2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Real-Time Navigation in Crowded Dynamic Environments Using Gaussian Process Motion Control

Sungjoon Choi, Eunwoo Kim, and Songhwai Oh

Abstract— **In this paper, we propose a novel Gaussian process motion controller that can navigate through a crowded dynamic environment. The proposed motion controller predicts future trajectories of pedestrians using an autoregressive Gaussian process motion model (AR-GPMM) from the partially-observable egocentric view of a robot and controls a robot using an autoregressive Gaussian process motion controller (AR-GPMC) based on predicted pedestrian trajectories. The performance of the proposed method is extensively evaluated in simulation and validated experimentally using a Pioneer 3DX mobile robot with a Microsoft Kinect sensor. In particular, the proposed method shows over 68% improvement on the collision rate compared to a reactive planner and vector field histogram (VFH).**

I. Introduction

Robots are becoming capable of adapting to complex and unstructured environments and interacting with humans to assist our daily life tasks [1]. Together with this technological development and careful consideration of the human nature, it is easy to expect that more service robots will be assisting us in the near future in places, such as offices, malls, and homes. However, for a robot to coexist with humans and operate successfully in a crowded and dynamic environment, it must be able to navigate safely and harmoniously with human participants in the environment. In order to address this issue, autonomous navigation under uncertain dynamic environments has received much attention recently [2]–[7].

A number of existing state-of-the-art navigation methods are, however, designed to operate in a simulated or restricted environment. Many methods assume that the current positions of a robot and dynamic obstacles are available [3], [4] or can be estimated from an external tracking system [2]. Such assumptions cannot be satisfied in many real-world environments since collecting precise location information about many moving objects in a wide area is an expensive option. Furthermore, the high computational cost of proposed algorithms makes them difficult to be applied for real-time applications.

On the other hand, reactive control algorithms, such as the potential field method [5] and vector field histogram (VFH) [8], [9], are computationally efficient. But they require a tedious process of manual parameter tuning, making it impractical to be used for a new or dynamic environment. In a potential field method, one has to carefully design two conflicting potential functions (attractive and repulsive) such that the combined potential function provides the desired result. Hence, while a potential field approach is highly effective for a static environment, it is extremely difficult to apply the method for a dynamic environment. Vector field histogram (VFH) is proposed to overcome inherent limitations of potential field methods [8], [10]. But VFH also requires a good choice of parameters, e.g., a smoothing factor in a polar histogram and the parameter for measuring the influence of an obstacle based on its distance to a robot. Another weakness of VFH is that it only considers current measurements. VFH* is proposed to overcome this shortcoming [9] and it is described in more detail in Section II.

In this paper, we propose a novel non-parametric Bayesian motion controller that can effectively navigate through crowded dynamic environments using an autoregressive Gaussian process model (AR-GP). It can effectively solve limitations of tedious parameter tuning as well as restrictions of reference view based methods (see Section II) by using the non-parametric Bayesian method in a partially-observable egocentric view. In simulation, the proposed method outperforms a reactive planner and VFH in terms of the collision rate and the minimum distance to dynamic obstacles. The proposed algorithm achieves the average collision rate of $6.86\,\%$, while a reactive planner and VFH achieve $22\,\%$ and $27.29\,\%$, respectively. The proposed algorithm is also implemented using a Pioneer 3DX differential drive mobile robot with a Microsoft Kinect camera.

The remainder of this paper is organized as follows. In Section II, we discuss related work and make comparisons to the proposed method. In Section III, we propose a method for modeling dynamic motion patterns and a motion controller using an autoregressive Gaussian process. In Section IV and V, the performance of the proposed algorithm is evaluated in simulation and validated experimentally using a physical robot.

