Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison

Dizan Vasquez, Billy Okal, Kai Arras

Inverse Reinforcement Learning Algorithms and Features for Robot Navigation in Crowds: an experimental comparison

Dizan Vasquez Billy Okal Kai O. Arras

***Abstract*— For mobile robots which operate in human populated environments, modeling social interactions is key to understand and reproduce people's behavior. A promising approach to this end is Inverse Reinforcement Learning (IRL) as it allows to model the factors that motivate people's actions instead of the actions themselves. A crucial design choice in IRL is the selection of features that encode the agent's context. In related work, features are typically chosen ad hoc without systematic evaluation of the alternatives and their actual impact on the robot's task. In this paper, we introduce a new software framework to systematically investigate the effect features and learning algorithms used in the literature. We also present results for the task of socially compliant robot navigation in crowds, evaluating two different IRL approaches and several feature sets in large-scale simulations. The results are benchmarked according to a proposed set of objective and subjective performance metrics.**

I. INTRODUCTION

For robots that share a space with humans, the ability to perceive, understand, and act in a socially conform way is a key requirement in many application domains. An area of increasing research activity along this line is socially compliant or socially aware robot motion planning among humans [1]. The goal is to acquire, represent and incorporate domain knowledge about human behavior into the robot's navigation system. Unlike classical robot motion planning, dynamic obstacles are considered as individual rational agents which maintain social relations and follow rules of social behavior. There are two major groups of related works: those that apply models from social psychology and cognitive sciences and those relying on machine learning techniques.

The incorporation of human sciences models [2]–[7] is a promising and rigorous approach but whether these models scale to human-robot relations in uncontrolled conditions is an open research question. Many of these models, including the Proxemics theory [8], the social force model [9] or the back space model [10] used in [3] have been formalized and calibrated for inter-human relations, often in the controlled conditions that are typical for the experimental methodology of human sciences.

Alternatively, learning approaches seek to acquire those models through statistical learning from observational data about humans. This paper focuses on one of the most popular such approaches on recent years: inverse reinforcement learning (IRL). For instance, Ziebart [11], used the maximum-entropy IRL algorithm to learn and predict how people will move in a static environment. Dynamic environments were later addressed in [12], where Henry *et al.* use IRL to have a simulated agent learn to join the flow of pedestrians using density and average flow direction features. Another version of the maximum-entropy algorithm was applied by Kudererer *et al.* [13], to learn a behavior model which is used to predict socially compliant trajectories. More recently, Kim *et al.* [14] have applied a Bayesian approach to IRL for social navigation using depth-augmented optical flow features as input.

Despite the diversity of features, IRL learning algorithms, and system parameters found in the literature, there are no comparative studies –to the best of our knowledge– on the virtues or drawbacks of the different variants. This paper is a first step in this direction, in it we systematically investigate the effect of these choices in large-scale simulations using a set of representative objective and subjective performance metrics. Concretely, we analyze the impact of different features that have been used in the related literature and compare two popular IRL learning algorithms. Finally, we compare the results obtained to those obtained by manual tunning, as well as to a human-based baseline obtained through teleoperation.

Another major motivation that drives this work is a novel application of service robots in the aviation industry. In this scenario, developed by the EU-FP7 project SPENCER in collaboration with airline KLM as end-user[1], we will deploy an autonomous mobile robot at the Schipol Airport in Amsterdam to guide delayed transfer passengers from their gate of arrival to the Schenger barrier and instruct them to use the priority lane.

The paper is structured as follows: section II introduces Markov Decision Processes and Inverse Reinforcement Learning. In section III, we discuss the particular choices that we made for our experiments. The experimental platform is introduced in section IV and the experiment themselves are described in section V. Finally, we present our conclusions and discuss future work in section VI.

II. INVERSE REINFORCEMENT LEARNING

In this section, we briefly describe inverse reinforcement learning (IRL) focusing on the Markov Decision Process-based formulation. We also present two existing approaches to solve this learning problem.

Dizan Vasquez is with the Social Robotics Lab and INRIA Grenoble (dichodaemon@gmail.com).

Billy Okal and Kai. O. Arras are with the Social Robotics Lab, University of Freiburg ((okal, arras)@cs.uni-freiburg.de).

This work has been partly supported by the European Commission under contract number FP7-ICT-600877 (SPENCER).

