# Learning Optimal Navigation Actions for Foresighted Robot Behavior During Assistance Tasks

AbdElMoniem Bayoumi    Maren Bennewitz

***Abstract*— We present an approach to learn optimal navigation actions for assistance tasks in which the robot aims at efficiently reaching the final navigation goal of a human where service has to be provided. Always following the human at a close distance might hereby result in inefficient trajectories, since people regularly do not move on the shortest path to their destination (e.g., they move to grab the phone or make a note). Therefore, a service robot should infer the human's intended navigation goal and compute its own motion based on that prediction. We developed an approach that applies reinforcement learning to get a Q-function that determines for each pair of the robot's and human's relative positions the best navigation action for the robot. Our approach applies a prediction of the human's motion based on a softened Markov decision process (MDP). This MDP is independent from the navigation learning framework and is learned beforehand on previously observed trajectories. We thoroughly evaluated our method in simulation and on a real robot. As the experimental results show, our approach leads to foresighted navigation behavior and significantly reduces the path length and completion time compared to naive following strategies.**

## I. Introduction

Following humans with mobile robots is a challenging problem and is needed in several applications such as in industrial settings where robots are deployed as transportation systems, in home scenarios, especially for the elderly people, in stores with autonomous shopping carts, or in scenes where robotic wheelchairs should navigate next to an accompanying pedestrian. Various solutions have been proposed for such problem instances as discussed in the next section. However, these solutions mainly focus on following the human at a certain distance irrespectively of its intended destination. Accordingly, the robot will keep on following the human, even if he does not move on the shortest path to his goal. This inefficiency might arise since humans are easily interrupted by unexpected events during navigation. For example, consider a home-assistance robot performing a delivery task and following the human when suddenly the phone rings and needs to be grabbed, or the doorbell rings. Or it might be that in the middle of the task an elderly person decides to rest for a while. In such situations, a robot applying a naive navigation strategy can only follow the human since the destination is initially unknown to the robot. This leads to inefficient navigation behavior causing unnecessary battery consumption or wear.

All authors are with the Institute of Computer Science, University of Bonn, Germany. This work has been supported by the German Academic Exchange Service (DAAD) and the Egyptian Ministry for Higher Education as well as by the European Commission under contract number FP7-610532-SQUIRREL.

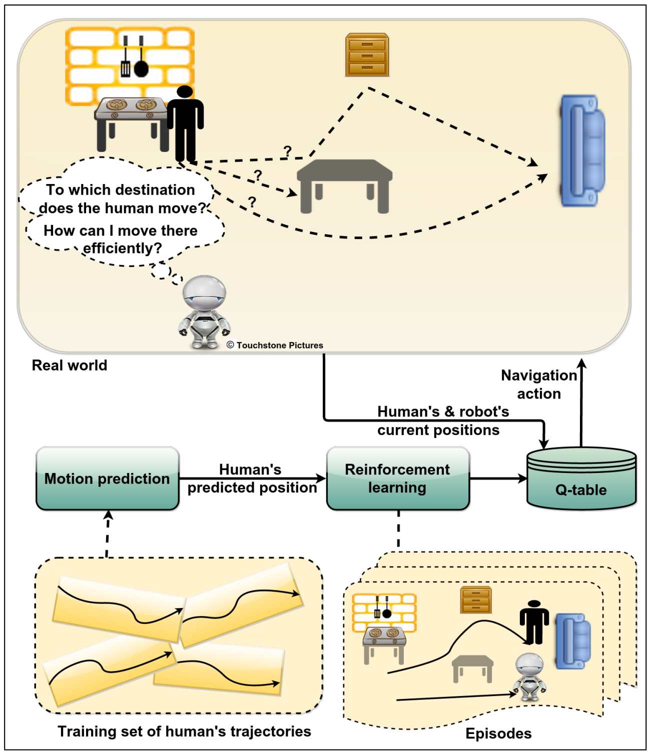

To which destination does the human move?
How can I move there efficiently?
© Touchstone Pictures
Real world
Navigation action
Human's & robot's current positions
Motion prediction
Human's predicted position
Reinforcement learning
Q-table
Training set of human's trajectories
Episodes

Fig. 1. The human moves through the environment between different possible designated locations (top) where it stays for a while and might need the help of a mobile robotic assistant. The task of the robot is to efficiently reach the initially unknown destination of the human, who might not move on the shortest trajectory. Our learning framework generates optimal navigation actions based on predicted motions of the human (bottom), which results in more foresighted behavior than just following the human at a close distance.

