [16] in using IRL in infinite state spaces via sampled trajectories.

III. Method

An MDP is denoted by $\langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$ where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces respectively. $T(s, a, s') = p(s' \mid s, a)$ is the transition function read as the probability of going to state s' by taking action a in state s, $\gamma \in [0, 1)$ is a discount factor and $r : \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$ is the reward function where $r(s, a)$ is the reward obtained by taking a in s. A policy $\pi : \mathcal{S} \longmapsto \mathcal{A}$ specifies actions in every state and gives rise to two additional quantities, a **value** function,

$$V^{\pi}(s) = r(s, \pi(s)) + \mathbb{E}_{s' \sim T(s, \pi(s), \cdot)} \left[V^{\pi}(s') \right] \tag{1}$$

and an action-value function (also called *Q-function*),

$$Q^{\pi}(s, a) = r(s, a) + \mathbb{E}_{s' \sim T(s, a, \cdot)} \left[V^{\pi}(s') \right]. \tag{2}$$

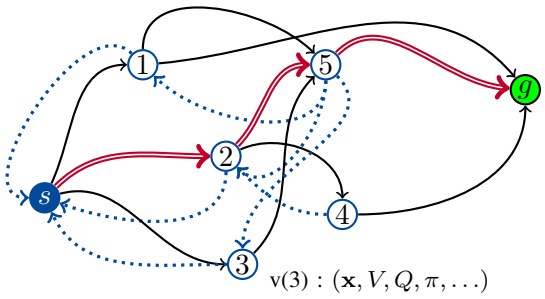

Fig. 2. Conceptual controller graph. s and g indicate start and goal states, the policy is shown in red double line. Example vertex labels are also shown. Reverse edges are shown with dotted lines.

A key task in reinforcement learning is to find an optimal policy π^* that maximizes $V^{\pi}(\cdot)$ for all states. Such an optimal policy is characterized by the Bellman optimality criteria (see [9] for details) as $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s, a)$. We interpret the reward function as the "intrinsic representation of behavior" and the policy execution traces (state-action pairs) to be the realization of such behaviors.

In this work, the reward function is a priori unknown and state and action spaces are very large i. e. $\mathcal{S} \triangleq \mathbb{R}^n$ while $\mathcal{A} \triangleq \mathcal{S}^{\mathcal{S}} = \{g \,|\, g : \mathcal{S} \longmapsto \mathcal{S}\}$. Additionally, we assume robots to be motion-constrained e. g. by non-holonomic constraints which make that not all actions in $\mathcal{A}$ can be executed. We address these challenges in Sec. III-A and III-B.

A. Representation Learning

In this section, we seek an efficient and flexible representation that is able to capture relevant structure of our navigation tasks. We build upon our previous work [17] from which we highlight the key aspects here, starting with the following definition,

Definition 1 (Controller Graph (CG))**.** A weighted labeled graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \boldsymbol{W} \rangle$ with vertices $\mathcal{V}$, edges $\mathcal{E}$ and a transition matrix $\boldsymbol{W}$, such that $\mathcal{V} \subset \mathcal{S}$ and $\mathcal{E} = \{(v_i, v_j)_a \,|\, w_{i,j} > 0, \forall v_i, v_j \in \mathcal{V}, a \in \mathcal{A}\}$.

Concretely, vertices are "macro states" containing state vectors $\boldsymbol{x} \in \mathbb{R}^n$ and edges $(v_i, v_j)_a$ are short trajectories $\boldsymbol{x}_{i:j}$ between vertices i and j, called **local controllers**. Additional action information can be stored in local controllers such as angles or speeds. As such, they provide task decomposition and can incorporate task-specific constraints like speed limits into the representation. For instance, one can use predefined sets of motion primitives as local controllers. In this work, we use the POSQ steer function by [18] which is well-suited for differential-drive mobile robots. Policies in a CG are graph walks (or paths) from start to goal states (see Fig. 2). CGs can thus be seen as discrete generative models of continuous domains as pointed out by [19]. Because the edge lengths vary, the resulting graph represents a semi-Markov decision process (SMDP). Ergo, the rewards accumulated