[1]`http://www.spencer.eu`

A. Markov Decision Processes

Before describing IRL, we need to briefly introduce Markov Decision Processes, a probabilistic planning technique which lies at the heart of the learning problem. A finite Markov Decision Process (MDP) is defined by the following five elements:

- A finite set of N *states* $\mathbb{S} = \{s_1, \cdots, s_N\}$.
- A finite set of K *actions* $\mathbb{A} = \{a_1, \cdots, a_K\}$.
- A *transition probability function* $P(S_t|A_{t-1}S_{t-1})$, where $S_t, S_{t-1} \in \mathbb{S}$ and $A_{t-1} \in \mathbb{A}$, describing the evolution of a previous state S_{t-1} to a new one S_t when executing a given action A_{t-1}.
- A *reward function* (conversely, a *cost function*) $R \colon \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ that depends on a state and an action.
- A *discount factor* $\gamma \in [0, 1)$ for penalizing future rewards.

The classical problem in MDPs assumes that all of the five elements are known and consists of finding an *optimal policy* which gives, for every state, the action that should be selected in order to maximize the expected reward.

B. Inverse Reinforcement Learning

Inverse reinforcement learning [15], deals with the inverse problem of finding the reward from either an existing policy, an action-state sequence, or as is the case in this paper, from a demonstrated, possibly suboptimal, state sequence[2] $D = \{S_1, \cdots, S_T\}$.

Most MDP-based IRL approaches assume that there is a set of M *features* associated with every state, which fully determine the value of the reward function R. Since finding a general form solution for R is very difficult, most approaches assume it to be a linear combination of the features. Thus, for a given state S, the cost[3] can then be expressed as the dot product $\Phi_s \cdot W$ of a *feature vector* $\Phi_S = [\phi_1, \cdots, \phi_M]$ and a *weight vector* $W = [w_1, \cdots, w_M]$. In this case, the IRL problem consists on estimating the values of the weight vector.

Here, we consider two inverse reinforcement learning algorithms, namely Max-margin IRL [16] and Maximum Entropy IRL [17]. They are based on a statistic $F \to \mathbb{R}^M$ which is used to compare the training data with the state sequences that are obtained when applying the optimal policy for a given weight estimate $\hat{W}_j$. This is integrated in an iterative minimization algorithm which is outlined in Alg. 1

The two IRL methods differ essentially in the chosen statistic and in the way of computing the weight estimate W_j. Due to space limitations, we provide only a general description of these differences, the interested reader may refer to the original papers:

- *Max-margin IRL:* uses the feature expectation as statistic. The weight estimation is computed by maximizing the difference (the margin) between all the previously found expected costs and the demonstrated expected cost. The solution is accepted when this margin goes below a given threshold.
- *Maximum Entropy IRL:* the statistic is the feature count or, for non-binary features, the feature sum. It estimates weights with a probabilistic approach which computes path probabilities using an approximate dynamic programming procedure inspired by the forward-backward algorithm for Hidden Markov Models and Conditional Random Fields.

[2]Most approaches, including those considered here, can deal with multiple demonstration sequences.

[3]In the context of motion planning, the problem is often formulated in terms of *costs* which can be seen as negative rewards.

Algorithm 1 General layout of an iterative IRL algorithm.

```
1) Set $j = 1$
2) Propose an initial weight vector estimate $\hat{W}_1$
   (e.g. random values)
3) Compute the optimal policy $\pi_j$ for the current
   weight vector
4) If the statistics for the data and for the optimal policy
   are similar enough (e.g. $\|F(D) - F(\pi_j)\| < \epsilon$) return
   $W = \hat{W}_j$ as the solution
5) Otherwise
   a) Set $j \to j + 1$
   b) Compute $\hat{W}_j$ using an algorithm-dependent method
   c) Jump to step 3
```

C. IRL in dynamic environments

These IRL approaches have been developed for environments where the features do not change over time. This is obviously not the case in dynamic environments and adaptations need to be done –in particular in the policy computation step of the learning algorithm. The most obvious strategy is to re-plan for every time step, updating the features and the cost function accordingly. Although the solution is not longer guaranteed to be optimal, it is often considered to be a good approximation but, unfortunately, this procedure is computationally expensive. This has motivated us to find a trade-off in similar way to [12]: by replanning every ten time steps, we considerably reduce the required resources while still representing the evolution of the environment in a way that we consider acceptable.