In this paper, we present a reinforcement learning framework that generates foresighted navigation actions in the described situations and is able to deal with such unexpected human behaviors during the tasks. Our approach constantly predicts the human's intended destination. Based on that prediction, the robot follows the learned navigation strategy and, thus, reaches the predicted navigation goal efficiently.

An overview of our framework is depicted in Fig. 1. We rely on a set of previously observed human trajectories between possible destinations where the human stays for a while and might need the help of the robot, i.e., for general assistance tasks, social interaction, or delivery tasks. Given these trajectories, we apply a technique to learn a prediction model that is used to reason about the future motions and the target destination of the human. The output of our learning framework is a table of Q-values that encodes the best navigation action for the robot based on the current robot's and human's positions as well as the predicted human's motion.

As we show in the experiments, our approach leads to foresighted navigation behavior. The robot focuses on reaching the human's final destination, which is initially unknown to the robot, efficiently. We demonstrate that a robot executing the learned navigation strategy can successfully handle cases in which the human's trajectory contains detours. Our method results in paths with significantly shorter path length and significantly reduced completion time compared to naive following strategies. To the best of our knowledge, we present the first solution for applications involving people following tasks that considers the efficiency of the generated robot paths.

## II. Related Work

The task of human following or target tracking has been thoroughly investigated using control theory, e.g., Huang [1] presented two control models for both the linear and angular velocity of a robot. These control models focus on originating velocity commands that allow the robot to follow a human while ensuring smoothness of the resulting trajectory. Harmati and Skrzypczyk [2] applied a fuzzy logic controller and proposed a game-theory based method for a team of robots tracking a target. In this approach, each robot has knowledge about the positions of its teammates and optimizes its control signals using on a cost function that takes also into account the predicted decisions of the other robots. Similarly, Nascimentoa *et al.* [3] developed an approach using a nonlinear predictive formation control model in a distributed architecture for collaborating robots. Each robot shares information about its pose with the rest of the team and optimizes its control signals under a prediction of the next states of the target as well as of the other team members. Pradhan *et al.* [4] proposed a path planning method with a navigation function that uses predictive fields of moving obstacles to follow a target.

Furthermore, there are approaches that aim at following a human with a fixed distance, e.g., Nishimura *et al.* [5] developed a modified shopping cart to follow a person in a certain range. Prassler *et al.* [6] considered side-by-side following within the application of a robotic wheelchair following a human. Also Kuderer and Burgard [7] considered the wheelchair application and developed an approach to predict the human trajectory taking into account information about obstacles in the environment. The authors compute the robot's trajectory so that the distance to the desired relative position along the predicted path is minimized. In this way, the robot can act foresightedly during the following task and local minima resulting from obstacles in the environment are avoided. Morales *et al.* [8] analyzed side-by-side trajectories of humans in order to predict the trajectory of a human during walking beside a robot. Using the learned utility function, the optimal robot trajectory can then be planned locally.

Further approaches predict human trajectories to generate robot motions that do not interfere with people. Bennewitz *et al.* [9] proposed to predict human trajectories based on learned motion patterns to avoid interferences in tight environments. Laugier *et al.* [10] use hidden Markov models to represent the motion of dynamic obstacles and avoid collision. Ziebart *et al.* [11] developed an approach using a Markov decision process (MDP) to learn from previously observed trajectories. The authors learn a Q-table to predict the human's trajectories for collision-free navigation. In our work, we use the method of Ziebart *et al.* to learn such a Q-table, however, we aim at making use of the prediction to learn how to efficiently follow a human.