1: **procedure** BUILDCG($D, s, g, N_{\max}, \delta$)
2: Generate k random states $\{s_1, \ldots, s_k\}$
3: $\mathcal{G} \leftarrow$ INITGRAPH($s, g, \{s_1, \ldots, s_k\}$)
4: $\xi_{best} \leftarrow$ FINDPOLICY($\mathcal{G}, s, g$)
5: **while** $|\mathcal{V}| \leq N_{max}$ **do**
6: $\mathcal{V}_b \leftarrow \{v \in \mathcal{V} | v \in \xi_{best}\}$ // Exploitation set
7: **if** $p > \delta$ **then** // $p \sim$ Uniform$(0, 1)$
8: Expand from $\mathcal{V}_b$
9: **else**
10: Expand from $\mathcal{V} \setminus \mathcal{V}_b$ // Exploration
11: **end if**
12: $\xi_{best} \leftarrow$ FINDPOLICY($\mathcal{G}, s, g$)
13: **end while**
14: **return** $\mathcal{G}$
15: **end procedure**

Algorithm 1: Building Controller Graphs

while traversing an edge e are given by (3).

$$r(e) \triangleq \int_0^{|e|} \gamma^t r(s_t, a_t)\, dt \tag{3}$$

where $r(s_t, a_t)$ represents instantaneous reward and $|\cdot|$ denotes length operator. Local controllers can thus be interpreted as *options* in hierarchical reinforcement learning [20]. We can therefore define values of vertices as follows,

$$V^\pi(v_i) = r(e) + \mathbb{E}_{v_j \sim T(v_i, \pi(v_i), \cdot)} \left[V^\pi(v_j) \right] \tag{4}$$

where $e = (v_i, v_j)_a$, and $T(v_i, \pi(v_i), \cdot) = \boldsymbol{W}_i$ and $\pi(v_i)$ selects one edge from the available outgoing edges at v_i. The Q function is analogously defined.

The graph is initialized using uniformly sampled states from $\mathcal{S}$ (lines 2-3) in Alg. 1, edges are added by running local controllers from each vertex. Alternatively, samples from expert demonstration trajectories can be used for initialization. A CG is iteratively built online using heuristics to trade off exploration and exploitation as shown in Alg. 1. *Exploitation* involves sampling new states around those with high values hence likely to be on the path of the optimal trajectory to the goal (policy), while *exploration* encourages coverage of $\mathcal{S}$ so that new regions are discovered by sampling nodes around those with few neighbors over some radius. Expanding from a set (lines 8 and 10 in Alg. 1) involves randomly selecting states from the set, executing the local controller for a limited time then adding the state at the end of the local controller to the graph. Vertices are labeled with data such as $V^\pi(\cdot)$, $\pi(\cdot)$ and $Q^\pi(\cdot, \cdot)$ which are updated using policy iteration at every iteration of adding new states. Altogether, the resulting CG usually contains a small number of vertices, for example a $10m \times 10m$ room can be represented by only 140 vertices in our context as shown in Fig. 3, while a standard grid representation with $10cm$ resolution would yield 10^4 cells, and much limited actions from cell to cell.

B. Learning Socially Normative Behaviors

We assume that M expert trajectories are given as $\Xi_E = \{\xi_1, \ldots, \xi_M\}$, where each $\xi_m = \{(s_1, a_1), \ldots, (s_{H_m}, a_{H_m})\}$