II. Related Work

A number of studies in autonomous navigation for a dynamic environment have been conducted. We can group navigation methods into four different categories using two main criteria. This first criterion is the perspective of a view used by the navigation method (reference or egocentric) and the second is the method used for predicting the future positions of dynamic obstacles (model-based or data-driven). The

This research was supported in part by a grant to Bio-Mimetic Robot Research Center funded by Defense Acquisition Program Administration (UD130070ID) and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2009348).

S. Choi, E. Kim, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 151-744, Korea (e-mail: {sungjoon.choi, eunwoo.kim, songhwai.oh}@cpslab.snu.ac.kr).

TABLE I: Classification of navigation algorithms

	Model-based prediction	Data-driven prediction
Reference view	[4], [5], [11]–[13]	[2], [3], [6], [14]–[18]
Egocentric view	[19], [9]	This work

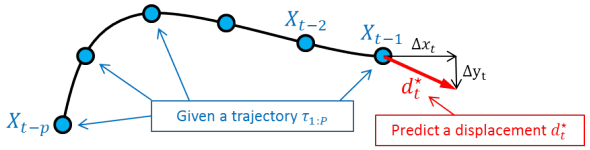


Fig. 1: Autoregressive motion model.

classification of state-of-the-art algorithms and our proposed method is summarized in Table I.

In a reference view approach, the position of a robot and other obstacles are assumed to be given or measured by external devices. From this complete environmental information, the level of safety of a planned path can be calculated or sometimes guaranteed in a probabilistic manner [3], [4]. However, these algorithms are limited to static environments or dynamic environments with known obstacle trajectories. An egocentric view approach uses information from the perspective of an operating robot. Since the information is coming from a partially-observable view, it is sometimes hard to guarantee complete collision avoidance [19]. However, since these algorithms do not need additional devices for localizing a robot and obstacles, they can be easily deployed in many practical settings.

We can also categorize navigation algorithms based on the method used to predict the future behavior of a dynamic obstacle. A model-based prediction method assumes that a dynamic obstacle follows a certain type of dynamics, e.g., a constant velocity model, and makes a prediction based on the chosen dynamic model. In a data-driven prediction method, the dynamic model of an obstacle is learned from a training set of past trajectories. Many data-driven prediction methods use the reference view approach. These algorithms are relatively hard to be used in real-world experiments since motion patterns must be learned for the entire operating space.

1) Reference View + Model-Based Prediction: Luders et al. [4] have proposed a chance-constrained rapidly-exploring random tree (CC-RRT) which guarantees probabilistic feasibility for obstacles with linear dynamics. Assuming that the linear dynamic model is corrupted by a white Gaussian noise and the shape of each obstacle is a polygon, probabilistic feasibility at each time step can be established based on the predictive distribution of the state of obstacles.

A potential function based path planner is proposed in [5]. An elliptical shape potential field is used for signifying the predicted position and the direction of an obstacle. It is shown that it is possible to encode probabilistic motion information into the potential field formulation. [13] proposed a model predictive control (MPC) approach under the RRT framework to speed up the computation time. The proposed algorithm selects the best trajectory according to the cost of traversing a potential field.

2) Reference View + Data-Driven Prediction: In [14], Joseph et al. used a mixture of Gaussian processes with a Dirichlet process prior to model and cluster an unknown number of motion patterns. In [15], Aoude et al. combined CC-RRT [4] and the motion model from [14] and proposed an RR-GP path planning algorithm [3], [15]. From diverse simulations, they have shown that an RR-GP can identify probabilistically safe trajectories in the presence of dynamic obstacles.

Trautman et al. [2] proposed multiple goal interacting Gaussian processes (mIGP) which are an extension of an interacting Gaussian process (IGP) from [16]. From a number of experiments in a university cafeteria, the proposed algorithm outperformed a non-cooperative algorithm similar to [15] and a reactive planner [20]. In their experiment, the positions of pedestrians and a robot are obtained using Point Grey Bumblebee2 stereo cameras mounted in the ceiling.