Regarding the use of machine learning techniques in related problems, Goldhoorn *et al.* [12] proposed solving a hide-and-seek game using reinforcement learning with the aim to find a human and follow him/her to a goal location. The reward here relies on minimizing the distance between the following robot and the human during the following task. Tipaldi and Arras [13] focus on encountering a person in the environment, i.e., the robot has to find the person. The authors present a model that learns the spatio-temporal behavior of humans and use this model within a MDP to increase the encounter probability.

As opposed to all the approaches discussed above, we present a learning framework that enables a service robot to efficiently reach the intended destination of the human. Using our approach, the robot is not required to follow the human with a fixed distance, which might lead to inefficient trajectories, instead the robot predicts the navigation goal, constantly updates it, and adapts its actions accordingly.

## III. Motion Prediction

In this section, we present the motion prediction technique applied in our learning framework. We predict the human's trajectory based on the trajectory observed so far and use the prediction in the state space representation of our learning approach for generating foresighted robot actions. Our technique for motion prediction is based on the work of Ziebart *et al.* [11]. This approach models the sequence of motion actions performed by a human as a softened Markov decision process (MDP) whose state space corresponds to the cells of a discretized grid map of the environment. The authors propose to train a prediction model using a softened version of the Bellman equation and value iteration to get a Q-table that represents the most likely motion action performed by the human at a certain position. This softened version uses the soft-maximum function instead of the ordinary maximum to be able to reason about the distribution of the trajectories, instead of considering only one single trajectory. The soft-maximum function is defined as

$$softmax_x f(x) \quad = \quad \log \sum\nolimits_x e^{f(x)} \qquad (1)$$

and is used within the computation of the state and action values $V(s)$ and $Q(s,a)$:

$$\begin{aligned} Q(s,a) &= R(s,a) + V(T(s,a)) && (2) \\ V(s) &= softmax_a \; Q(s,a) && (3) \end{aligned}$$

Here, $s$ and $a$ represent the human's current state and corresponding action, respectively, $T(s,a)$ is the transition function, and $R(s,a)$ is the reward after executing action $a$ at the current state $s$. Eq. (2) and Eq. (3) are used to train the motion predictor based on a set of training trajectories with a reward function that also takes into account obstacle locations. Ziebart *et al.* use the obtained Q-table to predict the future destination of the trajectory observed so far as explained in the following.

Let $\zeta_{A\to B}$ denote the observed trajectory of the human from the initial state $A$ to the current state $B$ and $\zeta_{B\to C}$ the future trajectory from $B$ to the unknown destination $C$. The probability $P(dest\ C|\zeta_{A\to B})$ of a certain destination $C$ given the observed trajectory $\zeta_{A\to B}$ can then be computed with Bayes' rule, where the likelihood $P(\zeta_{A\to B}|dest\ C)$ intuitively depends on the ratio of the reward of $\zeta_{A\to B}$ and the expected value of $\zeta_{B\to C}$ to the value of the whole trajectory to the destination $\zeta_{A\to C}$:

$$\begin{aligned} & P(dest\ C|\zeta_{A\to B}) \\ & \overset{\text{Bayes'}}{=} \frac{P(\zeta_{A\to B}|dest\ C)P(dest\ C)}{P(\zeta_{A\to B})} \qquad (4) \\ & = \frac{\frac{e^{R(\zeta_{A\to B})+V(B\to C)}}{e^{V(A\to C)}}P(dest\ C)}{\sum_D \frac{e^{R(\zeta_{A\to B})+V(B\to D)}}{e^{V(A\to D)}}P(dest\ D)} \qquad (5) \end{aligned}$$

Here, $D$ corresponds to a destination among the set of possible destinations according to the training trajectories. The prior distribution $P(dest\ D)$ is known from the training set[1]. Furthermore, the reward of the trajectory $R(\zeta)$ is the sum of all individual rewards of state-action pairs according to $\zeta$:

$$R(\zeta) = \sum_{(s,a)\in\zeta} R(s,a) \qquad (6)$$

and $V(X\to Y)$ denotes the softmax value function of the trajectory from state $X$ to state $Y$.