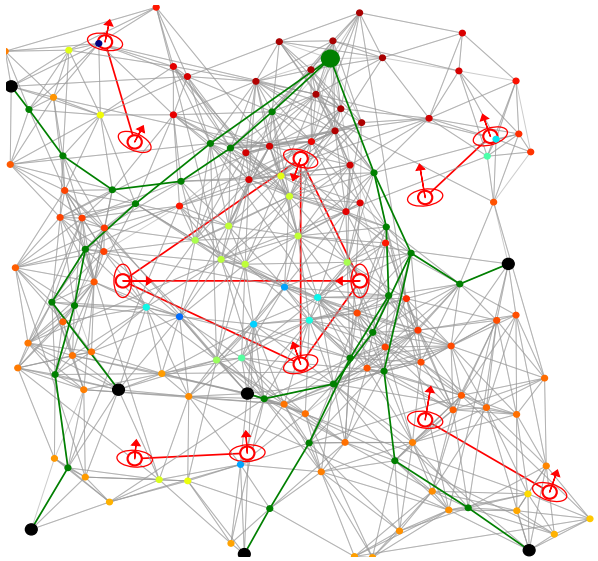

Fig. 3. Example learned graph representation for a simulated $10m \times 10m$ environment. Start states are indicated in black, goal in green, other states are colored using *jet* mapping indicating their values. The policies (trajectories) from the starts are shown in green. Persons in the scene are shown using ellipses with arrows indicating their headings, while pairwise relation between persons are shown using red lines between them. Edges are shown using straight lines for ease of visualization, but in practice they can be arbitrary curved trajectories resulting e.g. from POSQ [18] paths.

for some horizon H_m. We want to recover the reward function underlying the behavior that produced these trajectories. The set of such reward functions has been characterized in [16]. However, defining an appropriate objective function to find the underlying reward is a challenge since the reward is fundamentally unidentifiable [16]. We also want to model uncertainty e. g. from experts that acting sub-optimally during the demonstrations. We thus adopt the Bayesian IRL (BIRL) approach by Ramachandran and Amir [15] and thus seek to find the reward posterior distribution given by,

$$\begin{aligned} p(r \,|\, \Xi) &\propto p(\Xi \,|\, r)\, p(r) \\ &= \frac{p(\Xi \,|\, r)\, p(r)}{\int p(\Xi \,|\, r)\, p(r)\, dr} \end{aligned} \tag{5}$$

where the prior is given by

$$p(r) = \prod_{(s,a) \in \mathcal{S} \times \mathcal{A}} p(r(s,a))\,, \tag{6}$$

assuming *i.i.d.* rewards, and the data likelihood term by

$$p(\Xi \,|\, r) = \prod_{\xi \in \Xi} \prod_{(s,a) \in \xi} \frac{\exp\left(\beta Q^*(s,a,r)\right)}{\sum_{b \in \mathcal{A}} \exp\left(\beta Q^*(s,b,r)\right)} \tag{7}$$

where $Q^*(\cdot, \cdot, r)$ is equivalent to $Q^\pi(\cdot, \cdot)$ in which π is obtained using reward function r. β is an expert optimality parameter interpreted as our confidence on the demonstrator picking optimal actions. The posterior in (5) is hard to compute due the intractable normalizing factor, thus [15] uses PolicyWalk, a Markov chain Monte Carlo (MCMC) algorithm to approximate the reward posterior from samples which only requires ratios of the posterior. However, such MCMC approaches typically involves solving an MDP to

evaluate every sample (including rejected ones), which is costly for large MDPs and undermines scalability.

Ng and Russell [16] developed an iterative procedure that uses randomly sampled trajectories to recover rewards in very large (possibly infinite) spaces. In their work, they used linear programming (LP) to search for a new reward in a restricted space of policies at every iteration. We extend their approach so as to develop Trajectory BIRL (TBIRL) algorithm. We represent the reward function as a linear combination of d features $f(s,a) : \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$ so that $r(s,a) \triangleq \boldsymbol{w}^T \boldsymbol{f} = \sum_{i=1}^{d} w_i f_i$ where $\boldsymbol{w}$ is a vector of weights which control importance of each feature. The values of expert policy and candidate policies need to be computed at every iteration. The expert policy value V^{π_E} can be empirically estimated by,

$$\hat{V}^{\pi_E} = \sum_{\xi \in \Xi_E} \sum_{(s,a) \in \xi, t=0}^{|\xi|} \gamma^t \sum_{i=1}^{d} w_i f_i(s,a) \tag{8}$$

while the values of candidate policies can be estimated in a similar way using sampled trajectories or by dynamic programming [21]. We use trajectories sampled from a CG which are already better suited for estimating values of policies as they capture relevant task structure. By employing a Bayesian formulation we are able to explicitly model uncertainty in expert demonstrations such as optimality. Such a Bayesian formulation also enables us to examine the distribution of reward functions to better understand the resulting behaviors. We therefore define a new data likelihood for BIRL that directly uses sampled trajectories as