3) Egocentric View + Model-Based Prediction: Park et al. [19] proposed the model predictive equilibrium point control (MPEPC) framework to generate safe and smooth trajectories. By using MPEPC, a locally optimal decision is made at every step. In [9], Ulrich et al. presented the VFH* algorithm, which is an enhanced version of VFH. While the original VFH only looks for gaps in locally constructed polar histogram, VFH* predicts the future condition of a robot with a look-ahead verification and generates a control input considering both current and future conditions.

III. Non-Parametric Bayesian Motion Controller

A. Autoregressive Gaussian Process Motion Model

In this section, we focus on the problem of predicting the future trajectory of a dynamic obstacle (pedestrian) given current and past observed positions and propose an autoregressive Gaussian process motion model (AR-GPMM). Predicting the future trajectory of a pedestrian is significantly important because an accurate motion prediction is a key element in successful navigation in a human-robot environment.

We represent positions of obstacles in an egocentric Cartesian coordinate where the origin is at the center of the robot and the y-axis corresponds to the heading of the robot. A motion pattern is defined as a mapping from autoregressive (AR) positions of order p in the egocentric Cartesian coordinate, i.e., $\{(x_{t-1}, y_{t-1}), (x_{t-2}, y_{t-2}), \ldots, (x_{t-p}, y_{t-p})\}$, to a distribution over displacements at the current position, $d_t^\star = (\Delta x_t, \Delta y_t)$, as shown in Figure 1. This approach is similar to [15], however, instead of using only the current position in the reference Cartesian coordinate, we use p current and past positions in the egocentric Cartesian coordinate.

An autoregressive (AR) model is a random process for describing a certain time-varying process. An AR model of

order p is defined as

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + w_t$$

where $\varphi_1, ..., \varphi_p$ are the parameters of the model, c is a constant, and w_t is a white noise. Let $X_t = (x_t, y_t) \in \mathbb{R}^2$ be a position at time t, then the current velocity can be modeled using the AR model as follows:

$$\begin{aligned} \Delta X_t &= X_t - X_{t-1} \\ &= c + \sum_{i=1}^{p} \varphi_i X_{t-i} + w_t - X_{t-1}. \end{aligned} \quad (1)$$

We can generalize (1) using a Gaussian process as follows:

$$\begin{aligned} \Delta X_t &= f(X_{t-1}, X_{t-2}, ..., X_{t-p}) \\ &\sim GP_f(X_{t-1}, X_{t-2}, ..., X_{t-p}). \end{aligned} \quad (2)$$

A Gaussian process (GP) can be specified by a covariance function $k(\cdot, \cdot)$ which measures the correlation between two given inputs [21]. In our case, an input trajectory $\tau = \{X_{t-1}, X_{t-2}, \ldots, X_{t-p}\}$ is a set of p recent positions of a pedestrian and we use the following squared exponential function as a kernel function

$$k_t(\tau, \tau' \,|\, \theta_{traj}) = \sigma_f^2 \exp\left(-\sum_{i=1}^{p} \frac{\|\tau_i' - \tau_i\|^2}{2\sigma_\tau^2}\right) + \sigma_w^2 \delta(\tau, \tau'), \quad (3)$$

where $\theta_{traj} = (\sigma_f^2,\ \sigma_\tau^2,\ \sigma_w^2)$ are hyperparameters and $\delta(\tau, \tau') = 1$ if $\tau = \tau'$ and zero otherwise.

To generate realistic trajectories of pedestrians, we use the potential field method. Figure 2(a) shows the potential field used by our algorithm to generate trajectories (see Figure 2(b)). Using these generated trajectories, we make a training set for an AR-GPMM, i.e., $D_t = \{(\tau_i,\, d_i) \mid i = 1, 2, ..., N_t\}$, where τ_i is a trajectory of length p, $d_i = (\Delta x,\, \Delta y)$ is the displacement at the current position, and N_t is the number of trajectories.