We can then compute the probability of the future trajectory $\zeta_{B\to C}$ to the unknown future destination $C$ given the so far observed trajectory $\zeta_{A\to B}$:

$$\begin{aligned} & P(\zeta_{B\to C}|\zeta_{A\to B}) \\ & = P(\zeta_{B\to C}|dest\ C)P(dest\ C|\zeta_{A\to B}) \qquad (7) \\ & = \frac{e^R(\zeta_{B\to C})}{e^{V(B\to C)}}P(dest\ C|\zeta_{A\to B}) \qquad (8) \\ & = e^{R(\zeta_{B\to C})-V(B\to C)}P(dest\ C|\zeta_{A\to B}) \qquad (9) \end{aligned}$$

The term $P(dest\ C|\zeta_{A\to B})$ is hereby computed using Eq. (5). This posterior probability is used to weight the conditional probability $P(\zeta_{B\to C}|dest\ C)$ of the expected future trajectory given the destination $C$.

In our implementation, we assume that the human can move one step at a time step in any of the eight possible directions corresponding to the neighbor cells or remain at the same state. Accordingly, our action space consists of nine actions. We set the reward to be inversely proportional to the distance from obstacles within a certain range of the human.

Thus, using the probability distribution about the future trajectory, we can predict the position of the human at a certain time step given its trajectory history. Note that since there is uncertainty in the prediction, the robot does not know the human's destination in advance and, thus, cannot directly plan a path towards it. Accordingly, we use the information about possible destinations in our navigation framework and learn optimal navigation actions as described in the following section.

## IV. Learning Navigation Actions

### A. Reinforcement Learning

To model a problem as reinforcement learning task, one needs to define a state space $\mathcal{S}$ that describes the relevant aspects of the situation the agent encounters and an action space $\mathcal{A}$ that represents the set of actions from which the agent can choose according to its policy $\pi : \mathcal{S} \to \mathcal{A}$. The state of the agent at a time step $s_t$ is transformed to the new state $s_{t+1}$ after executing the action $a_t$ according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and the agent gets an immediate reward $r_t \in \mathcal{R}$.

The learning process is iterated multiple times until convergence is reached, where each iteration is called a learning episode. A learning episode is considered complete when a terminal state is reached at time $T$. During each learning episode, the value of each state-action pair $Q^\pi(s,a)$ following the policy $\pi$ is updated according to the achieved reward $R_t$ with

$$R_t = \sum_{i=t+1}^{T} r_i \qquad (10)$$

as follows [14]:

$$Q^\pi(s_t,a_t) = E_\pi\{R_t|s_t,a_t\} \qquad (11)$$

Thus, $Q^\pi(s_t,a_t)$ denotes the expected return of taking action $a_t$ in state $s_t$ and following the policy $\pi$ afterwards. The optimal policy maximizes the expected return, which corresponds to choosing always the action with the maximum Q-value. To update the Q-values, we apply Sarsa reinforcement learning:

$$\begin{aligned} & Q^\pi(s_t,a_t) \leftarrow Q^\pi(s_t,a_t) \\ & \quad + \quad \alpha[r_{t+1} + \gamma Q^\pi(s_{t+1},a_{t+1}) - Q^\pi(s_t,a_t)] \qquad (12) \end{aligned}$$

Here, the parameters $\alpha$ and $\gamma$ correspond to the step size and the discounting factor, respectively. In our implementation, we use a variant called Sarsa($\lambda$) that looks several steps into the future and considers the rewards of these future steps to update $Q(s_t,a_t)$ [15], unlike Sarsa which considers only one-step forward. In this way, a more accurate estimate of $R_t$ can be obtained. Throughout the learning phase, we use an $\varepsilon$-greedy action selection. This selection method chooses the action with the highest $Q$-value with probability $1-\varepsilon$ and all non-greedy actions with equal probability.

[1]When using a motion capture system, possible destinations can be identified as places where the person frequently stays for a while without moving much. The distribution $P(dest\ D)$ can be learned by counting how often the person moves between possible destinations and might depend on the starting location.