$$p(\Xi_E, \Xi_G \,|\, r) = \prod_{\xi^e \in \Xi_E} \frac{\exp\left(\beta \zeta(\xi^e)\right)}{\exp\left(\beta \zeta(\xi^e)\right) + \sum_{i=1}^{k} \exp\left(\beta \zeta(\xi_i^g)\right)} \tag{9}$$

where $\zeta(\xi, r) = \sum_{(s,a) \in \xi} Q^{\pi}(s,a)$, with π obtained from r. Further $\xi_i^g \in \Xi_G$ is a trajectory sampled using a candidate policy at iteration i, while k is the current iteration. Intuitively, as the reward improves, we are able to generate sample trajectories of increasing similarity to the ones from the expert. The new likelihood term in (9) is similar to the one in (7) when each trajectory is interpreted as a single action from start to goal. The negative log-likelihood of (9) is convex, therefore we can pick a convex prior so that the full posterior is convex, and then use tools from optimization to quickly find a mode of the distribution for our reward estimate. Alg. 2 shows the details of our TBIRL algorithm.

C. Reward Features

As already mentioned in Sec. III we model the reward function as a linear combination of features of the state and action spaces. The choice of these features is vital as they have the task to fully encode the relevant influences that govern the desired behaviors (as also found in our past work [7]). Here, we use a set of simple domain-specific features based on insights from past work in social navigation. They include *distance to entities in the scene* (persons, obstacles, pair-wise relations within social groups represented as lines), making three features, and *relative heading of persons* with respect to the robot's goal, *heading deviation from goal* and *distance to goal*, making another three features. Unlike [5], [8] we do not discretize the real-valued feature responses. Note that more features such as high-level human attribute information of age, gender, visual appearance or head orientation can be readily employed in such a learning framework.

procedure TBIRL(Ξ_E, ϵ, K)
  Pick π_0 randomly
  $\Xi_G \leftarrow \{\xi_0^g\}$ // Generate ξ_0^g using π_0
  for $k \leftarrow 1, K$ **do**
    $r_k \leftarrow \arg\max_r p(\Xi_E, \Xi_G \,|\, r)\, p(r)$ // (6), (9)
    Estimate $\hat{V}^{\pi_E}, \hat{V}^{\pi_E}, \pi_k$ using r_k
    if $||\hat{V}^{\pi_E} - \hat{V}^{\pi_E}||_p \leq \epsilon$ **then**
      return r_k
    else
      $\Xi_G \leftarrow \Xi_G \cup \{\xi_k^g\}$ // Generate ξ_k^g using π_k
    end if
  end for
  return r_k
end procedure

Algorithm 2: BIRL using sampled trajectories

D. Navigation using Learned Rewards

Once we have learned a reward function for a navigation task, we use the function to define costs of social normativeness for planning. This can be done either by dynamic costmaps [12] for planning or by injecting the costs at an early stage into an objective function of a cost-optimal planner. When using RRT-based planners, for example, the learned rewards can be used to evaluate extensions while growing the tree, hence producing a socially normative tree.

IV. Experiments

The purpose of the experiments is twofold. First, to demonstrate that, using the proposed learning approach, different complex robot behaviors of varying degrees of social normativeness can be learned with the same set of basic features, and second, that the model generalizes with respect to various behavioral goals. We carry out quantitative experiments in simulation and qualitative experiments on a real robot. The three simulated environments, as mentioned in Sec. I, are the lobby scenario ($30m \times 30m$, medium density non-moving crowd), the hallway scenario ($120m \times 40m$, high density crowd with persistant flows), and the intersection scenario ($40m \times 40m$, densly crowded, chaotic), see Fig. IV. The sizes were chosen to reflect real-world applications which serve as practical motivation for this work. We present 10 expert demonstrations of each behavior by manually driving the robot through the scenes using a joystick.