The future displacement $d_\star = (\Delta x_\star,\, \Delta y_\star)$ of a given trajectory $\tau_\star$ can be predicted as follows [21]:

$$d_\star = k_t(\tau_\star,\, \tau_{1:N_t})^T \Gamma_t, \quad (4)$$

where

$$\Gamma_t = (k_t(\tau_{1:N_t}, \tau_{1:N_t}) + \sigma_w^2 I)^{-1} d_{1:N_t}. \quad (5)$$

Note that we precompute (5) to reduce the required computation time during the run time. Figure 2(c) shows predicted trajectories using an AR-GPMM.

B. Autoregressive Gaussian Process Motion Controller

This section proposes a Gaussian process motion controller which can avoid pedestrians with only information gained from the egocentric view of a robot. A number of related navigation algorithms require trajectories of obstacles in the reference (overhead-view) coordinate. However, humans can navigate through a dense and crowded environment with only local information obtained from the egocentric view. Our motivation starts from the idea that if we can capture how a human reacts to an incoming object, then it will be possible to implement a human-like navigation algorithm for a crowded and dynamic environment.

A motion controller is a mapping $\mathcal{F}_u : \mathcal{T} \to \mathcal{U}$ from a trajectory space $\mathcal{T} \in \mathbb{R}^{2p}$, where p is the length of a trajectory, to a control input space $\mathcal{U} \subset \mathbb{R}^2$. Gaussian process regression is again used to control a robot based on an AR-GP model. An input of an autoregressive Gaussian process motion controller (AR-GPMC) is the past trajectory of length p in the polar coordinate. The polar coordinate is known to capture the relationship between an operating robot and pedestrians effectively [9].

The covariance function for an AR-GPMC is as follows:

$$\begin{aligned} k_u(\tau,\, \tau' | \,\theta_u) &= \sigma_f^2 \exp\left(-\sum_{i=1}^{p} \frac{(R_i' - R_i)^2}{2\sigma_R^2} + \frac{g(d_i' - d_i))^2}{2\sigma_d^2}\right) \\ &+ \sigma_w^2 \delta(\tau,\, \tau'), \end{aligned} \quad (6)$$

where an element of trajectory τ is now $(R,\, d)$, R is the distance from a robot to a pedestrian, d is the angle from the y-axis to the pedestrian, $\theta_u = (\sigma_f^2,\ \sigma_R^2,\ \sigma_d^2,\ \sigma_w^2)$ are hyperparameters, and $\delta(\tau, \tau') = 1$ if $\tau = \tau'$ and zero otherwise. $g(d) = d$ if d is smaller than 180° and $d - 360^\circ$ otherwise.

To obtain training data for an AR-GPMC, i.e., pedestrian trajectories and corresponding robot control inputs, we implemented a simulator. A human operator manually controls a robot to avoid incoming virtual pedestrians in a simulated environment using a keyboard and the simulator collects positions of virtual pedestrians and robot control inputs from the human operator as a training set $D_u = \{(\tau_i,\, u_i) \mid i = 1, 2, \cdots, N_u\}$, where $u_i = (v,\, w) \in \mathbb{R}^2$ is a control input and N_u is the number of examples.

A robot control is generated by an AR-GPMC as follows:

$$u_\star = k_u(\tau_\star,\, \tau_{1:N_u})^T \Gamma_u, \quad (7)$$

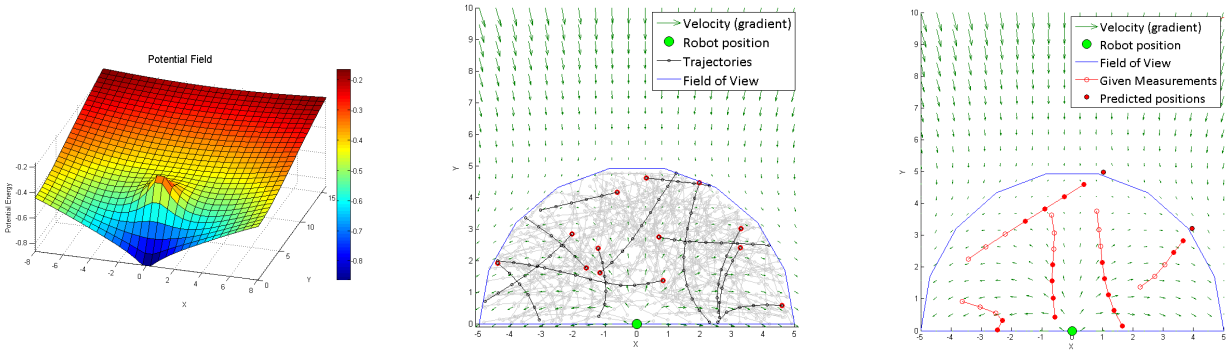
where

$$\Gamma_u = (k(\tau_{1:N_u}, \tau_{1:N_u}) + \sigma_w^2 I)^{-1} d_{1:N_u}. \quad (8)$$

Note that we also compute (8) in advance to avoid the computation of the inverse of the kernel matrix.

Since we do not assume any additional tracking system in our proposed framework, inputs to our motion controller are current position measurements $x_{ped} \in \mathbb{R}^2$ of detected pedestrians in an egocentric view (field of view). A nearest-neighbor tracking algorithm is used to handle multiple trajectories generation by assigning a detected pedestrian to the closest known trajectory, but a more sophisticated multi-target tracking algorithm, such as [22], can be applied. A new trajectory is formed, if there is no nearby trajectory.

In the egocentric view approach, the view of a robot is constantly changing as a robot moves, and it makes hard to obtain continuous trajectories. However, motion planning considering the current measurement only is likely to fail to provide safe navigation. In our approach, we tightly integrates an AR-GPMM by predicting the positions of pedestrians outside the current field of view (FOV). By using



(a) (b) (c)

Fig. 2: (a) Potential field. (b) Generated trajectories using the potential field from (a). A few trajectories are in a darker color for better visualization. (c) Examples of trajectories predicted by an AR-GPMM.

both observed and predicted trajectories, the AR-GPMC can effectively overcome the limitation of egocentric view approaches. When more than one pedestrian is detected, we first find a control input for each pedestrian and assign a different weight for each control input based on the distance between a robot and a corresponding pedestrian. The AR-GPMC algorithm is described in Algorithm 1.

Algorithm 1 Autoregressive Gaussian Process Motion Controller (AR-GPMC)

Input θ_u, Γ_u, $\tau_{1:N_{ped}}$, and x_{robot}
Output u_{robot}: A robot control
Update $\tau_{1:N_{ped}}$.
$sum_w = 0$
$sum_u = [0, 0]$
for every τ_i in $\tau_{1:N_{ped}}$
  d_i = calculateDistance(x_{robot}, τ_i)
  w_i = weight(d_i)
  $\tau_{egoPloar}$ = EgocentricPolar(τ_i, x_{robot})
  $u_\star = \mathcal{F}_u(\tau_{egoPloar} | \theta_u, \Gamma_u)$ using (7)
  $sum_w = sum_w + w_i$
  $sum_u = sum_u + u_\star$
end
$u_{robot} = sum_u / sum_w$

IV. Simulation

In this section, we perform a number of comparative simulations to validate the performance of our proposed navigation algorithm.

A. Setup

To make a realistic setup, we assume that a robot can only obtain information from its egocentric view. The field of view (FOV) is set to 60° and this is due to the fact that Microsoft Kinect used in our experiments has 57° FOV. The number of pedestrians varies from one to seven to validate the safety of the proposed method under different conditions. For each setting, 100 independent runs are performed.

