Improving Navigation with the Social Force Model by Learning a Neural Network Controller in Pedestrian Crowds

Peter Regier Ibrahim Shareef Maren Bennewitz

***Abstract*— In this paper, we present a novel, efficient approach to improve the acceleration commands computed by the popular social force model (SFM) [1] for navigation through pedestrian crowds. Our method consists of two stages. In the first phase, we collect training data with a simulated approach. In this step, we modify the steering acceleration commands from the SFM according to a set of discrete alterations and simulate the motion of the robot as well as the pedestrians into the future for each alteration. We rate each resulting trajectory based on a cost function and apply the best steering command to the robot. While controlling the robot in such way, we collect for every time step the input and output training data. In the second stage, we then learn a neural network given the collected training data. We use the best acceleration values experienced in the first phase as target values for the neural network and define simple input features describing the local surrounding of the robot. In extensive simulation experiments using different pedestrian densities, we demonstrate that the controls generated by the learned neural network lead to a significantly reduced number of collisions with pedestrians compared to the results of the basic SFM controller, while achieving similar or even shorter completion times.**

I. Introduction

The ability to navigate through crowds of people is essential for all service robots operating in human populated environments such as office corridors, hospitals, or shopping malls. Generating motion commands for navigation among people is a complex task for which the prediction of the motions of surrounding people needs to be taken into account to be able to anticipate and avoid collisions (see Fig. 1).

A popular approach to model human-aware navigation is the social force model (SFM) [1], which drives individuals through the combination of repulsive and attractive virtual forces called "social forces". However, such virtual force field methods have limitations since mobile robots controlled by using force fields can be trapped in local minima and show oscillation behavior in narrow spaces. In addition, encountering dense crowds of pedestrians can lead to a high number of collisions [2], [3].

We therefore propose a two-stage approach to improve the controls generated by a SFM based controller to realize efficient and collision-free robot navigation through pedestrian crowds. The idea of the first training phase is to improve the SFM based robot navigation. To do so, we simulate the positions of the robot as well as of the pedestrians in the robot's local environment a few time steps into the future. We hereby assume the positions and velocities of the

All authors are with the Humanoid Robots Lab, University of Bonn, Germany. This work has been supported by the DFG Research Unit FOR 2535 Anticipating Human Behavior.

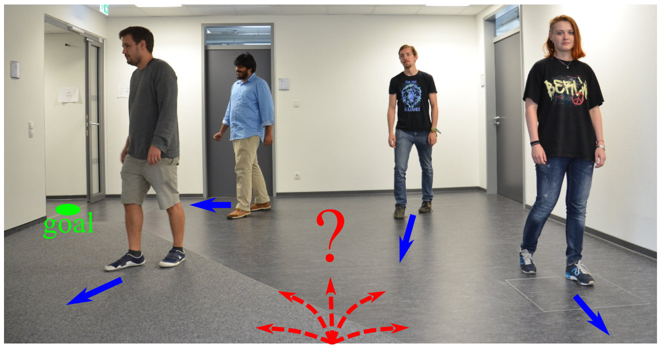


Fig. 1: Example observation of a robot navigating through environments populated by humans. To avoid collisions and efficiently reach the goal, the robot has to decide on its control commands. We propose to use a neural network in combination with the popular social force model (SFM) to generate acceleration commands given features describing the predicted configuration of the humans in the robot's vicinity.

pedestrians to be known. For the robot's control, we consider a discrete set of incremental alterations of the accelerations commands generated by the SFM based controller. Based on the prediction for a given steering command, our system then evaluates the surroundings of the robot, i.e., the number of pedestrians in the robot's vicinity and the distance to its navigation goal. The best result of the acceleration values is chosen as the control command to be executed by the robot. With this predictive controller, the robot navigation behavior can be seriously improved since it forecasts the situation and reacts accordingly.