## B. State Space $\mathcal{S}$

We use a discretized representation of the environment in form of a grid map in which static obstacles are represented as occupied cells. The state representation for reinforcement learning includes the relative distance between the current 2D position of the human $\mathbf{x}_t^h$ and the current 2D position of the following robot $\mathbf{x}_t^r$ at time $t$ as well as the relative distance between $\mathbf{x}_t^r$ and the predicted position of the human $\mathbf{x}_{t+i}^h$ after $i$ further time steps

$$s_t = \begin{bmatrix} \mathbf{x}_t^h - \mathbf{x}_t^r \\ \mathbf{x}_{t+i}^h - \mathbf{x}_t^r \end{bmatrix}. \tag{13}$$

Here, we compute $\mathbf{x}_{t+i}^h$ according to Eq. (9). We use relative positions instead of global positions to reduce the learning problem. The corresponding state space is much more compact than one that considers all possible combinations of current and predicted global positions. As shown in the experiments, this generalization works well in practice.

## C. Action Set $\mathcal{A}$ and Q-Table

The action space consists of a set of discrete moving actions in the eight neighbor cells in addition to standing still. The Q-table computed by our framework contains entries $Q(s, a)$ for each state-action pair where $s$ is defined as in Eq. (13) and $a$ is one of the nine navigation actions.

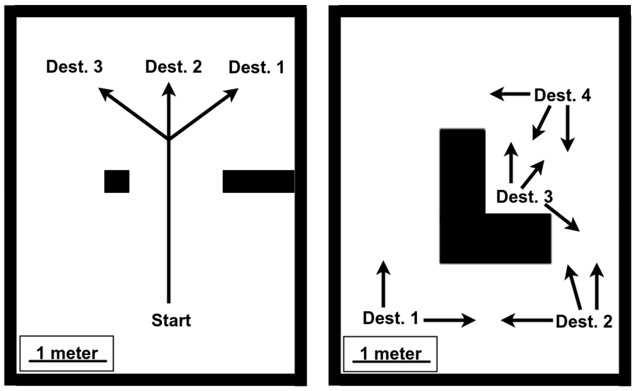
We assume that the human moves between different locations where it stays for a while and might need the help of the robot. One of them is the starting position of the human before moving to the next destination. Accordingly, we learn an independent Q-table for each of these designated locations within the map.

## D. Reward $\mathcal{R}$

We designed the reward function so as to combine both the shortest path to the predicted human position as well as the difference between the distance traveled so far by the human and the traveled distance by the follower robot. The first term aims at following the human, whereas the second term is for preferring shorter paths during the task. Accordingly, we define the intermediate reward $r_t$ at time $t$ as follows

$$r_t = \begin{cases} 10,000 & \text{if } t = T \\ -dist_{A^*}(\mathbf{x}_t^r, \mathbf{x}_{t+i}^h) + (trav_t^h - trav_t^r) & \text{otherwise,} \end{cases} \tag{14}$$

where $T$ is the final time step and $dist_{A^*}$ is a function that applies the A* algorithm on the grid map to compute the length of the shortest path between the current robot position $\mathbf{x}_t^r$ and the predicted position of the human $\mathbf{x}_{t+i}^h$. Furthermore, $trav_t$ refers to the distance traveled until time step $t$. The final state with time step $T$ is reached when the robot is sufficiently close to the human after he has reached the next destination, which is the case when the human stays for a while close to a designated destination. The reward function leads to the generation of efficient navigation actions that minimize the cost of reaching the human's destination.



(a) (b)

Fig. 2. Maps used for the experiments: (a) Environment with a fixed start position and three possible destinations with overlapping trajectories, making the prediction more difficult. (b) Environment with several designated locations between which the human moves.

## V. Experimental Results

## A. Environment Setup

We evaluated our approach in simulated and real-world experiments in two environments each of the size of 4.8 m$\times$3.6 m. The first one contains three possible destinations reachable with overlapping trajectories from an initial position while the other one contains trajectories between multiple designated locations (see Fig. 2). We used a map resolution of 60 cm, which yields reasonable navigation actions for the real robot with a diameter of 45 cm. Note that even if the environment consists of 48 grid cells, the size of the actual state space is much bigger since we do not only consider the relative distance between the current positions of the robot and the human but also the predicted difference in a certain number of time steps (see Eq. (13)).

