Anticipatory Robot Navigation: Incorporating Estimated Obstacle Behaviors with the Social Force Model

Fengkai Liu*, Yuichi Ohsita†, Kenji Kashima‡, Shinya Yasuda§, Taichi Kumagai§,
Hiroshi Yoshida§, Dai Kanetomo§, Masayuki Murata*
*Graduate School of Information Science and Technology, Osaka University, Osaka, Japan 565-0871
Email: {f-liu, murata}@ist.osaka-u.ac.jp
†Cybermedia Center, Osaka University, Osaka, Japan 560-0043
Email: yuichi.ohsita.cmc@osaka-u.ac.jp
‡Graduate School of Informatics, Kyoto University, Kyoto, Japan 606-8317
Email: kk@i.kyoto-u.ac.jp
§Visual Intelligence Research Laboratories, NEC Corporation, Kanagawa, Japan 211-8666
Email: {shinya-yasuda, t_kumagai, yoshida, d-kanetomo}@nec.com

***Abstract*—In environments populated by various dynamic obstacles, such as people and robots, it is essential for each robot to operate efficiently while avoiding these obstacles. Therefore, in this paper, we propose a method for navigating a robot to avoid obstacles and ensure smooth and efficient movement in dynamic environments. Moving obstacles may also actively attempt to avoid the robots. Therefore, our navigation method should consider such interactions to achieve an efficient movement. To achieve this, our method integrates principles from the Social Force Model and models the movement of nearby obstacles by estimating the parameter of the Social Force Model. Then, our method controls the robot to ensure a balance between efficiency and safety based on the model. We evaluate our method by simulation and demonstrate that it achieves high efficiency and safety by utilizing a model of nearby obstacles.**

***Index Terms*—Obstacle Avoidance, Obstacle Behavior, Social Force Model**

I. Introduction

With the continuous growth of the e-commerce and retail sectors, there's an increasing demand for automation within warehouses and logistic centers. Being central to this automation transition, the robot, specifically the Automated Guided Vehicle (AGV), offers efficiency for warehouse operations, especially in tasks like material transport. However, the warehouse environment is unique, with robots interacting with moving obstacles, e.g., humans or robots operating in other systems, presenting distinctive challenges for robot navigation. Particularly, given the efficiency demands of warehouse operations, the movement efficiency of the robot and safety become paramount.

Traditionally, in situations with a possibility of collision, actions like stopping or significantly limiting speed have been taken to ensure human safety [1]. Several approaches have been proposed in the past, for example, Feder et al. [2] presented a method for real-time path planning in dynamic environments using harmonic potentials. Arslan et al. [3] proposed a robot obstacle avoidance method based on power diagrams. Missura et al. [4] enhanced the Dynamic Window Approach and incorporated a dynamic collision model that predicts future collisions. However, neither of these methods considers the potential impact of the robot's movement on the trajectory of obstacles, indicating that they lack interaction-awareness. Such neglect could lead the robot to perceive a greater intrusion by obstacles into its trajectory, prompting it to adopt a more conservative control strategy. As a result, these methods may introduce inefficiencies and, in extreme scenarios, give rise to the Freezing Robot Problem [5].

Therefore, several interaction-aware methods have been proposed. Helbing et al. [6], [7] introduced the Social Force Model, a simple yet effective framework for describing pedestrian movement adaptable to a myriad of contexts, which was also used in robot navigation. Ferrer et al. [8] presented a human-aware navigation approach based on the Social Force Model. Shiomi et al. [9] improved the Social Force Model based approach by introducing the concept of "social distance" and respecting personal spaces, aiming for human-like collision avoidance in robots. Kamezaki et al. [10] enhanced the Social Force Model by adding an inducible feature to handle the Freezing Robot Problem. However, a common limitation across the methods in the approaches presented in [8]–[10] is the use of fixed parameter values. This assumes uniform repulsive forces across all objects, implying that all obstacles exhibit the same behavioral pattern. However, each obstacle has its own behavior. Some obstacles may change their trajectories to avoid the robot, while other obstacles may not change their trajectories. If an obstacle does not avoid the robot but the robot assumes it will, this mismatch can result in a collision.