However, the prediction is computationally expensive and, furthermore, a real robot will not have knowledge about the velocity and the target destination of all surrounding people. Therefore, we propose to train in a second step a neural network (NN) based on simple features describing the configuration of the pedestrians in the robot's vicinity and the output of the predictive controller. The NN combined with the SFM can then be used for safe and efficient navigation through dense crowds.

As we show in extensive simulation experiments with different densities of pedestrians, the controls generated by the NN lead to a significantly reduced collision rate of the robot with pedestrians in comparison to the basic SFM controller. Furthermore, the experiments demonstrate that a robot controlled by the NN achieves a speed that is comparable to the one generated by the SFM controller.

The contribution of our work is a controller that is learned in a supervised manner based on simple features describing the pedestrian state around the robot. To generate the training data, we predict the state of the local environment by simulating forward the motions of pedestrians and the robot.

II. Related Work

A navigation approach that uses the SFM in combination with machine learning was presented by Ferrer *et al.* [4]. The authors introduced a metric computed as the weighted sum of the robot's social forces and determined the weights of the different forces using a Markov Chain Monte Carlo Metropolis-Hastings algorithm. Our approach, in contrast, learns to improve the acceleration commands of the robot provided by the SFM depending on the current situation.

Inverse reinforcement learning (IRL) has been the most widely-used approach applied to achieving social navigation. Henry *et al.* [5] utilized IRL to teach a path planner on a 2D grid from example traces of pedestrians. The authors note that a robot agent in such a context would have limited knowledge of its environment and therefore estimate the density and flow of pedestrians around it using Gaussian processes. Gaussian Processes have also been utilized for modeling pedestrian motion by Vemula *et al.* [6]. The authors used real human trajectory data to train their model and propose to discretize each agent's neighborhood and construct an occupancy grid per agent containing all information of its neighbours. This approach is able to predict future trajectories of observed pedestrians using a Monte Carlo sampling approach as well as to compute the path of a robot through a dense crowd. Kuderer *et al.* [7] and Vasquez [8] used maximum-entropy IRL to learn a behaviour model of pedestrians in order to generate socially compliant trajectories. In the work of Vasquez [8], the speed and orientation of the pedestrians in relation to the robot are used as observations for the robot. However, IRL is typically computationally expensive leading to long training times and requires a large training set in order to learn the reward function.

Kim and Pineau [9] proposed to use maximum mean discrepancy as a query metric to determine how far an observation is from the distribution of the training data up to that point. Teaching values are requested only if this metric is above a certain threshold. This approach produces safe trajectories, however, it is rather computationally expensive to train. Recently, Tai *et al.* [10] presented generative adversarial imitation learning that directly learns a policy from expert demonstrations using the so-called trust-region policy optimization without going through the intermediate step of learning a reward function and without needing a map of the environment. The authors make use of the SFM and generate a large set of training examples in order to train their model based on raw depth data. So far, the approach has been only applied to specific basic navigation scenarios.

Recently, long short-term memory (LSTM) have become popular for motion prediction. For example, Everett *et al.* [11] consider motion planning for a group of robots. The authors defined a reward function based on velocity, change in heading direction, and distances to the goal as well as the closest other agent. They use a LSTM cell at the input layer where each agent in the environment subsequently feeds the cell its own state at every decision step. By considering the information of all agents about the state, the LSTM contains a fixed-length, encoded state of the world for the current decision step. The authors then apply reinforcement learning to generate the navigation policy. As each learning episode contains information from several agents the learning is accelerated. However, the requirement of this method to have global knowledge about the states of all agents limits its applicability in a real-world scenario. Alahi *et al.* [12] proposed an LSTM for modeling human motions in crowds. The focus in this approach was on trajectory prediction without hand-crafted functions. Also Pfeiffer *et al.* [13] use LSTM neural networks to predict the motion of humans, in this case taking explicitly into account static obstacles to improve the prediction in cluttered environments. In the future, we plan to investigate whether our predictive controller can be improved even more with those recently presented capable prediction methods. However, for this work we predict the motions of pedestrians using the social force model.