For each simulation, the goal is to reach the goal position. For comparison, we apply three different navigation algorithms: an AR-GPMC, a reactive planner, and vector field histogram (VFH). When there is no pedestrian, the robot tries to reach the goal using pure pursuit (PP) low-level controller [23].

B. Compared Navigation Algorithms

1) Reactive Planner: This planner first predicts future trajectories of pedestrians using our AR-GPMM. Then it randomly generates 20 trajectories and calculates the cost of each trajectory. The cost function is defined as the minimum distance between a randomly generated trajectories and predicted pedestrian trajectories. The minimum cost control input is applied at each time. This approach is similar to [13], [19] in that it makes a locally optimal decision using the model predictive control framework.

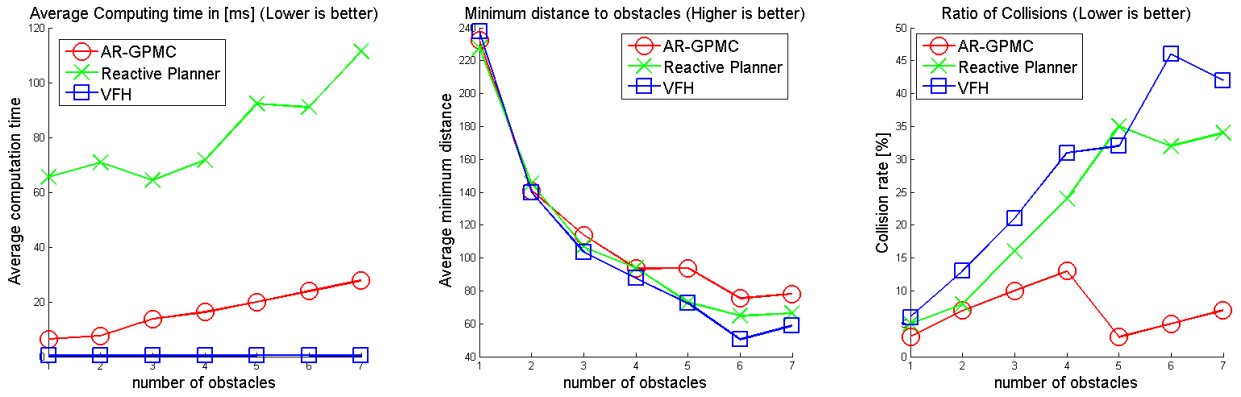
2) Vector Field Histogram (VFH) [8]: VFH is used as the baseline navigation algorithm. Since VFH controls a robot based on the current measurement only, it needs a relatively large FOV to provide reasonable performance. In VFH, we set the FOV to be 160° assuming that the robot uses a laser range finder.

3) Discussion of Untested Navigation Algorithms: In Section II, a number of navigation algorithms for dynamic environments are listed. However, algorithms categorized under the reference-view approach are not compared because our goal of this work is to develop a navigation algorithm which can be applied to a wide range of situations without the help of an expensive tracking system.

A potential field method is not compared since it requires tedious parameter tuning. Moreover, Koren and Borenstein [10] have discussed substantial shortcomings of potential field based methods and proposed VFH to overcome discussed shortcomings.

C. Results

Three different criteria have been used to evaluate the performance of the proposed algorithm: the average computation



(a) (b) (c)

Fig. 3: Simulation results of three navigation algorithms. (a) Average computation times in millisecond. (b) Average minimum distances to obstacles. (c) Collision rates. Results are computed from 100 independent runs.

time, the average minimum distance to the nearest object, and the collision ratio. The overall results are shown in Figure 3. The proposed method outperforms VFH and the reactive planner in terms of the average minimum distance to the nearest object and the collision ratio and its computation time is relatively small, requiring only $30\,ms$ for the case with seven pedestrians. Figure 3(b) shows the average minimum distance of each algorithm. The average minimum distance calculates the average of distances to the nearest pedestrian encountered at each trial. It measures how close a robot moves in front of a pedestrian. Our proposed algorithm showed the best performance because the AR-GPMC can make a human-like control by predicting the future trajectories of pedestrians. The average collision rate shown in Figure 3(c) calculates the collision rate from 100 trials and it is the most important measure. The proposed AR-GPMC achieves 6.86% average collision rate while the reactive planner and VFH achieve 22% and 27.29%, respectively. The proposed method shows over 68% improvement compared to other approaches.