In the lobby scenario, we are interested in replicating the complex human navigation behaviors observed in parties or general gathering. We learn three navigation behaviors that account for varying degrees of social compliance:

1) *Polite* – always avoid breaking pair-wise social relations and intruding people's intimate or personal spaces

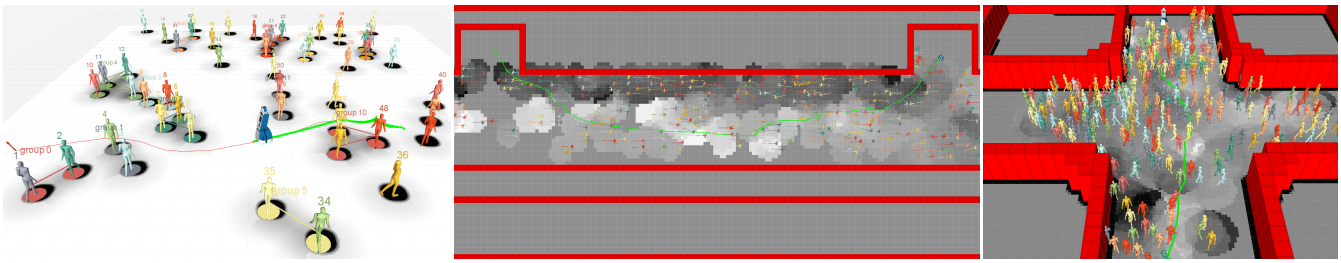

Fig. 4. The three scenarios used in the experiments. Pair-wise social relations are visualized with lines between agents, the path planned by the robot is shown in red while its executed trajectory is shown in green. Red cells indicate walls in the environments. The costmaps which represent the learned behaviors are shown in gray, with dark regions indicating higher cost. **Left:** lobby-like scene with 50 persons and 18 pair-wise relations, **center:** hallway scene with 400 persons moving in two distinct "flows" in which the robot correctly avoids the opposing flow at the expense of taking a longer path, **right:** intersection scene with 280 persons crossing each others' paths. The robot performs slipstream navigation by following persons with similar bearing.

on the way to the goal. We make the assumption that it is impolite to split up groups or "creep" on people.
2) *Sociable* – avoid pair-wise social relations and get as close to people as possible. This is an example of a robot serving drinks at a party or handing out pamphlets in a shopping mall.
3) *Rude* – getting to the goal as fast as possible ignoring social relations and personal spaces.

In the hallway and intersection scenarios, we aim at an understanding of how social compliance affects efficiency in terms of task metrics. To this end, we demonstrate two behaviors for the hallway scenario,

1) *Merge* – joining the flow of people heading towards the robot goal, i. e. "gliding in the flow towards goal"
2) *Nomerge* – "wading" through the flows using the shortest path to goal,

and two behaviors for the intersection scenario,

1) *Slipstream* – following closely behind a person or a group of persons on the way to the goal i. e. using other agents that help to "clear the road".
2) *Direct* – again "wading" through the crowd using the shortest path to goal.

For simulation, we use an open-source[1] pedestrian simulator [22] in which agents are guided by potentials governed by the social force model [23]. The simulator provides realistic walking pedestrians as well as social groupings using one unified model. For both simulation and the experiments with the real robot we use the learned behaviors to generate costmaps, on which we plan global paths using weighted A^* then drive the robot locally using an elastic band local planner. We use the `move_base`[2] framework from `ROS` [24] to ensure we have the same configurations in simulation and on the robot, with only differences in low-level driving control. As such we implement new costmap layers akin to [12] in the `move_base` framework which then represent the new behaviors. This way the behaviors can be intepreted as a policies for motion planning. Using the standard A^* planner from `ROS` [24] eliminates experimental variation, but advanced planning algorithms such as RRT can also be used in practice.

[1] `https://github.com/srl-freiburg/pedsim_ros`
[2] `http://wiki.ros.org/move_base`

A. Experimental Metrics

For evaluation, we use the set of metrics given in [7] and add a mission completion metric. They include objective criteria for measuring how efficient the robot behaves in a standard path planning sense, and subjective criteria that measure how socially normative a robot behavior is:

- *Objective metrics* – *(i)* path length, *(ii)* cumulative heading changes (CHC), estimating path smoothness or how much a behavior causes the robot to maneuver, *(iii)* mission completion, the number of times the robot successfully reached the goal over multiple runs (mission is aborted if the robot is "stuck" for more than 10 mins), *(iv)* time to reach goal.
- *Subjective metrics* – *(i)* number of crossed pair-wise social relations between persons, *(ii)* number of intrusions into circular Proxemics [2] spaces of radii, social space $(1.2m, 3.6m]$, personal space $(0.45, 1.2m]$, and intimate space $[0, 0.45m]$.