III. Social Force Model

In the following, we first introduce the social force model (SFM) [1], [14], [15], which is the basis of our approach. The SFM models human motion as the sum of three different *social forces*:

- The *desired force* $\vec{f}_i^0$ reflecting the desire of pedestrian i to reach a particular destination at a particular speed,
- The *obstacle force* $\vec{f}_i^{wall}$ as repulsive force between static obstacles and pedestrian i,
- The *social interaction force* $\vec{f}_{ij}$ as repulsive force between pedestrians i and j.

The desired force $\vec{f}_i^0$ is defined as

$$\vec{f}_i^0 = \frac{\partial \vec{v}_i^0}{\partial t} = v_i^0 \vec{e}_i^0 - \vec{v}_i(t) \ , \quad (1)$$

where $\vec{v_i}$ is the current velocity of pedestrian i, $\vec{e}_i^0$ is a unit vector pointing from the pedestrian to the goal and v_i^0 is the desired speed.

The obstacle force $\vec{f}_i^{wall}$ decays with the distance d_w between the pedestrian and a static obstacle:

$$\vec{f}_i^{wall}(d_w) = e^{\frac{-d_w}{0.2}} \quad (2)$$

The social interaction force is defined as the sum of two components: f_v that describes the deceleration along the interaction direction $\vec{t}_{ij}$ and f_θ that describes the directional changes along $\vec{n}_{ij}$.

The interaction direction is given by

$$\vec{t}_{ij} = \frac{\lambda(\vec{v}_i - \vec{v}_j) + \vec{e}_{ij}}{\|\lambda(\vec{v}_i - \vec{v}_j) + \vec{e}_{ij}\|} \ , \quad (3)$$

where $\vec{v}_i - \vec{v}_j$ is the relative motion between the pedestrians, $\vec{e}_{ij}$ the unit vector pointing from pedestrian i to j and λ is the relative weight of the two directions.

Let $\vec{n}_{ij}$ be the normal vector of $\vec{t}_{ij}$, oriented to the left. The forces f_v and f_θ are defined as

$$f_v(d_{ij},\theta_{ij}) = -Ae^{\frac{-d_{ij}}{B-(n'B\theta_{ij})^2}} \tag{4}$$

and

$$f_\theta(d_{ij},\theta_{ij}) = -AKe^{\frac{-d_{ij}}{B-(nB\theta_{ij})^2}} \ , \tag{5}$$

where d_{ij} denotes the distance between two pedestrians i and j and θ_{ij} is the angle between $\vec{t}_{ij}$and $\vec{e}_{ij}$. The parameter K is the sign of the angle θ_{ij} and B is modelled as

$$B = \gamma||\lambda(\vec{v}_i - \vec{v}_j) + \vec{e}_{ij}|| \ . \tag{6}$$

The social interaction force $\vec{f}_{ij}$ can now be defined by combining the Equations 4 and 5:

$$\vec{f}_{ij}(d_{ij},\theta_{ij}) = -Ae^{\frac{d_{ij}}{B}}[e^{-(n'B\theta_{ij})^2}\vec{t}_{ij} + Ke^{-(nB\theta_{ij})^2}\vec{n}_{ij}] \tag{7}$$

Moussaïd [14] defined the parameters $A, B, n, n', \gamma, \lambda$ by fitting the function in Eq. (7) to real-world data of pedestrian behavior. The complete equation of motion for a pedestrian, which determines how its velocity changes per unit time, is then defined as the sum of the three forces:

$$\frac{\partial \vec{v}_i}{\partial t} = \vec{f}_i^0 + \vec{f}_i^{wall} + \sum_j \vec{f}_{ij} \tag{8}$$

IV. Pedestrian and Robot Motion

Since the SFM can still lead to collisions, especially in scenarios with dense crowds of pedestrians, our aim is to improve the robot's acceleration commands given the SFM commands. We propose to train a neural network that outputs improved acceleration commands. To learn the NN, we use a simulator that models the pedestrians and robot motion according to the SFM.