We found that AR-GPMC can make some interesting motions. One is to go backward if a pedestrian is coming too closely. Since a typical differential drive mobile robot has a nonholonomic constraint (robot can only move in the direction normal to the axis of the driving wheels), we controlled a robot to go backward when it cannot avoid incoming pedestrian by going forward in the training stage. Since AR-GPMC makes control based on the human control data, the robot showed similar motion patterns.

V. Experiments

A. Setup

We used a Pioneer 3DX differential drive mobile robot with a Microsoft Kinect camera mounted on top of the robot as shown in Figure 4. All programs are written in MATLAB using mex-compiled ARIA package. The position of pedestrians are detected using the skeleton grab API of a Kinect camera. Pedestrian tracking and the proposed navigation algorithm runs at about $5\,$Hz on a $2.1\,$GHz notebook. Eight sonar sensors equipped in the Pioneer platform are used to avoid a collision. This exception routine using sonar sensors is activated if the minimum distance between the robot and pedestrians are less that 40 cm.

Fig. 4: A Pioneer 3DX mobile robot with a Kinect camera.

TABLE II: Experimental results

	AR-GPMC
Collision rate	12.21 %
Average minimum distance	64.97 cm

B. Results

We have performed 34 experiments in our laboratory. The results are summarized in Table II. If the minimum distance between a robot and pedestrians is smaller than $40cm$, we declare a collision has occurred. From 34 experiments, the average minimum distance was 64.97cm and the collision rate was only 12.21%.

We have also tested our algorithm in an L-shaped corridor and a school cafeteria. The goal of this experiment was to follow a given path while avoiding dynamic or static pedestrians. Some snapshots from the experiment are shown in

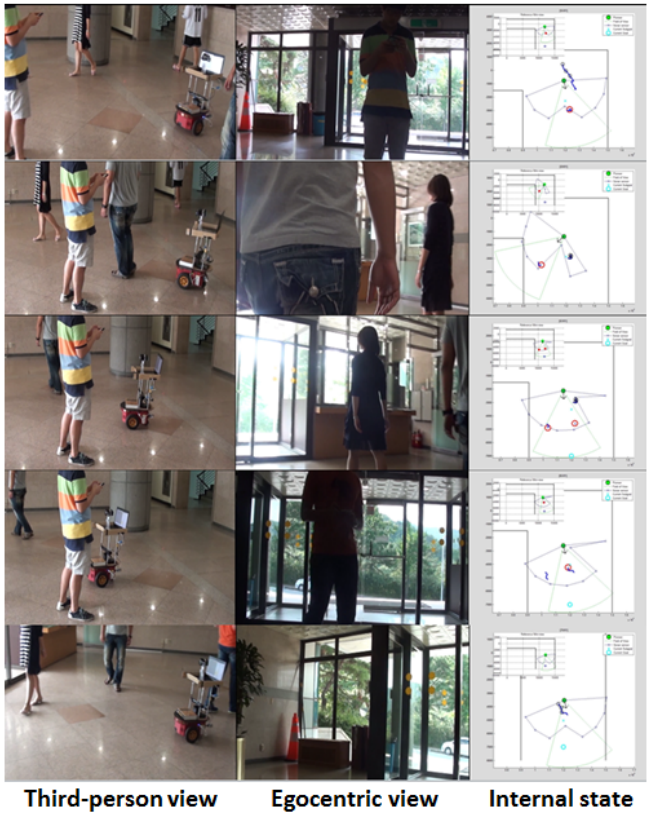


Fig. 5: Snapshots from the L-shaped corridor experiment. Each snapshot shows a photo taken by a third person, a photo taken by the robot, and the robot's internal state visualized on the floor plan. Note that the robot does not know the floor plan or the map of the area.