III. IRL for Robot Navigation in Crowds

This section presents the features and algorithmic choices that are specific for the task of socially compliant robot navigation among people in crowds.

A. Features

At time t, the state of the robot is defined by a four-dimensional vector $S_t^r = [x_t^r, y_t^r, \dot{x}_t^r, \dot{y}_t^r]$, obtained by concatenating its position $X_t^r = [x_t^r, y_t^r]$ and velocity $V_t^r = [\dot{x}_t^r, \dot{y}_t^r]$ vectors. The state of a simulated pedestrian is described in the same fashion and denoted by a p super-index S_t^p.

Features are computed from the robot state and pedestrian information and are used to describe the robot's context. In our experiments, we have both used features that have previously proposed in the literature and also defined new ones. Something that all features have in common is that, in order to compute them, only the human agents which are contained in a neighborhood of radius r around the robot position are taken into account. All features are described below in more detail.

a) Density features: The principle of these features [14] is to encode the Local density in the neighborhood (Fig. 1) with, in our case, three mutually-exclusive binary features. Every ϕ_{d_n} feature represents thus a range. The values used in our experiments are listed in Table I. As for all the other ranges in this paper, they have been obtained by and informed trial and error process.

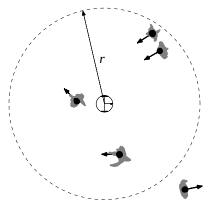


Fig. 1. Density features are computed by counting the number of human agents n contained in a circle of radius r and centered at the robot's position. In this example, the count is 4, corresponding to feature ϕ_{d_2}.

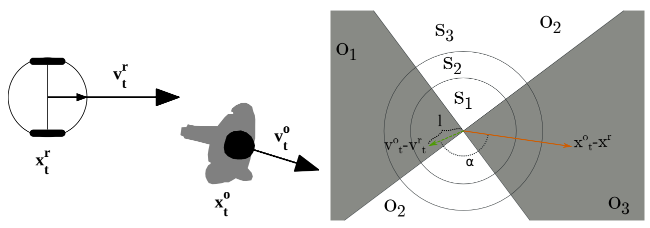
TABLE I
DENSITY FEATURE THRESHOLDS

Feature	*Range*
ϕ_{d_1}	$n \in (0, 2)$
ϕ_{d_2}	$n \in [2, 5)$
ϕ_{d_3}	$n \in [5, \infty)$

b) Speed+orientation features: These features [14] represent the relative speed and orientation of the pedestrians with respect to the robot assuming that they can be considered independently. This results in three magnitude features: $\phi_{s_{1:3}}$ and three orientation features: $\phi_{o_{1:3}}$. The thresholds are the same than for the velocity features described below.

c) Velocity features: Figure 2 illustrates the relative motion of the simulated pedestrians in the neighborhood with respect to the robot. The idea, inspired on [12], is to compute the average magnitude of robots moving in three different ways: (o_1) towards the robot, (o_2) perpendicular or (o_3) away from the robot. First, every agent is classified in the corresponding orientation bin according to the minimum angle between its relative position $X_t^p - X_t^r$ and velocity $V_t^p - V_t^r$. Then, for each bin, the average speed of the associated agents is computed and discretized in one of three levels (s_1, s_2, s_3). It is important to notice that features belonging to different orientation bins are not mutually exclusive. The orientation and magnitude thresholds used in our experiments are listed in Table II.

d) Default cost feature: This feature, denoted by ϕ_{def}, is always set to one. It has been proposed by [12] to allow IRL to learn how people balance the other features against the travelled distance. A similar idea has been previously proposed in [18] for a car parking application.



(a) Situation overview (b) Feature computation

Fig. 2. Relative orientation bins (o_1, o_2, o_3) are computed by estimating the minimum angle α between the relative position and velocity vectors. Magnitude is then computed per bin as the mean of the corresponding individual relative velocity magnitudes l. For the human agents in this example, feature ϕ_{o_1,s_1} will be turned on.