A. Pedestrian and Robot Controller

Our simulated environment is based on an implementation of the SFM as described in the previous section. Additionally, we modify Eq. (8), that determines the pedestrian and robot motion, by applying weights F_{df}, F_{of}, F_{sf} to each component:

$$\frac{\partial \vec{v}_i}{\partial t} = F_{df}\vec{f}_i^0 + F_{of}\vec{f}_i^{wall} + F_{sf}\sum_j \vec{f}_{ij} \tag{9}$$

B. Robot Model

We assume a non-holonomic robot and define the robot state as tuple of the global xy-position and orientation θ. The robot control is determined by v^x and v^θ, which are its linear velocity in the heading direction and the angular velocity around the vertical axis, respectively. Thus, the trajectories for any constant velocity have a curvature with the radius $r = v^x/v^\theta$ and the new state of the robot at time step $t+\Delta t$ is determined as:

$$\begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ \theta(t+\Delta t) \end{pmatrix} = \begin{pmatrix} x + r\big(\sin(\theta + \Delta t \cdot v^\theta) - \sin(\theta)\big) \\ y + r\big(\cos(\theta + \Delta t \cdot v^\theta) - \cos(\theta)\big) \\ \theta + v^\theta \Delta t \end{pmatrix} \tag{10}$$

The velocities are clamped to reasonable values, i.e., $|v^x| \leq 1.0$ m/s and $|v^\theta| \leq 1.0$ rad/s. Given the linear and angular accelerations a^x and a^θ we can compute the robot's velocities as

$$v^x = v_r^x + a^x \cdot \Delta t \tag{11}$$

$$v^\theta = v_r^\theta + a^\theta \cdot \Delta t \tag{12}$$

In our simulation, we determine a^θ based on the resulting social force vector and the robot orientation, and the robot rotational velocity.

$$a^\theta = K_p \cdot \Delta\alpha + K_d \cdot (-v^\theta) \tag{13}$$

where K_p and K_d denote the coefficients and $\Delta\alpha$ is the angle between the social force vector and the heading direction of the robot. To compute the linear acceleration a^x, we project the social force vector onto the heading direction of the robot and take the magnitude of the new vector as value for a^x, if $\Delta\alpha$ is less than or equal to 70°. That means, we only apply forward acceleration when the social force vector is within the field of view of the robot, whereby $\Delta\alpha = 0$ means the maximal acceleration. In case $\Delta\alpha$ is greater than 70°, we set a^x to the maximum possible negative value to slow down or stop the robot.

V. Predictive Controller

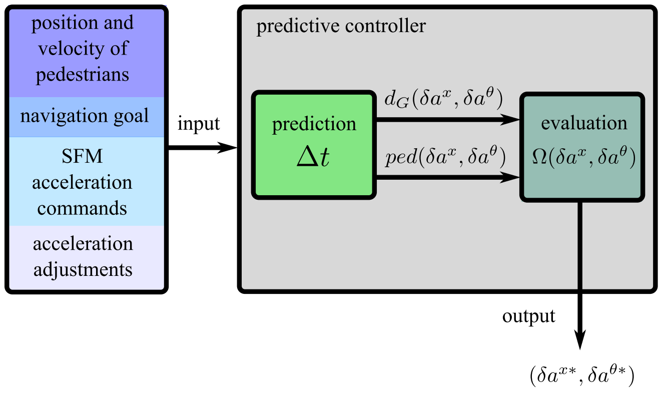
We developed a predictive controller that aims at improving the computed SFM commands. The predictive controller considers a set of adjustment values of the acceleration commands generated by the SFM controller and simulates the pedestrians' as well as robot motion some time steps into the future, evaluates the resulting configuration based on an objective function, and chooses the best result. The best acceleration values are then used as teacher values of the neural network described in the next section.