An interaction-aware method based on reinforcement learning has also been proposed [11]. This method trains a model to control a robot without violating social norms by using a

trajectory dataset. However, this method also does not consider the variation in the behavior of the obstacles.

Hence, in this paper, we propose a method for navigating a robot to avoid obstacles and ensure smooth and efficient movement by considering the interaction between the robot and obstacles. In our method, we use the Social Force Model to model the behavior of obstacles. By estimating the parameter of the Social Force Model based on the observations of obstacles, our method reflects the varied behaviors exhibited by different obstacles. While we apply the Social Force Model to prevent collisions, its parameters are adjusted in our system to achieve a balance between safety and efficiency, factoring in the behaviors of surrounding obstacles. By repeatedly updating the models of the behaviors of nearby obstacles and updating the parameters for controlling robots, our method achieves safety and efficiency in dynamic environments.

Our primary contributions in this study are:

1) We propose a method to model the behaviors of each obstacle that avoids collisions with the robot by using observations of obstacles.
2) We also propose a method to refine the robot's control input, ensuring a balance between efficiency and safety, based on the model of the behaviors of nearby obstacles.

In the following sections, we elaborate on our methodology. Section II delves into the Social Force Model. Section III outlines our approach for calculating the control input, while Section IV-B presents the experimental setup, results, and discussions related to our evaluation.

II. Social Force Model

Social Force Model is a simple model designed to describe pedestrian motion, and takes into account individual preferences, local intentions with neighbors, and the surrounding environment. In this model, interactions with other objects influence the moving velocity. This influence is defined in terms of repulsive forces. One object also experiences a simulated attractive force originating from destinations or places they intend to go to, guiding them towards their target. Figure 1 shows the overview of the Social Force Model. By utilizing the principles of the Social Force Model, we can model the behavior of the robot and the obstacles within a shared space, especially in contexts where their paths might intersect or come into proximity.

In the Social Force Model, we can use (1) to determine the position of object i for the subsequent time step $t + \Delta t$.

$$\boldsymbol{r}_i(t + \Delta t) = \boldsymbol{r}_i(t) + \boldsymbol{v}_i(t) \cdot \Delta t + \frac{1}{2}\boldsymbol{a}_i(t) \cdot \Delta^2 t, \quad (1)$$

where $\boldsymbol{r}_i(t)$ and $\boldsymbol{v}_i(t)$ are the position and velocity of object i at time step t, respectively. In the Social Force Model, the acceleration $\boldsymbol{a}_i(t)$ is obtained by (2).

$$\boldsymbol{a}_i(t) = \sum_{j \neq i} \boldsymbol{F}_{i,j}^{rep}(t) + \boldsymbol{F}_i^{att}(t), \quad (2)$$

where $\boldsymbol{F}_{i,j}^{rep}(t)$ represents the repulsive force exerted between object i and any distinct obstacle j, and $\boldsymbol{F}_{i,j}^{att}(t)$ denotes the attractive force directing object towards its designated destination. We accumulated all the repulsive forces $\boldsymbol{F}_{i,j}^{rep}(t)$ that arise between object i and any obstacles.

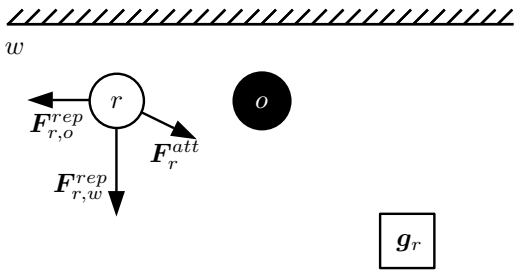

Fig. 1. An example for Social Force Model. In this scenario, a robot r navigates in a space with a moving obstacle o and a wall w, which serves as a static obstacle. Therefore, there are three forces acting on robot r: two repulsive forces exerted by moving obstacle o, $\boldsymbol{F}_{r,o}^{rep}$ and wall w, $\boldsymbol{F}_{r,w}^{rep}$, and the attractive force $\boldsymbol{F}_r^{att}$ directed towards its goal $\boldsymbol{g}_r$.