TABLE II
VELOCITY FEATURE THRESHOLDS

Feature	*Range*
$\phi_{o_1,\cdots}$	$\alpha \in (-\frac{3\pi}{4}, \frac{3\pi}{4}]$
$\phi_{o_2,\cdots}$	$\alpha \in [\frac{\pi}{4}, \frac{3\pi}{4}) \cup [-\frac{3\pi}{4}, -\frac{\pi}{4})$
$\phi_{o_3,\cdots}$	$\alpha \in [-\frac{\pi}{4}, \frac{\pi}{4})$
$\phi_{\cdots,s_1}$	$l \in [0, 0.015)$
$\phi_{\cdots,s_2}$	$l \in [0.015, 0.025)$
$\phi_{\cdots,s_3}$	$l \in [0.025, \infty)$

e) Social force features: These features are inspired on the social forces model proposed by Helbing [9], is used to compute the repulsive force between agents i and j according to the following equation:

$$f_{ij}^{\text{soc}} = ae^{\left(\frac{r_{ij}-d_{ij}}{b}\right)} N_{ij}\big(\lambda + 0.5(1-\lambda)(1+\cos(\varphi_{ij}))\big) \quad (1)$$

where a and b are tuning constants, r_{ij} is the sum of the radius of both agents, d_{ij} is their respective distance, N_{ij} is their normalized relative position and φ_{ij} is the relative orientation of the motion of agent i with respect to the vector that goes from agent i to agent j. Finally, λ is an important parameter which allows to define an anisotropic influence region that represents the fact that the obstacles in front of an agent are usually more relevant that those located behind it (see also Fig.4).

The force is used to compute features, denotes by $\phi_{sf_{1:3}}$, which are discretized in bins according to φ_{ij}. These features, however, are not binary. Instead, for every agent in the neighborhood, the corresponding relative orientation bin is computed from φ_{ij} which is then incremented by one if the force exceeds a predetermined threshold τ. In our experiments, we have fixed parameters as follows: $a = b = 1$, $\lambda = 2.0$ and $\tau = 0.5$. Relative orientation bins are discretized according to Table III.

TABLE III
SOCIAL FORCE FEATURES

Feature	*Range*
ϕ_{sf_1}	$\varphi_{ij} \in (-\frac{3\pi}{4}, \frac{3\pi}{4}]$
ϕ_{sf_2}	$\varphi_{ij} \in [\frac{\pi}{4}, \frac{3\pi}{4}) \cup [-\frac{3\pi}{4}, -\frac{\pi}{4})$
ϕ_{sf_3}	$\varphi_{ij} \in [-\frac{\pi}{4}, \frac{\pi}{4})$

f) Social+Relative Velocity forces: These last features extend the social force ones by proposing an additional force which takes into account the motion directions of both agents as follows:

$$f_{ij}^{\text{vel}} = (1 + \cos(\psi_{ij}))\; e^{(r_{ij} - d_{ij})}\; \|V_i - V_j\| \tag{2}$$

where ψ_{ij} is the relative orientation, computed in the same way as for the velocity features described above, and V_i, V_j are the agent's velocities.

Instead of discretizing them, the results of equations 1 and 2 are summed up for all agents and stored in the respective continuous features φ_{f_1} and φ_{f_2}.

B. *Motion Planning*

For motion execution, we recompute a plan every time step using a grid-based GPU implementation of Dijkstra's algorithm. We have discretized the state space in a three-dimensional grid in which each cell's side measures 1.0 m. The third dimension is used to represent the robot's orientation, discretized in eight values. Also, in order to take into account dynamic constraints, the robot is only allowed, when moving, to either keep its heading angle or to increase/decrease it by $\pi/4$ rad.

IV. Experimental platform

One of this paper's contributions is the software platform we used for all our experiments. Although it is still on pre-alpha stage, we intend to release it publicly, together with the final version of this paper. The platform is implemented as a number of learning algorithms, together with three ROS modules, which will facilitate the transition to experiments with our real robot. The software is open source and is being continuously improved, it is available at:
`https://github.com/srl-freiburg`.

- *Learning algorithms:* this includes GPU based implementations of the two algorithms presented here.
- *ROS Pedestrian simulator:* an improved ROS wrapper for the PedSim microscopic pedestrian crowd simulator [19].
- *ROS planning and control module:* GPU based implementations of the Dijkstra and Forward-Backward algorithms using gradient interpolation for control.
- *ROS Teleoperation module:* allows a human to operate the simulated robot via keyboard.

V. Experiments

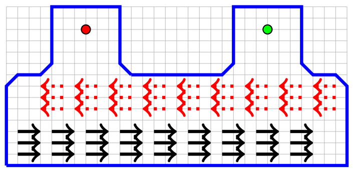
The main objective of our experiments was to evaluate the implemented IRL algorithms (c.f. Sec. II-B) and features. In order to do so, we have defined three scenarios (c.f. Sec. V-A), and four sets of features chosen from Sec. III-A according to selected papers on the literature.