In more detail, the predictive controller has knowledge about the positions and velocities of the pedestrians within a certain range around the robot and tries to improve the acceleration commands computed by the SFM controller at every time step by considering a set of adjustments, defined as the tuple of linear acceleration adjustment δa^x and angular acceleration adjustment δa^θ from a set of discrete possible adjustments tuples. The predictive controller then simulates the motion of the pedestrians as well as the robot for each considered adjustment for time Δt into the future and evaluates the resulting configuration with the following objective function

$$\begin{aligned} \Omega(\delta a^x, \delta a^\theta) = & \qquad \qquad (14) \\ & -\alpha d_G(\delta a^x, \delta a^\theta) - \beta ped(\delta a^x, \delta a^\theta) - \gamma col(\delta a^x, \delta a^\theta), \end{aligned}$$

where $d_G(\delta a^x, \delta a^\theta)$ is the resulting Euclidean distance from the robot to its goal, $ped(\delta a^x, \delta a^\theta)$ is the number of pedestrians that are predicted to be in a $3\,m$ radius around the robot, $col(\delta a^x, \delta a^\theta)$ is the number of collisions encountered during the simulation step, and α, β, and γ are weights.

The tuple $adj^* = \{\delta a^{x*}, \delta a^{\theta*}\}$ that achieves the highest evaluation is then added to the acceleration values returned



position and velocity of pedestrians
navigation goal
SFM acceleration commands
acceleration adjustments
input
predictive controller
prediction
Δt
$d_G(\delta a^x, \delta a^\theta)$
$ped(\delta a^x, \delta a^\theta)$
evaluation
$\Omega(\delta a^x, \delta a^\theta)$
output
$(\delta a^{x*}, \delta a^{\theta*})$

Fig. 2: Overview of the predictive controller that outputs the best acceleration adjustment $adj^* = \{\delta a^{x*}, \delta a^{\theta*}\}$ for the calculated SFM command. The objective function Ω takes into account the progress towards the goal d_G and the resulting number of pedestrians within a certain range ped. See text for a detailed explanation.

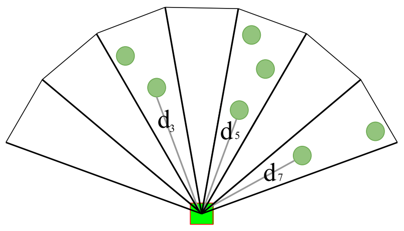


Fig. 3: The features used as input to the neural network are the robot's observations of the pedestrians within its field of view. The robot's sensor field is divided into seven zones. The feature vector consists of the distance to the closest pedestrian per zone, as well as the number of pedestrians per zone. In this example, the vector would be as follows: [(0,0),(0,0),(d_3,2),(0,0),(d_5,3),(0,0),(d_7,2)]

by the SFM and chosen as the command to be executed by the robot. Fig. 2 illustrates the different components of the predictive controller.

VI. Training a Neural Network

The predictive controller introduced in the previous section makes observations from its simulated environment which cannot be obtained in the real world. In addition, the calculations are too computationally expensive. We therefore propose to learn a neural network that can subsequently be used for a real robot to generate improved acceleration commands based on the SFM. The output of the predictive controller is hereby used as the training data.

We generated a training set by running our simulator and recording at each time step a set of features describing the situation and the adjustment tuple from the predictive controller. The final training set contained approximately 300,000 data pairs of features and acceleration adjustments and was collected with different densities of pedestrians.

Since typically the velocity of the people cannot be reliably observed, we use a different set of features to describe the configurations of pedestrians around the robot based on the available information. In particular, we define a field of view in front of the robot with an opening angle of $140°$ with a 3 m radius and divide this area into seven equally sized angular zones as depicted in Fig. 3. For each of these zones, we determine the distance of the robot to the nearest

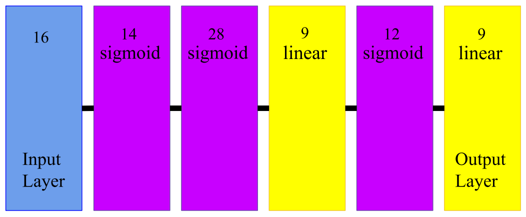


Fig. 4: The neural network architecture used for the robot controller (input layer in *blue*, layers of linear neurons in *yellow*, and layers of sigmoid neurons in *purple*).

pedestrian as well as the number of pedestrians currently within the area. We use this information to describe the local environment around the robot during training and application of the NN.