$\boldsymbol{F}_{i,j}^{rep}(t)$ is the repulsive force to act the interaction, calculated by (3), which is a simplified version of the Social Force Model from [7], often referred to as the circular specification.

$$\boldsymbol{F}_{i,j}^{rep}(t) = A_i \cdot e^{-||\boldsymbol{d}_{i,j}(t)||/B} \frac{\boldsymbol{d}_{i,j}(t)}{||\boldsymbol{d}_{i,j}(t)||}, \quad (3)$$

where $\boldsymbol{d}_{i,j}(t)$ means the distances between the object i and object j, calculated by $\boldsymbol{d}_{i,j}(t) = \boldsymbol{r}_i(t) - \boldsymbol{r}_j(t)$. The parameter A_i determines the strength of repulsive force exerted on object i, which can be used to flexibly depict the behavior of object i. The parameter B determines the range of the repulsive force. In this paper, by setting A_i for each object, we capture its tendency to avoid obstacles.

$\boldsymbol{F}_i^{att}(t)$, the attractive force, is calculated by (4), using a specific variable $\boldsymbol{v}_i^0(t)$ that represents the desired velocity.

$$\boldsymbol{F}_i^{att}(t) = \frac{1}{\tau}\big(\boldsymbol{v}_i^0(t) - \boldsymbol{v}_i(t)\big), \quad (4)$$

where τ represents the relaxation time. The desired velocity $\boldsymbol{v}_i^0(t)$ can be calculated by (5) from the concept in [6].

$$\boldsymbol{v}_i^0(t) = \boldsymbol{v}_i^{max}(t) = v_i^{max} \cdot \frac{\boldsymbol{g}_i - \boldsymbol{r}_i(t)}{||\boldsymbol{g}_i - \boldsymbol{r}_i(t)||}, \quad (5)$$

where the notion v_i^{max} represents the maximum velocity, while $\boldsymbol{g}_i$ denotes the goal of the object i.

For static obstacles, we only employ (3) to compute the repulsive force between the object and the static obstacle and do not use other equations to calculate the displacement of static obstacles.

III. Control of the Robot

In this section, we explain our assumption of the environment. Then, we explain the overview of our method. Following that, we delve into the mechanisms by which the robot processes information about obstacles to ascertain the appropriate control input.

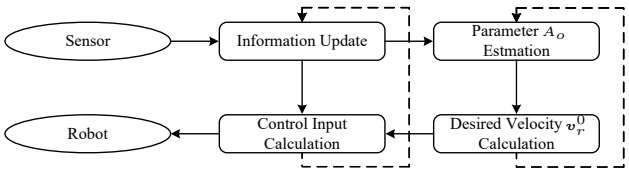

Fig. 2. The procedure of control.

A. Assumed Environment

In our setup, we assume robot r can observe the positions of obstacle o as $\widetilde{\boldsymbol{r}}_o(t)$ by sensors installed within the environment or sensors mounted on the robot. Similarly, the obstacle can observe the position of the robot as $\boldsymbol{r}_r(t)$. We also assume that the robot's observation of the obstacle's position $\widetilde{\boldsymbol{r}}_o(t)$ contains noises, potentially due to sensor precision. That is, $\widetilde{\boldsymbol{r}}_o(t)$ is represented by

$$\widetilde{\boldsymbol{r}}_o(t) = \boldsymbol{r}_o(t) + \boldsymbol{e}(t), \tag{6}$$

where $\boldsymbol{r}_o(t)$ is the actual location of o at time t, and $\boldsymbol{e}(t)$ is errors. We assume that the error term $\boldsymbol{e}(t)$ follows a multivariate normal distribution where the components in each dimension are independent. To simplify our analysis, we consider the errors included only in the observations by the robot and do not consider the errors included in the observations by the obstacles.

B. Method Overview

Our method employs the model introduced in Section II to compute interaction and the robot's control input. Using the movement model, we assume both the robot r and the obstacle o have their own goals, denoted as $\boldsymbol{g}_r$ and $\boldsymbol{g}_o$. respectively. We assume that we can use the goal for the obstacle estimated by the methods proposed in [12], [13]. We also assume that both the robot and the obstacle will make every effort to move towards their destinations, without exhibiting unreasonable behaviors like getting lost.