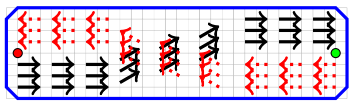
For every scenario, we have performed a set of teleoperated sequences which have been used for training, computing weight vectors for all the combinations of feature sets and IRL algorithms. For every such combination, we have also obtained –by trial and error– a manually tuned set of weights to be used as a baseline.

Additional teleoperated sequences have been used for evaluation, using a number of proposed objective and subjective metrics (c.f. Sec. V-C). The results are presented and analyzed in Sec. V-D.

A. *Scenarios*

In order to evaluate the capacity of the various features to represent different situations, we have defined three simulation scenarios (Fig. 3). Each such scenario presents particular challenges, for instance, in the airport gate scenario, the robot should join the left-right moving people "lane" in the bottom and then go up at the end, probably overshooting a little bit to avoid being "carried out" by the crowd. In the crossing hallway scenario, the main difficulty is to go through the central part, in which people is moving in all directions. Finally, in the intersection scenario, we would expect the robot to follow the natural "lanes" in similar fashion to a vehicle.


(a) Airport gate


(b) Crossing hallway

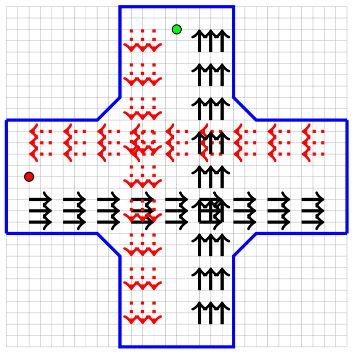

(c) Intersection

Fig. 3. Experimental scenarios (red circles = start, green circles = goal).

B. *Feature sets*

In order to study their respective performance to reproduce human-like behavior, we have organized the features presented in Sec. III into four *feature sets* (Table IV). In three cases, the selection corresponds to existing papers in the literature, as indicated in the table.

An interesting insight concerning the features sets was the need for the default feature ϕ_{def} in all cases. Without it, planning algorithms tended to behave erratically, and produced infinite paths in cases where all features were set to zero.

TABLE IV

EVALUATED FEATURE SETS

Name	*Features*	*Based on*
$\mathcal{F}_1$	$\phi_{d_{1:3}}, \phi_{o_{1:3}, s_{1:2}}, \phi_{def}$	[12]
$\mathcal{F}_2$	$\phi_{d_{1:3}}, \phi_{s_{1:3}}, \phi_{o_{1:3}}, \phi_{def}$	[14]
$\mathcal{F}_3$	$\phi_{sf_1}, \phi_{sf_2}, \phi_{sf_3}, \phi_{def}$	[9]
$\mathcal{F}_4$	$\phi_{f_1}, \phi_{f_2}, \phi_{def}$	—

C. Evaluation metrics

In order to compare the different algorithms and feature sets, we have defined two kinds of evaluation metrics:

1) *Objective metrics:* These task-oriented metrics quantify the efficiency of the robot while performing the given task. In our case, these metrics are:
   - *Path length:* distance traveled from the start to the goal location.
   - *Path smoothness:* smoothness of a trajectory. Computed as the sum of heading changes in subsequent steps along the path.
2) *Subjective metrics:* These metrics aim to reflect more intangible human factors, such as comfort. In human sciences, these are often obtained from questionnaires given to experimental subjects. Since we are using a simulator, we have resorted to numeric metrics to approximate them in similar fashion to [1]. For our experiments, we have defied two kinds of metrics:
   - *Proxemic intrusions:* number of intrusions into the intimate, personal, social and public space as defined by the Proxemics model [8] (see also Fig. 4).
   - *Anisotropic intrusions:* number of frontal, lateral and back intrusions into an area defined by the anisotropic influence model by Helbing *et al.* [20] (see also Fig. 4).