The goal of the subjective metrics is to emulate the perception of social normativeness by users. Ideally, empirical studies are conducted to quantify such metrics but given our simulation setup with very large crowds, the number of conditions and runs, such studies are beyond the scope of this paper.

B. Parameters

The algorithms presented here include a number of parameters. In building controller graphs, we use $\delta = 0.4$ and $N_{\text{max}} = 440$, which we empirically found satisfying. For learning rewards, we used $K = 20$ and $\epsilon = 0.01$. Other parameters such as Proxemics distances used in the metrics are acquired from accompanying literature [1].

V. Results

We first give quantitative and qualitative results from the simulation experiments. All metrics are given as means and 95% intervals from 15 different runs with different start and goal positions of the robot and different start positions of the simulated pedestrians. For the hallway and intersection scenarios, individual walking speeds are randomized as well.

Fig. 5. Navigation results in the lobby scenario using polite, sociable and rude behaviors showing the two objective metrics path length and path smoothness in terms of numbers of heading changes along the path. Intervals show 95% confidence from 15 different runs.

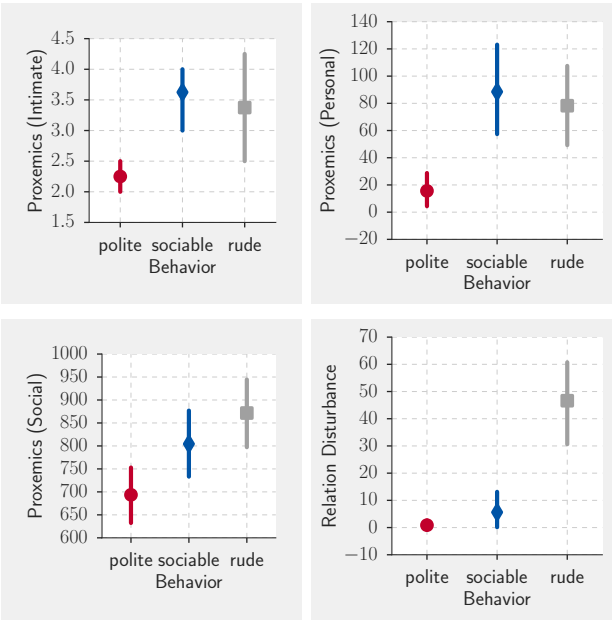

Fig. 6. Navigation results in the lobby scenario using polite, sociable and rude behaviors showing the subjective metrics.

A. Simulation Experiments

Figs. 5 and 6 give the objective and subjective metrics for the lobby scenario. The objective metrics indicate that socially normativeness leads to a penalty in having to take longer and less smooth paths with more maneuvers. For the sociable behavior, we observe even more maneuvers which is expected for a robot serving cocktails at a party while respecting relations between persons. Also as expected, the rude behavior results in the most efficient paths. The results in the subjective metrics reinforce the message from the objective counterparts in that social normativeness results in minimal number of intrusions into people's spaces while being rude leads to the opposite. The sociable behavior gets the robot close to people (large numbers of intrusions into personal and intimite spaces in Fig. 6), while minimizing the number of social relation disturbances. People in goups are correctly approached without breaking social ties at the expensive of more maneuvers as observed in the decreased smoothness metric in Fig. 5.