Our method operates based on a distinct workflow, comprising two cycles, as shown in Figure 2. The first cycle focuses on determining the avoidance behavior and is characterized by its shorter duration. This cycle involves obtaining the observation from sensors, as well as calculating the control input for the robot based on the Social Force Model. The second, longer cycle primarily deals with determining the parameters needed to compute the robot's control input. This not only includes the parameter value A_o that reflects the tendency of obstacles to avoid the robot, but also the desired velocity $\boldsymbol{v}_r^0$ used to calculate the robot's attraction. The desired velocity $\boldsymbol{v}_r^0$ affects the attractive force from the goal. That is, by setting the desired velocity, we can strike a balance between the attractive force and the repulsive force from obstacles.

We will delve into further details in the subsequent Section III-C and Section III-D.

C. Avoidance Behavior based on the Social Force Model

At each time step t, the system first employs the Kalman filter to refine the observed position of an obstacle, and then calculates the control input of the robot by integrating forces derived from the Social Force Model.

1) Estimation for the position of obstacle: At a discrete time step t, upon observing the position of the obstacle, we use the Kalman filter to minimize the impact of observation errors. The filter refines this prediction by (7), using the newly observed position, $\widetilde{\boldsymbol{r}}_o(t)$, to produce an estimation of the true value, $\widehat{\boldsymbol{r}}_o(t)$.

$$\widehat{\boldsymbol{r}}_o(t) = \widehat{\boldsymbol{r}}'_o(t) + \Big[\boldsymbol{K}(t) \cdot \big(\widetilde{\boldsymbol{r}}_o(t) - \widehat{\boldsymbol{r}}'_o(t)\big)\Big], \tag{7}$$

where $\widehat{\boldsymbol{r}}'_o(t)$ is the predicted value from the previous time step, while $\boldsymbol{K}(t)$ is the Kalman gain, calculated by (8).

$$\boldsymbol{K}(t) = \frac{\boldsymbol{P}(t)}{\boldsymbol{P}(t) + \boldsymbol{R}}, \tag{8}$$

where $\boldsymbol{P}$ and $\boldsymbol{R}$ are pre-defined covariances matrix. After this step, we update the covariance matrix $\boldsymbol{P}$ in preparation for the next prediction by (9).

$$\boldsymbol{P}(t + \Delta t) = (\boldsymbol{I} - \boldsymbol{K}(t))\boldsymbol{P}(t), \tag{9}$$

We then predict the obstacle's position for the subsequent time step $\widehat{\boldsymbol{r}}'_o(t + \Delta t)$. In the prediction period, we use the model we introduced in Section II, but use (10) to estimate the obstacle's velocity.

$$\widehat{\boldsymbol{v}}_o(t) = \frac{\widehat{\boldsymbol{r}}_o(t) - \widehat{\boldsymbol{r}}_o(t - \Delta t) + 1/2 \cdot \widehat{\boldsymbol{a}}_o(\Delta t) \cdot \Delta^2 t}{\Delta t} \tag{10}$$

This prediction is aimed at refining the subsequent time step's estimation at $t + \Delta t$, as we showed in (7).

2) Behavior based on Social Force Model: Using the estimated position of the obstacle $\widehat{\boldsymbol{r}}_o(t)$, and the model from Section II, we can utilize (3) to compute the repulsive forces and (4) to calculate the attractive force acting on the robot. Once both forces are obtained, the actual control input $\boldsymbol{v}_r(t + \Delta t)$ is computed using (11).

$$\boldsymbol{v}_r(t + \Delta t) = \boldsymbol{v}_r(t) + \boldsymbol{a}_r(t)\Delta t, \tag{11}$$

where the desired velocity used in (4) is determined using the method detailed in Section III-D2.