Finally, we have computed, for every feature set, three metric vectors –two for the learning algorithms and one for the manually tuned weights– containing the concatenation of the average objective and subjective values obtained from five runs of the respective learned weights. We have also computed, in every case, a an analogous reference metric vector R from the teleoperated paths, which is used to compute a normalized score, where lower values indicate more human-like behavior. For example, let $M_{\mathcal{F}_1,MM}$ be the metric vector obtained from feature set $\mathcal{F}_1$ using the max margin algorithm, the normalized score $\mathcal{S}_{\mathcal{F}_1,MM}$ is computed as:

$$\mathcal{S}_{\mathcal{F}_1,MM} = 100 * {}^{\|R - M_{\mathcal{F}_1,M}\|}/_{\|R\|} \tag{3}$$

D. Results

Figure 6 (left) shows the average normalized scores obtained for all scenarios using the weights learned from teleoperated data gathered in the same scenario. In order to study how the learned weights generalize to different scenes, Fig. 6 (right) shows the scores obtained from applying the weights learned from scene one and applied to scene three. From the figure, it can be concluded that, in our experiments:

- Both learning algorithms have similar performance and there is no clear winner among them.
- When using weights learned on the same scenario, learned weights are either comparable or much better than manually tuned ones.
- Feature set $\mathcal{F}_4$ is clearly the one that performs worse, while $\mathcal{F}_1$ and $\mathcal{F}_2$ are comparable, with a slight advantage for $\mathcal{F}_2$.
- Feature set $\mathcal{F}_3$ has the best performance on same scene weights, but seems to be the one that generalizes worst to other scenarios.

As an additional qualitative comparison of individual features, instead of feature sets, we ran additional tests using only one feature at a time and visually comparing the resulting paths against the teleoperated ones. Fig. 5 illustrates a typical result, with velocity features consistently producing the most similar paths, although considerably less smooth. Further studying the smoothness metric, we found out that, in all cases, teleoperation produced considerably smoother paths than using any of the learned weights and features. We attribute this to the lack of true motion prediction in all the approaches, which is something that people, on the other hand, naturally do when teleoperating the robot. An additional factor is the relative coarseness in the domain of the obtained cost functions (e.g. Fig. 7) which, even for our largest feature set are limited to 1024 cost values.

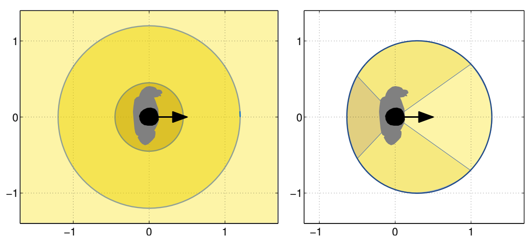


Fig. 4. Simulated human comfort measures. *Left:* we count intrusions of the robot into the intimate, personal, social and public space as defined by the Proxemics model (only the intimate, personal and part of the social spaces are shown). *Right:* we count intrusions of the robot into an anisotropic sphere based on the model of [20]. We quantize the robot's angle of approach and count the intrusions into the four directions head-on, sideways and alongside.

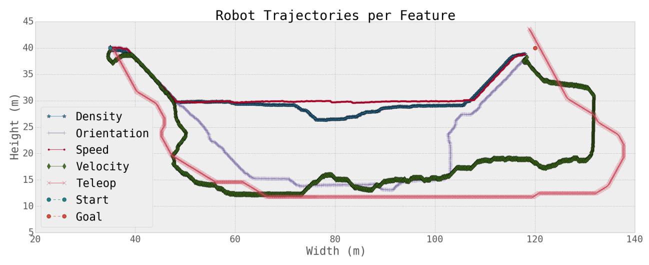


Fig. 5. Paths obtained in scenario one (airport) when only one of the features defined in Sec. III-A is active.

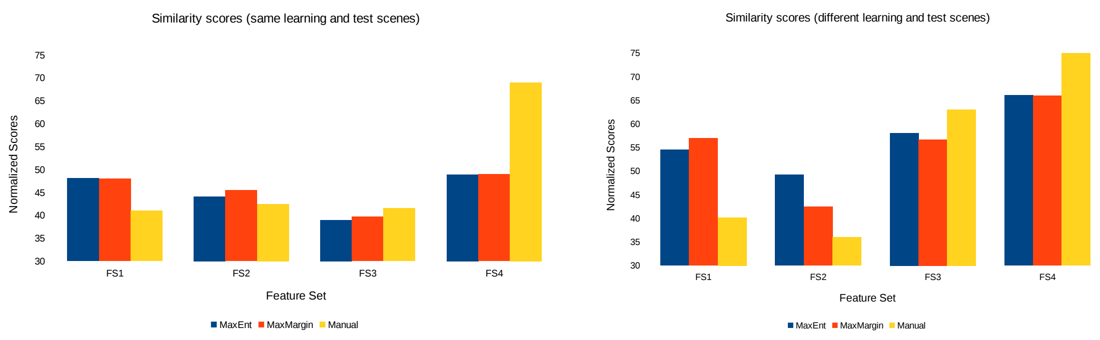


Fig. 6. Normalized scores per feature and learning algorithms (smaller is better). Left: same scenario. Right: different scenario.