Currently, we assume perfect knowledge about the robot's and the human's trajectory and that both are moving approximately with the same velocity, which only affects the prediction.

## B. Trajectory Generation for Training and Testing

We randomly generated human trajectories for both, the training and test phases composed of straight line segments of 25 cm length (in the future we will use real data acquired with a motion capture system). The orientation of the segments relative to the destination is chosen uniformly from the interval $[0°, 60°]$. For the map in Fig. 2 (a), our training set consists of 60 trajectories, 20 for each of the three possible destinations. For the map in Fig. 2 (b), we generated a training set consisting of 15 trajectories for each of the possible combinations between the designated locations. For testing, we used five randomly generated trajectories for each motion class.

## C. Parameters

During the learning of the Q-table, we used an initial value of 0.4 for $\varepsilon$ of the greedy action selection to allow for exploring the state space and decreased the value to 0.2 after a certain number of iterations. For the execution, we used a value of 0.05, i.e., the robot chooses the action with the highest Q-value with probability 0.95. The Q-tables for our experiments were learned from a minimum of 12,000 learning episodes (learning was stopped after convergence). We consider a learning episode as successful if the distance of the robot to the human destination is smaller than 1.2 m within a maximum number of 100 time steps, otherwise the episode is aborted. We experimentally determined the value of 3 for $i$ for the prediction in Eq. (13) and Eq. (14) to work best with the chosen map resolution of 60 cm.

In the learning episodes and test runs, we generated the robot's starting position randomly within a range of 60 cm around the human's initial position. In the test runs, we abort the execution in case the robot does not reach the human destination within 25 time steps after the human arrives there.

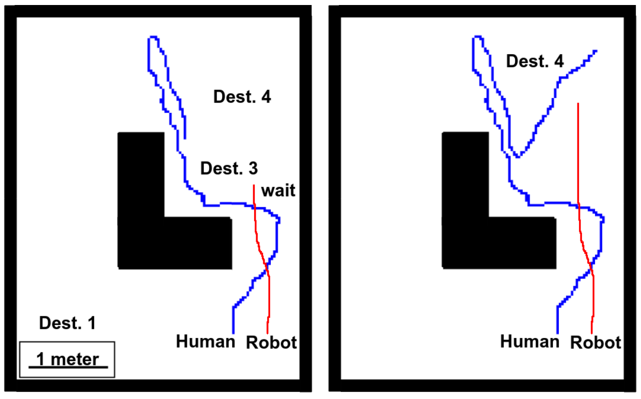
## D. Evaluation Metrics

In order to evaluate our approach, we computed the saving w.r.t. the path length by comparing the distance traveled by the robot according to our learned navigation actions to that of a naive following method, in which the robot follows the shortest path to the position of the human at each time step. Furthermore, we evaluated the completion time of our approach to a 'wait-and-observe-first' strategy in which the robot waits until the human reaches its destination and only then starts moving according to the shortest path to the destination. To show statistical significance of our results, we applied a *statistical two-tailed paired t-test* with a significance level of $\alpha$ = 0.05.

## E. Experiments in Simulation

We performed 250 runs for each test trajectory to alleviate the randomness arising from the generation of the initial position of the robot as well as the randomness involved in the generation of the actions. As can be seen from Tab. I (first and third row, first column), our approach significantly outperforms the naive following strategy w.r.t. the trajectory length in both environments. Note that in the environment of Fig. 2 (b), in some cases the robot can directly predict the correct navigation goal after one step and immediately move there. Depending on the human's and the robot's start location, the robot then reaches this destination faster than the human.

Furthermore, we added a number of non goal-directed trajectories to the test set (25%) of the environment in Fig. 2 (a) to show the effectiveness of the robot behavior generated by our learning framework and its ability to handle unexpected trajectories. These trajectories were not included in the training of the Q-table for the navigation actions. In the corresponding runs, the human trajectory leads around the obstacle in the left part of the map (similar to the human's trajectory in Fig. 5). This can be seen as the case where the human moves to fetch some items and then continues walking to its actual destination. If such scenarios are included in the test set we even achieve an overall average gain of 19.1% (see Tab. I second row, first column).