D. Control Considering the Avoidance Behavior of Obstacle

Within this discussion, we delve into a control strategy that addresses the evasive actions of dynamic obstacles. This includes estimating the tendency of nearby obstacles to avoid the robot and the calculation of the desired velocity of the robot. First, based on the observed obstacle position, we infer the parameter value A_o of the repulsive force acting on the obstacle, in (3). Then, using the estimated parameter value $\widehat{A}_o$, we compute the desired velocity $\boldsymbol{v}_r^0$, ensuring the formulation of a harmonious path. In the estimation step, we use the observed obstacle position instead of the optimized outcomes from the Kalman filter, aiming to minimize the influence of the Kalman filter's behavior on the estimation results.

1) Parameter Estimation for the Model: Using Bayesian estimation (12), we estimate the value of the parameter A_o.

$$P(A_o^i|\widetilde{\boldsymbol{r}}_o(t),\widetilde{\boldsymbol{r}}_o(t-\Delta t)) = \frac{L\big(\widetilde{\boldsymbol{r}}_o(t)|A_o^i,\widetilde{\boldsymbol{r}}_o(t-\Delta t)\big)\cdot P(A_o^i)}{\sum_i L\big(\widetilde{\boldsymbol{r}}_o(t)|A_o^i,\widetilde{\boldsymbol{r}}_o(t-\Delta t)\big)\cdot P(A_o^i)}, \quad (12)$$

where A_o^i is a value sourced from the prior probability distribution. This prior is derived from the posterior probability of the previous time step. We set the initial distribution of A_o^i to a normal distribution $\mathcal{N}(0,\sigma^2)$, assuming that the obstacle might not change its trajectory. $L\big(\widetilde{\boldsymbol{r}}_o(t)|A_o^i,\widetilde{\boldsymbol{r}}_o(t-\Delta t)\big)$ is the value of the likelihood function defined by

$$L\big(\widetilde{\boldsymbol{r}}_o(t)|A_o^i,\widetilde{\boldsymbol{r}}_o(t-\Delta t)\big) = f(\widetilde{\boldsymbol{r}}_o(t);\widetilde{\boldsymbol{r}}_{o,i}(t),\boldsymbol{\Sigma}), \quad (13)$$

where f is the probability density function of the normal distribution of $\widetilde{\boldsymbol{r}}_o(t)$, given by $\widetilde{\boldsymbol{r}}_o(t)\sim\mathcal{N}\big[(\widetilde{\boldsymbol{r}}_o(t-\Delta t)+\widehat{\boldsymbol{v}}_o(t-\Delta t)\cdot\Delta t+\frac{1}{2}\widehat{\boldsymbol{a}}_{o,i}(t-\Delta t)\cdot\Delta^2 t),\boldsymbol{\Sigma}\big]$, with the error terms $\boldsymbol{e}(t)$ is assumed to follow a same normal distribution $\mathcal{N}(0,\boldsymbol{\Sigma})$. $\boldsymbol{\Sigma}$ is the covariance matrix of the error term's components in each dimension. $\widetilde{\boldsymbol{r}}_{o,i}(t)$ determined based on the sampled A_o^i by

$$\widetilde{\boldsymbol{r}}_{o,i}(t) = \big(\widetilde{\boldsymbol{r}}_o(t-\Delta t)-\boldsymbol{e}(t-\Delta t)\big)+ \widehat{\boldsymbol{v}}_o(t-\Delta t)\cdot\Delta t+\frac{1}{2}\widehat{\boldsymbol{a}}_{o,i}(t-\Delta t)\cdot\Delta^2 t-\boldsymbol{e}(t), \quad (14)$$

where $\widehat{\boldsymbol{a}}_{o,i}(t-\Delta t)$ is calculated by (2) using A_o^i.

After the calculation on time step t, we determine the estimated value $\widehat{A}_o$ from the expectation of the posterior probability distribution, and we then treat the posterior probability distribution as the prior probability distribution on the next time step $t+\Delta t$.