As a result, the input dimension of the neural network is 16 (2 features for each of the 7 zones and the linear and angular SFM acceleration) and the output, corresponding to the acceleration values $\delta a^{x*}, \delta a^{\theta*}$ given by the predictive controller, has a dimension of 2. An example how the feature vector is computed is provided in Fig. 3.

We used the mean-square error as a loss function and optimized training using RMSProp. The network as depicted in Fig. 4 achieved the best validation loss rates for all architectures we evaluated, with an overall training time of only 10 minutes with an Intel Core i5 CPU for 300.000 collected data points.

The weights of the neural network are initialized with the Glorot normalization [16]. The output layer of the neural network has two linear neurons corresponding to δa^x or δa^θ. Note that in general it might also be possible to train absolute velocities which, however, would require a much larger dataset for training and a more complex network architecture.

VII. Experiments

We performed extensive experiments in simulation to evaluate the performance of the neural network controller in comparison to the basic SFM and the predictive controller in terms of number of collisions and completion time of the navigation task.

A. Parameters and Setup

For the SFM we used the following parameters: $A = 1.0, n = 2, n' = 3$ in Eq. (7). The desired velocity was set to $v_i^0 = 0.8m/s$ and the parameters of Eq. (9) were set to $F_{sf} = 2.1$, $F_{df} = 1.0$, $F_{of} = 1.0$.

For the predictive controller, we experimentally determined the range of $\delta a^x, \delta a^\theta \in [-0.3, -0.2, -0.05, 0.0, 0.05, 0.2, 0.3]$ for the adjustment values of the predictive controller and thus considered 49 adjustment tuples. For the objective function in Eq. (14), we chose $\alpha = 5, \beta = 1, \gamma = 6$. These values showed the best performance for the pedestrian controller. The prediction time was set to $\Delta t = 0.4\,s$ and the simulated environment ran with a frequency of 10Hz.

Our experimental environment consists of a corridor with a width of $10\,m$, bounded by two parallel walls. The robot's

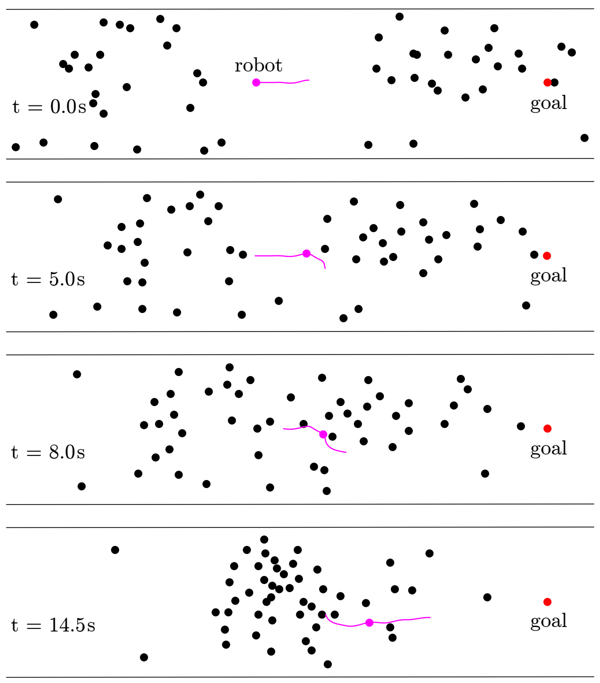


Fig. 5: Experimental setup with 50 pedestrians (black) and a corridor size of 50x10 meters. The pedestrians on the left side have to reach the right end of the corridor and vice versa. The robot (magenta) starts in the middle of the corridor and has to reach the goal on the right (red). The resulting trajectory is illustrated for five seconds before and after each shown time step.

starting and goal position across all runs was the same with a distance of $21\,m$ between these two points. We performed the evaluation for six different pedestrian densities. For each run, we distributed the pedestrians equally at either end of the corridor and randomly initialized their position in a certain region (Fig. 5). The goal positions of the pedestrians are randomly sampled along the width of the corridor at the opposite end.