VI. Conclusions

To the best of our knowledge, this paper presents the first experimental comparison of IRL based learning methods and feature sets for socially compliant robot navigation in crowds. It proposes two main contributions:

- An experimental open software platform to evaluate diverse IRL algorithms and feature sets for service robotics applications.
- A set of evaluation metrics and methodologies which constitute a first effort toward building a full benchmark for these techniques.

This work has also provided three important insights concerning the current state of the literature: a) the importance of the default cost feature; b) the need of motion prediction to obtain smoother human-like motion; and c) for this kind of cost function (i.e. linear combination of weights) it seems to be better to put the effort on feature design than on the learning algorithms. Conversely, in order to simplify the task of designing features, richer, more complex cost functions and learning algorithms are required.

Concerning our future work, we are interested in incorporating other algorithms (e.g. Bayesian IRL) and integrating feature prediction algorithms into the planning loop. We also explore the use of planning as a way of predicting other people's motion as in [11]. We also intend to extend our studies to the literature of Inverse Optimal Control (e.g. [21]).

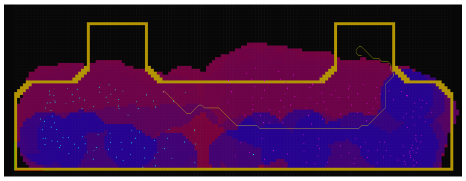


Fig. 7. Example costmap for scenario one, notice the relative lack of smoothness in the cost values, illustrated by the colors.

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. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Embodied social interaction for service robots in hallway environments," in *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, 2005.

[3] L. Scandolo and T. Fraichard, "An anthropomorphic navigation scheme for dynamic scenarios," in *Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, Chine, February 2011.

[4] J. Rios-Martinez, C. Laugier, and A. Spalanzani, "Understanding human interaction for probabilistic autonomous navigation using risk-RRT approach," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011.

[5] J. Mainprice, E. Sisbot, T. Simeon, and R. Alami, "Planning safe and legible hand-over motions for human-robot interaction," in *IARP Workshop on Techical Challenges for Dependable Robots in Human Environments*, Shanghai, China, 2010.

[6] Y. Nakauchi and R. Simmons, "A social robot that stands in line," *Autonomous Robots*, vol. 12, no. 3, pp. 313–324, May 2002.

[7] Y. Tamura, T. Fukuzawa, and H. Asama, "Smooth collision avoidance in human-robot coexisting environments," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[8] E. Hall, *Handbook of Proxemics Research*. Society for the Anthropology of Visual Communications, 1974.

[9] D. Helbing and P. Molnar, "A social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4284–4286, 1995.

[10] R. Middlemist, E. Knowles, and C. Matter, "Personal space invasions in the lavatory: suggestive evidence for arousal," *Journal of Personality and Social Psychology*, vol. 33, no. 5, pp. 541–6, 1976.

[11] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proc. of the Int. Conf. on INtelligent Robots and Systems (IROS)*, 2009.

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

[13] 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.

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

[15] S. Russell, "Learning agents for uncertain environments," in *Proc. of the 11th Annual Conf. on Computational Learning Theory*, 1998.

[16] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the 21st Int. Conf. on Machine Learning*, Banff (CA), 2004.

[17] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI Conference on Artificial Intelligence (AAAI'08)*, July 2008.

[18] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[19] C. Gloor, "PedSim: A Microscopic Pedestrian Crowd Simulation System," http://pedsim.silmaril.org, 2012, [Online; accessed Sept-2013].

[20] D. Helbing, I. J. Farkás, P. Molnár, and T. Vicsek, "Simulation of pedestrian crowds in normal and evacuation situations," in *Pedestrian and Evacuation Dynamics*, M. Schreckenberg and S. D. Sharma, Eds. Springer, Berlin, Germany, 2002.

[21] S. Levine, Z. Popovic, and V. Koltun, "Feature construction for inverse reinforcement learning," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 1342–1350.