2) Calculation of the Desired Velocity: Given the position of the robot $\boldsymbol{r}_r(t)$ and the Kalman filter corrected position of the obstacle $\widehat{\boldsymbol{r}}_o(t)$ at current time step t, we calculate the desired velocity used to calculate the robot's control input by considering their future interactions. The desired velocity $\boldsymbol{v}_r^0$ for robot, used in (4), is computed by minimize objective function $\mathcal{J}([\boldsymbol{v}_r^0]_t^{t+(H-1)\Delta t})$ in (15), where the independent variable $[\boldsymbol{v}_r^0]_t^{t+(H-1)\Delta t}$ denotes the sequence of desired velocity vectors from time t to $t+(H-1)\Delta t$. For brevity, we denote the array of the desired velocity as $[\boldsymbol{v}_r^0]$ in subsequent equations. In our objective function, we aim to minimize both the time the robot takes to reach its destination and the risk of collision with the obstacle.

$$minimize \quad \mathcal{J}([\boldsymbol{v}_r^0]) = \mathcal{J}_t([\boldsymbol{v}_r^0]) + \mathcal{P}_v([\boldsymbol{v}_r^0]) + \mathcal{P}_d([\boldsymbol{v}_r^0]) \quad (15)$$

This objective function consists of a function $\mathcal{J}_t([\boldsymbol{v}_r^0])$ that calculates the required time to the destination for the robot, a penalty function $\mathcal{P}_v([\boldsymbol{v}_r^0])$ to ensure movement efficiency, and a penalty function $\mathcal{P}_d([\boldsymbol{v}_r^0])$ to ensure safety.

We define the function $\mathcal{J}_t([\boldsymbol{v}_r^0])$ by

$$\mathcal{J}_t([\boldsymbol{v}_r^0]) = \left|\frac{||\boldsymbol{g}_r-\boldsymbol{r}_r(t+H\Delta t,[\boldsymbol{v}_r^0])||}{\mathcal{V}\big(\boldsymbol{v}_r(t+H\Delta t,[\boldsymbol{v}_r^0]),\boldsymbol{g}_r,\boldsymbol{r}_r(t+H\Delta t,[\boldsymbol{v}_r^0])\big)+\varepsilon}\right| + \Delta t\cdot\sum_{i=1}^{H}\Big[\mathcal{I}\big(\boldsymbol{g}_r,\widehat{\boldsymbol{r}}_r(t+i\Delta t)\big)\Big], \quad (16)$$

where $\boldsymbol{r}_r(t+H\Delta t,[\boldsymbol{v}_r^0])$ and $\boldsymbol{r}_r(t+H\Delta t,[\boldsymbol{v}_r^0])$ denote the position and velocity of robot. Both of them are generated based on the model introduced in Section III-B. $\mathcal{I}(\boldsymbol{g},\boldsymbol{r})$ is 1 if $[\boldsymbol{g}\neq\boldsymbol{r}]$, otherwise 0. The function $\mathcal{V}(\boldsymbol{v},\boldsymbol{g},\boldsymbol{r})$ calculates the projection of the velocity v on the vector formed by the destination g and the position r,

$$\mathcal{V}(\boldsymbol{v},\boldsymbol{g},\boldsymbol{r}) = \boldsymbol{v}\cdot\frac{\boldsymbol{g}-\boldsymbol{r}}{||\boldsymbol{g}-\boldsymbol{r}||+\varepsilon}, \quad (17)$$

where ε is a small constant added to prevent division by zero. To summarize the function $\mathcal{J}_t([\boldsymbol{v}_r^0])$, the first term of (16) calculate the required travel time to the destination for the robot r after a time horizon of $H\Delta t$, and the second term is added for the case if the robot arriving at the destination within the time horizon $H\Delta t$.

However, merely minimizing the function $\mathcal{J}_t$ might lead to a local optimum solution. Therefore, we add a penalty function related to the robot's movement velocity, $\mathcal{P}_v([\boldsymbol{v}_r^0])$, defined by

$$\mathcal{P}_v([\boldsymbol{v}_r^0]) = \sum_{i=t+\Delta t}^{t+H\Delta t}\Big[v_r^{max}-\mathcal{V}\big(\boldsymbol{v}_r(i,[\boldsymbol{v}_r^0]),\boldsymbol{g}_r,\boldsymbol{r}_r(i,[\boldsymbol{v}_r^0])\big)\Big]. \quad (18)$$

The addition of a penalty function related to the robot's movement velocity aims to ensure the velocity is maximized.