For each of the six density groups we performed 50 runs using the SFM controller, the predictive controller, and the neural network controller in combination with the SFM.

B. Average Number of Collisions for Different Pedestrian Densities

We first evaluated the average number of collisions between the robot and pedestrians for all three controllers. We hereby assume a collision to happen when the outer boundaries of the robot and a pedestrian come into contact, where both radii were set to 0.2 m. The results depicted in Fig. 6 show that both, the predictive controller and the neural network controller achieve a seriously reduced collision rate for all densities compared to the basic SFM controller. Over all runs, the NN controller shows a reduction of 31% in the number of collisions compared to the SFM controller. Most importantly, we show that the NN controller achieves a performance comparable with the predictive controller, even if it uses simple features instead of the complete information from the simulation. As these results demonstrate, the NN successfully imitates our predictive controller and seriously improves the SFM commands.

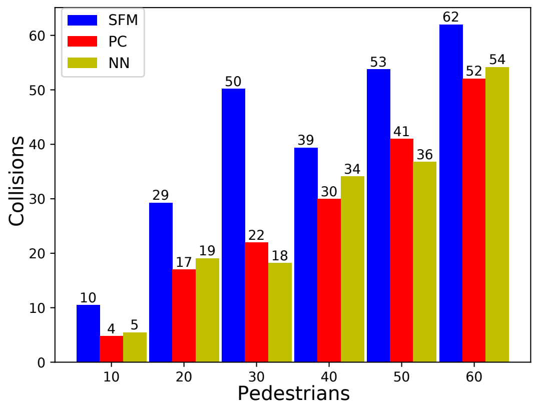


Fig. 6: Average number of collisions per run for the basic SFM controller (SFM), the predictive controller (PC), and the neural network (NN) for different pedestrian densities. Both, the predictive controller and the neural network seriously outperform the basic SFM for all densities.

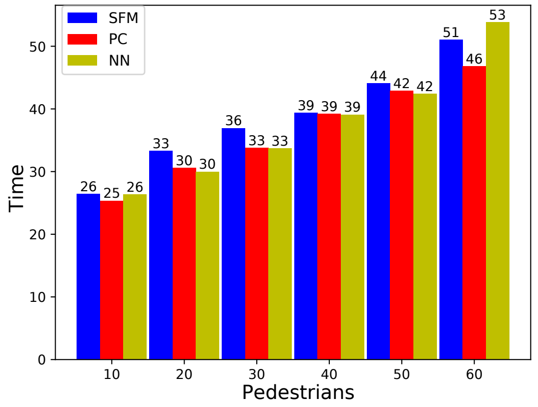


Fig. 7: Average completion time for the SFM controller, predictive controller (PC), and the neural network (NN) for different pedestrian densities. As can be seen, there is no significant difference between the three controllers up to 50 pedestrians even if the NN and PC controller manage better to avoid collisions with pedestrians (as shown in Fig. 6). Only for 60 pedestrians the completion time of the NN controller is larger than the time of the PC controller, suggesting that additional training for very dense crowds is needed. These results demonstrate that the behavior of the predictive controller can be mimicked successfully by the NN controller, which is not using the omniscient simulation for improving control commands.

C. Average Completion Time for Different Pedestrian Densities

Furthermore, we compared the completion time of the robot when controlled by the SFM controller, the predictive controller and the neural network to navigate through different pedestrian densities.

The results illustrated in Fig. 7 show that the NN controller achieves a completion time that is comparable to the one of the PC controller on average and outperforms the SFM controller up to 50 pedestrians. The evaluation confirms again that our NN is able to imitate the behavior of the predictive controller. For 60 pedestrians the completion time of the NN approach is higher than the time of both, the SFM and the PC controller, suggesting that additional training for very dense crowds is needed.