(a) (b)

Fig. 3. Experiment in which the human first makes a large detour before walking towards his intended destination. (a) The robot does not follow the human but waits first. (b) Only as the human continues moving in the direction of the actual destination, the robot follows and, thus, avoids the detour that would have resulted from closely following the human.

When comparing our approach to the 'wait-and-observe-first' strategy regarding completion time, we achieve an average gain of 13.1% - 14.6% (Tab. I second column).

In 8% of the test runs, the robot was caught in local minima and did not reach the destination within a fixed time limit while choosing actions according to the Q-table. The corresponding runs are not included in the evaluation. Note that we applied only the Q-table for action selection in our approach and did not combine it with moving on the shortest path to the actual destination after the human arrived there, which would be reasonable in practice and improve the performance in some cases, in particular when facing the local minimum problem mentioned above.

We performed an additional experiment to illustrate the strength of our approach. Here, the human was only walking to his final destination after performing a larger detour (see Fig. 3 (a)). This unexpected behavior was correctly handled by the robot as it did not follow the human but waited. Only as the human continued his path towards the destination, the robot started moving again (see Fig. 3 (b)). Thus, the robot could avoid the large detour that would have resulted from closely following the human.

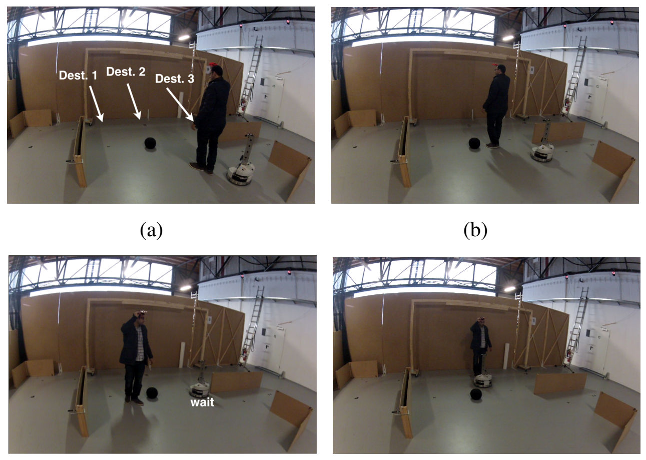
## F. Experiments with a Real Robot

We also carried out experiments with a real robot (Robotino by Festo) to test the learned navigation strategy. To detect the position of the human and localize the robot, we used an external motion capture (MoCap) system. We focused on the scenario where the trajectory of the human does not directly lead to his/her destination.

TABLE I

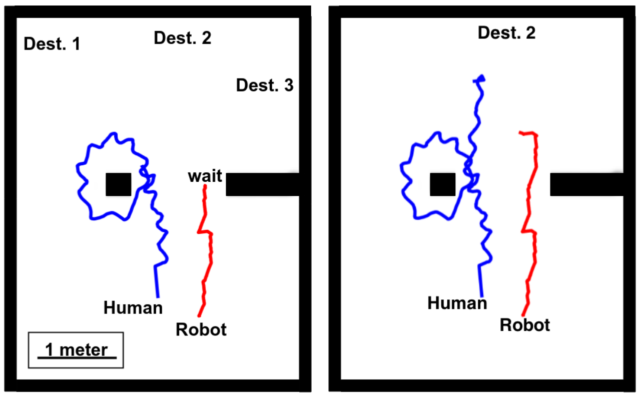
GAIN IN TRAVELED DISTANCE AND COMPLETION TIME

| Test trajectories of environments | | Gain in distance | Gain in time | Statistically significant ($\alpha = 0.05$) | |
|---|---|---|---|---|---|
| | | | | Dist. | Time |
| Fig. 2 (a) | Basic set | 7.9% | 13.1% | Yes | Yes |
| | With cycles | 19.1% | 14.6% | Yes | Yes |
| Fig. 2 (b) Basic set | | 18.3% | 14.2% | Yes | Yes |



(a) (b) (c) (d)

Fig. 4. (a) The human starts walking towards his (unknown) destination and the robot executes navigation actions to follow him. (b) The human does not move directly to his destination but walks in a cycle around an object. (c) The robot behaving according to our learned policy ignores this cycle and waits. (d) The human continues walking towards his destination, followed by the robot.