For safety, we also add a penalty function related to the distance between both subjects, $\mathcal{P}_d([\boldsymbol{v}_r^0])$ defined by

$$\mathcal{P}_d([\boldsymbol{v}_r^0]) = \sum_{i=t+\Delta t}^{t+H\Delta t} a\cdot e^{k_d\left(d_{min}-||\boldsymbol{d}_{r,o}(t,[\boldsymbol{v}_r^0])||\right)}, \quad (19)$$

where $||\boldsymbol{d}_{r,o}(t,[\boldsymbol{v}_r^0])||$ denotes the distance between the robot and the obstacle, calculated by on the model in Section II. a and k_d are parameters to determine the behavior of the function, and d_{min} is the parameter to decide the minimum distance between the robot and the obstacle. If the distance between both subjects becomes less than d_{min}, the value of the function rises sharply. By adding a distance-related penalty function, it's possible to avoid the computational issues of providing the minimum distance between both subjects as a constraint condition up to the time horizon $H\Delta t$.

The values of the two penalty functions, $\mathcal{P}_v([\boldsymbol{v}_r^0])$ and $\mathcal{P}_d([\boldsymbol{v}_r^0])$, will be repeatedly calculated until the time horizon $H\Delta t$.

IV. Evaluation

In this research, we conduct a simulation using the MATLAB Optimization Toolbox to validate the effectiveness of the proposed method. In the simulation, we simulate a robot and a moving obstacle. Subsequently, we analyze the minimum

TABLE I
SETTINGS FOR THE ROBOT AND THE OBSTACLE

	Obstacle o	Robot r
Initial position $\boldsymbol{r}(0)$	[1.5, 0.8]	[1.5, 2.2]
Initial velocity $\boldsymbol{v}(0)$	[0, 1]	[0, -1]
Goal g	[1.5, 2.2]	[1.5, 0.8]
Maximum speed v_{max}	$1.5m/s$	$1.5m/s$

TABLE II
SETTING OF THE PARAMETER VALUES

Parameter	Value
Time slot Δt	$0.1s$
Parameter to control the strength of repulsive force on robot A_r	20
Parameter to control the influence range of the repulsive force B	2
Relaxation time τ	$0.1s$
Observation error covariance $\boldsymbol{\Sigma}$	$0.1 \cdot \boldsymbol{I}$
Time horizon $H\Delta t$	$15\Delta t$
a in (19)	20
k_d in (19)	20
Minimum distance d_{min}	$1m$

distance between the robot and the moving obstacle during their interaction to demonstrate the safety of the proposed method. Additionally, we evaluate the robot's movement time to validate the movement efficiency of our approach.

A. Experimental Setup

We set a flat area of size $3m \times 3m$ as the evaluation environment, without static obstacles. The settings for the robot and the obstacle are presented in Table I. We ensure that the trajectories of the robot and the obstacle inevitably intersect, leading to an interaction between the two. We generated the obstacles to behave following the Social Force Model, introduced in Section II. In this evaluation, we simulate two cases; the case that the obstacle avoids the robot (this involves assuming the parameter A_o, used in (3) to decide the tendency to avoid the robot, is set to 20) and the case that the obstacle maintains its original trajectory without any deviation (this involves assuming the parameter $A_o = 0$). We generate the observation errors of the obstacle's position as follow a normal distribution $\mathcal{N}(0, 0.1^2)$. In this evaluation, we set the parameters as shown in Table II. We ran the simulation 100 times.

In this evaluation, we compare our method with a method that calculates the desired velocity using a fixed estimation $\widehat{A}_o$ to evaluate the significance of estimating A_o.