Figure 5. The robot successfully avoided moving pedestrians in both cases and arrived at the goal location.

VI. Conclusion

In this paper, we have proposed a novel Gaussian process motion controller which can navigate through a crowded dynamic environment. The proposed motion controller predicts future trajectories of pedestrians using an autoregressive Gaussian process motion model (AR-GPMM) and controls a robot using an autoregressive Gaussian process motion controller (AR-GPMC) based on predicted trajectories. The performance of the proposed method is validated in a number of simulations and physical experiments in a laboratory, an L-shape corridor, and a school cafeteria. In particular, the proposed method shows over 68% improvement in terms of the collision rate compared to a reactive planner and VFH. By using only data from the egocentric view of a robot, our algorithm can be easily implemented and applied in any environment without additional equipments.

References

[1] C. H. Chen, Y. Weng, and C.-T. Sun, "Toward the human-robot co-existence society: on safety intelligence for next generation robots," *International Journal of Social Robotics*, vol. 1, no. 4, pp. 267–282, 2009.

[2] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *Proc. of the International Conference of Robotics and Automation (ICRA)*, 2013.

[3] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, pp. 1–26, 2013.

[4] B. Luders, M. Kothari, and J. P. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2010.

[5] N. Pradhan, T. Burg, and S. Birchfield, "Robot crowd navigation using predictive position fields in the potential function framework," in *American Control Conference (ACC)*. IEEE, 2011, pp. 4628–4633.

[6] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Proc. of the International Conference of Robotics and Automation (ICRA)*. IEEE, 2010, pp. 981–986.

[7] C.-P. Lam, C.-T. Chou, K.-H. Chiang, and L.-C. Fu, "Human-centered robot navigation towards a harmoniously human–robot coexisting environment," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 99–112, 2011.

[8] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[9] I. Ulrich and J. Borenstein, "VFH*: local obstacle avoidance with look-ahead verification," in *Proc. of the International Conference of Robotics and Automation (ICRA)*, vol. 3. IEEE, 2000, pp. 2505–2511.

[10] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. of the International Conference of Robotics and Automation (ICRA)*. IEEE, 1991, pp. 1398–1404.

[11] D. Bodhale, N. Afzulpurkar, and N. T. Thanh, "Path planning for a mobile robot in a dynamic environment," in *Proc. of the International Conference on Robotics and Biomimetics*. IEEE, 2009, pp. 2115–2120.

[12] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *Proc. of the International Conference of Robotics and Automation (ICRA)*. IEEE, 2007, pp. 1603–1609.

[13] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 4293–4298.

[14] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A Bayesian nonparametric approach to modeling motion patterns," *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[15] G. S. Aoude, J. Joseph, N. Roy, and J. P. How, "Mobile agent trajectory prediction using Bayesian nonparametric reachability trees," in *Proc. of the AIAA Infotech*, 2011.

[16] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 797–803.

[17] 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 International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 3931–3936.

[18] F. Rohrmuller, M. Althoff, D. Wollherr, and M. Buss, "Probabilistic mapping of dynamic obstacles using markov chains for replanning in dynamic environments," in *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008, pp. 2504–2510.

[19] J. J. Park, C. Johnson, and B. Kuipers, "Robot navigation with model predictive equilibrium point control," in *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 4945–4952.

[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[21] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. Cambridge, MA: MIT Press, 2006.

[22] S. Oh, S. Russell, and S. Sastry, "Markov chain Monte Carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, 2009.

[23] O. Amidi and C. E. Thorpe, "Integrated mobile robot control," in *Fibers' 91, Boston, MA*. International Society for Optics and Photonics, 1991, pp. 504–523.