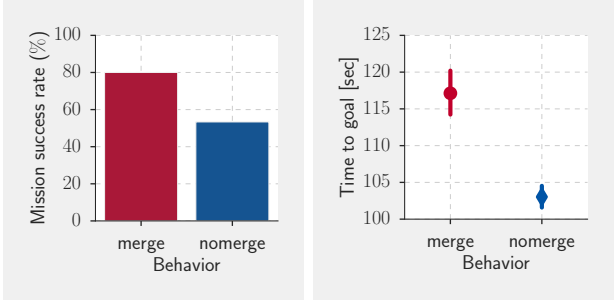

Fig. 7. Navigation in the hallway scenario. **Left:** fraction of times robot successfully reached the goal. **Right:** average mission times

For the hallway navigation task, we get the results shown in Fig. 7. They indicate that being socially normative in flows i. e. merging with a flow into the direction of one's goal, improves reliability in terms of mission completions at a small price of longer mission times due to the longer paths that the robot has to take to avoid the opposing flow. Similarly, for navigation in densely crowded intersections (Fig. 8), the robot improves its efficiency by identifying and following persons with similar goal bearings (being in the slipstream of a person or a group of persons), as manifested in improved mission completion. The time to arrive at the goal is again longer for being socially normative, i. e. by following a person's slipstream the robot makes more turns as shown in Fig. IV(right).

These results demonstrate the generalization ability of the learning approach and the ability of its features to encode the relevant influences for learning complex navigation behaviors of varying social normativeness. They also show that the robot is able to generate behavior that involves temporal credit assignment (deciding to take a longer path) in order to achieve a goal that was not specified in its demonstrated objective (mission completion metric).

B. Experiments with the Robot

We also perform experiments using our robot Daryl, a differential drive robot with a mildly humanized appearance for human-robot interaction research. In the experiments, the robot runs a laser-based group tracker [25] to obtain social grouping probabilities between individuals. In case of grouping probabilities above 0.5, we assume a pair-

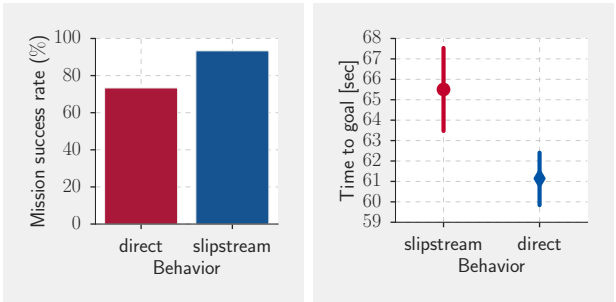

Fig. 8. Navigation in the intersection scenario. **Left:** fraction of times robot successfully reached the goal. **Right:** average mission times.

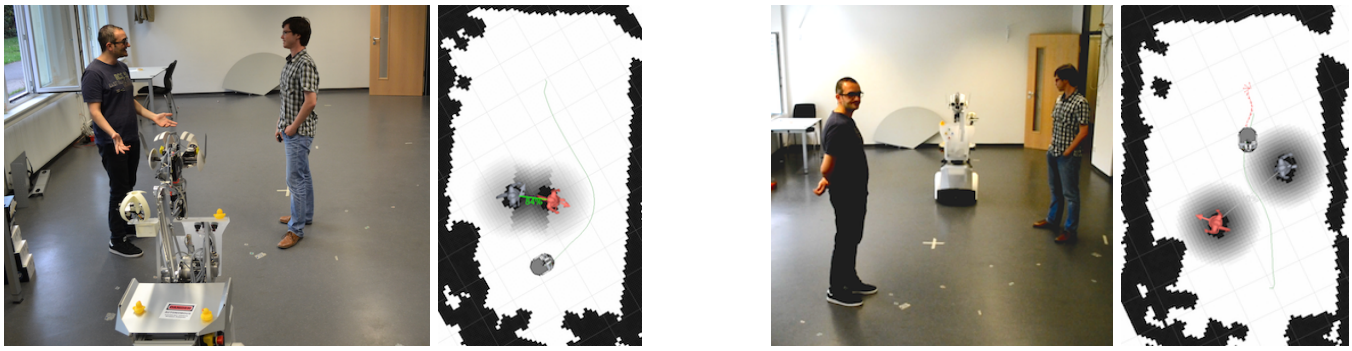

Fig. 9. Experiments with the robot Daryl showing a socially normative behavior in a group of people. Setup and resulting costmaps are shown side-by-side, darker cells are higher cost, the robot path is shown in green. **Left:** two persons engaged in a conversation, the robot drives around them for social normativeness. **Right:** two persons without social relation, the robot drives in between them while respecting their personal spaces.

wise social relation. We again use the learned behaviors to generate costmaps on which we plan global paths with A^* and drive the robot using an elastic band local planner from ROS. Fig. 9 shows two example behaviors realized on the robot. The robot avoids splitting a group of people engaged in a conversation for which the tracker has estimated a high-probability pair-wise social relation. Without such an estimate, the robot drives in between the persons still respecting their personal spaces. Fig. 1 is an example with people engaged into a photography activity.