D. Qualitative Evaluation

Fig. 5 shows how the robot navigates through a pedestrian crowd of 50 people towards its goal location (red). The

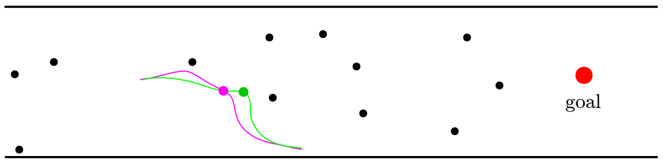


Fig. 8: Qualitative comparison of the NN controller and the SFM controller for the same pedestrians configuration. The NN controlled robot initiates the evasive action earlier (magenta) than the SFM controlled robot (green).

magenta line illustrates the robot trajectory five seconds before and after the current time step. As can be seen, the robot moves smoothly through the pedestrians by applying the acceleration adjustments of the NN. The reduced number of collisions of the NN controller results from foresighted behavior in terms of early evasive actions into areas with only few pedestrians (see the example Fig. 8).

VIII. Discussion

To use our NN controller in real-world scenarios, the robot system needs to be equipped with a people tracking system similar as in [17]. Furthermore, the robot needs to localize itself in an environment with a highly occluded field of view due to the presence of people around it. Both aspects introduce errors and noise into the system compared to our noise-free simulation. Thus, currently we are working on modifying our simulation so that it is closer to real-world scenarios by modeling the errors introduced by people tracking and motion prediction and the localization module.

For this work, we chose supervised learning to train a neural network. Another option to solve the problem would be to use reinforcement learning, which works with a reward function to train the network based on experienced runs, instead of taking the output of the predictive controller as teaching example as in our approach. In practice, however, the development of a network structure is much more convenient and faster with a dataset collected using the predictive controller, as we can quickly re-run the training when modifying the structure of our network. Also reinforcement learning processes are more sensitive to hyper parameters of the system and require a lot of tuning to achieve the desired outcome.

IX. Conclusion

In this paper, we proposed a novel robot controller based on a neural network to improve the navigation behavior of a robot steered by the popular social force model (SFM). To train the neural network, we developed a predictive controller that uses information about the positions and velocities of pedestrians within the vicinity of the robot and simulates their motions a few time steps ahead. For the robot's motions, the predictive controller evaluates a set of incremental adjustments of the SFM computed acceleration commands and determines the best result based on the number of close pedestrians and the progress towards the navigation goal. The best acceleration adjustments values are then used as target values for the neural network. As input, we use simple features describing the configuration of the pedestrians in the robot's vicinity.

The learned network can be efficiently applied on a robot in combination with an SFM based controller and uses only features that can be derived from real observations. As our simulation experimental results with different pedestrian densities show, the motion commands generated by the predictive controller as well as by the neural network lead to a significantly reduced collision rate in comparison to the basic SFM controller while maintaining a comparable velocity.

References

[1] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[2] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, and D. Prattichizzo, "Walking ahead: The headed social force model," *PloS one*, vol. 12, no. 1, 2017.

[3] F. Zanlungo, T. Ikeda, and T. Kanda, "Social force model with explicit collision prediction," *EPL (Europhysics Letters)*, vol. 93, no. 6, p. 68005, 2011.

[4] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *Proc. of the European Conference on Mobile Robots (ECMR)*, 2013.

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

[6] A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2017.

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

[8] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2014.

[9] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *Int. Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[10] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially-compliant navigation through raw depth inputs with generative adversarial imitation learning," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2018.

[11] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2018.

[12] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2018.

[14] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz, "Experimental study of the behavioural mechanisms underlying self-organization in human crowds," *Proc. of the Royal Society of London B: Biological Sciences*, vol. 276, no. 1668, pp. 2755–2762, 2009.

[15] 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, p. e10047, 2010.

[16] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Int. Conf. on Artificial Intelligence and Statistics*, 2010.

[17] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, *et al.*, "Spencer: A socially aware service robot for passenger guidance and help in busy airports," in *Proc. of the Conf. on Field and Service Robotics*, 2016.