(a) (b)

Fig. 5. Trajectories of the human and the robot corresponding to the experiment shown in Fig. 4 recorded by the MoCap system. (a) The robot's trajectory is rather efficient, as the robot correctly predicts that the human will continue moving to one of the destinations in the area on top and therefore waits for the human half way. (b) Both human and robot resume the path to the destination.

As shown in Fig. 4, the human performed an inefficient trajectory by walking around the obstacle in the left part instead of directly going to his/her goal location. As can be seen, the follower robot ignored this cycle and waited instead before proceeding to follow the human afterwards to its intended navigation goal. The corresponding path (see Fig. 5) is much shorter than the human's trajectory.

## VI. Conclusions

We developed an approach to generating foresighted navigation behavior of an assistance robot. We consider the scenario in which the human moves between different designated locations where it might interact with the robot. Thus, the task of the robot is to reach these places while minimizing trajectory length and completion time. Our framework relies on a learned prediction model for the human's motions that is used to reason about its future trajectory and target destination. Based on that prediction and the current robot position, we apply reinforcement learning to generate the optimal navigation action for the robot. As the experiments carried out in simulation and with a real mobile robot demonstrate, our approach leads to foresighted navigation behavior resulting in significantly shorter paths and a significantly reduced completion time compared to naive following strategies.

We are currently extending our system to use the robot's on-board sensors for people tracking, which will lead to occlusions and uncertainty about the human's position. So the robot will also need to learn when to consider active re-localization of the human based on its predicted motions.

## References

[1] L. Huang, "Control approach for tracking a moving target by a wheeled mobile robot with limited velocities," *Control Theory & Applications, IET*, vol. 3, no. 12, pp. 1565–1577, 2009.

[2] I. Harmati and K. Skrzypczyk, "Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 75–86, 2009.

[3] T. P. Nascimento, A. P. Moreira, and A. G. Scolari Conceio, "Multi-robot nonlinear model predictive formation control: Moving target and target absence," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1502–1515, 2013.

[4] N. Pradhan, T. Burg, S. Birchfield, and U. Hasirci, "Indoor navigation for mobile robots using predictive fields," in *Proc. of the American Control Conference*, 2013.

[5] S. Nishimura, H. Takemura, and H. Mizoguchi, "Development of attachable modules for robotizing daily items – Person following shopping cart robot," in *Proc. of the IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2007.

[6] E. Prassler, D. Bank, B. Kluge, and M. Hägele, "Key technologies in robot assistants: Motion coordination between a human and a mobile robot," *Transactions on Control, Automation and Systems Engineering*, vol. 4, no. 1, 2002.

[7] M. Kuderer and W. Burgard, "An approach to socially compliant leader following for mobile robots," in *Social Robotics*, ser. Lecture Notes in Computer Science, M. Beetz, B. Johnston, and M.-A. Williams, Eds., vol. 8755. Springer International Publishing, 2014.

[8] Y. Morales, S. Satake, R. Huq, D. Glas, T. Kanda, and N. Hagita, "How do people walk side-by-side? Using a computational model of human behavior for a social robot," in *ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2012.

[9] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *Int. Journal of Robotics Research (IJRR)*, vol. 24, no. 1, 2005.

[10] C. Laugier, S. Petti, D. Vasquez, M. Yguel, T. Fraichard, and O. Aycard, "Steps towards safe navigation in open and dynamic environments," in *Autonomous Navigation in Dynamic Environments*. Springer, 2007, pp. 45–82.

[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 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009, pp. 3931–3936.

[12] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu, "Continuous real time POMCP to find-and-follow people by a humanoid service robot," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014.

[13] G. Tipaldi and K. Arras, "I want my coffee hot! learning to find people under spatio-temporal constraints," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[14] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT Press, 1998.

[15] M. Wiering and J. Schmidhuber, "Fast online Q($\lambda$)," *Machine Learning*, vol. 33, no. 1, pp. 105–115, October 1998.