VI. Conclusions

We presented a learning approach to acquire socially normative robot navigation behavior from demonstrations using BIRL. To obtain a practical algorithm for large domains, we proposed a graph-based representation of the continuous environments on which we learn navigation behaviors using a new extension of BIRL that relies on sets of sampled trajectories and is applicable to large scale domains. We conducted experiments with a real robot and three different simulated environments with moving and non-moving crowds of pedestrians of varying densities. We were able to show that the approach is applicable to large domains and generalizes well from few expert demonstrations. In particular, we showed that the approach enables a robot to learn complex navigation behaviors of varying degrees of social normativeness using the same basic set of features.

References

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, June 2013.

[2] E. T. Hall, R. L. Birdwhistell, B. Bock, P. Bohannan, A. R. Diebold Jr, M. Durbin, M. S. Edmonson, J. Fischer, D. Hymes, S. T. Kimball, *et al.*, "Proxemics," *Current anthropology*, pp. 83–108, 1968.

[3] A. K. Pandey and R. Alami, "A framework for adapting social conventions in a mobile robot motion in human-centered environment," in *Int. Conf. on Advanced Robotics (ICAR)*, 2009.

[4] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *Int. Conf. on Intelligent Robots and Systems*, 2013.

[5] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010.

[6] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[7] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[8] B. Kim and J. Pineau, "Human-like navigation: Socially adaptive path planning in dynamic environments," in *RSS W.shop on Inverse Optimal Control and Learning from Demonstration*, Berlin, Germany, 2013.

[9] A. G. Barto and R. S. Sutton, *Reinforcement learning: An introduction*. MIT press, 1998.

[10] E. Sisbot, L. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Trans. on Robotics and Automation (TRO)*, vol. 23, no. 5, Oct 2007.

[11] M. Svenstrup, T. Bak, and H. Andersen, "Trajectory planning for robots in dynamic human environments," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[12] D. V. Lu, D. B. Allan, and W. D. Smart, "Tuning cost functions for social navigation," in *Social Robotics*. Springer, 2013.

[13] G. Neumann, M. Pfeiffer, and W. Maass, "Efficient continuous-time reinforcement learning with adaptive state graphs," in *European Conf. on Machine Learning (ECML)*, Warsaw, Poland, 2007.

[14] C. Guestrin and D. Ormoneit, "Robust combination of local controllers," in *Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, Seattle, USA, 2001.

[15] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

[16] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning." in *Int. Conf. on Machine Learning*, Haifa, Israel, 2000.

[17] B. Okal, H. Gilbert, and K. O. Arras, "Efficient inverse reinforcement learning using adaptive state-graphs," in *Learning from Demonstration Workshop at Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

[18] L. Palmieri and K. O. Arras, "A novel RRT extend function for efficient and smooth mobile robot motion planning," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[19] J. H. Metzen, "Learning graph-based representations for continuous reinforcement learning domains," in *Machine Learning and Knowledge Discovery in Databases*, 2013, vol. 8188, pp. 81–96.

[20] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 1-2, pp. 41–77, 2003.

[21] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[22] B. Okal and K. O. Arras, "Towards group-level social activity recognition for mobile robots," in *In IROS Assistance and Service Robotics in a Human Environments Workshop*, Chicago, USA, 2014.

[23] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PloS one*, vol. 5, no. 4, 2010.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[25] T. Linder and K. O. Arras, "Multi-model hypothesis tracking of groups of people in RGB-D data," in *IEEE Int. Conf. on Information Fusion (FUSION'14)*, Salamanca, Spain, 2014.