B. Results

1) Estimation of the obstacles behavior: Before comparing with the cases of fixed parameters, we investigate the parameter estimation result produced by our method. Figures 3 and 4 show examples of the estimated parameter at each time slot. Figure 3 indicates that $\widehat{A}_o$ is estimated to be a small value in the early time slots even when the actual value of $A_o = 20$. That is, our method considers a case where the obstacle may not change its trajectory. However, as the time

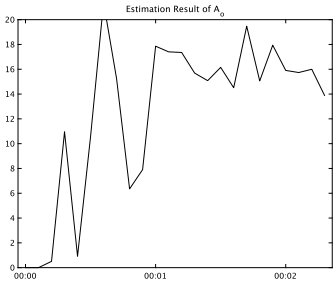

Fig. 3. Estimation result $\widehat{A}_o$ when the actual value of A_o is set to 20.

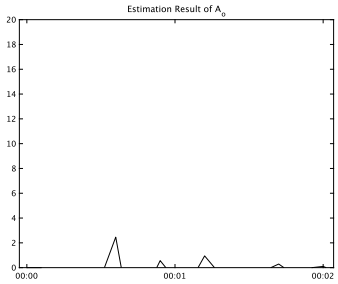

Fig. 4. Estimation result $\widehat{A}_o$ when the actual value of A_o is set to 0.

slot continues, $\widehat{A}_o$ is estimated to have a large value. This change is attributed to our method detecting that the obstacle is actively avoiding the robots. Consequently, our approach can predict the behavior of obstacles that maneuver to avoid robots. On the other hand, Figures 4 indicates that our method estimates $\widehat{A}_o$ to have a small value when the actual value of $A_o = 0$. As a result, our method can handle the obstacle as the obstacle that does not change its trajectory.

2) Result when the obstacle will avoid the robot ($A_o = 20$): Figure 5 presents the results for the scenario where the obstacle will avoid the robot, means that the actual value of A_o is set to 20. In this scenario, obstacles avoided the robot. By accurately predicting the behavior of the obstacles (in the case of the method with a fixed parameter $\widehat{A}_o = 20$), we can set an optimal strategy for controlling the robot to avoid collisions with obstacles and achieve efficient movement. In contrast, a method that assumes the obstacle does not change its trajectory, implying that the estimation of the parameter $\widehat{A}_o$ is fixed at 0, requires more substantial evasive actions. As a result, more time is required to reach its goal. Actually, Figure 5 shows that the minimum distance from the obstacle is slightly far and the movement time is large in the case of the method with the fixed parameter $\widehat{A}_o = 0$. Figure 5 also indicates that our method achieves similar movement time to the method with a fixed $\widehat{A}_o = 20$. This is because our method estimated $\widehat{A}_o$ as the values indicating that the obstacle avoids the robot.

3) Result when the obstacle will not avoid the robot ($A_o = 0$): Figure 6 presents the results for the scenario where the obstacle will not avoid the robot, means that the actual value of A_o is set to 0. In this scenario, the obstacle did not change its trajectory. In contrast, a method that assumes the obstacle will avoid the robot, implying that the estimation of the parameter $\widehat{A}_o$ is fixed at 20, results in ineffective robot control. Consequently, the minimum distance between the robot and the obstacle becomes less than 1, which violates our behavioral constraint, as shown in Figure 6(a). On the other hand, the method with the correct estimation $\widehat{A}_o = 0$ keeps the minimum distance larger than 1 in most cases, though the results for our method did exhibit a few outliers. Figure 6 also indicates that our method achieves a result similar to that of the method with the correct estimation $\widehat{A}_o = 0$. This is because our method can estimate A_o accurately even when the obstacle does not

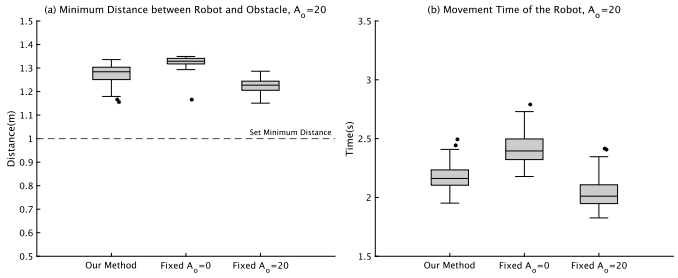

Fig. 5. Comparison with the scenario without parameter estimation. The results for the case where the actual value of $A_o = 20$, indicate that the obstacle will attempt to avoid the robot.

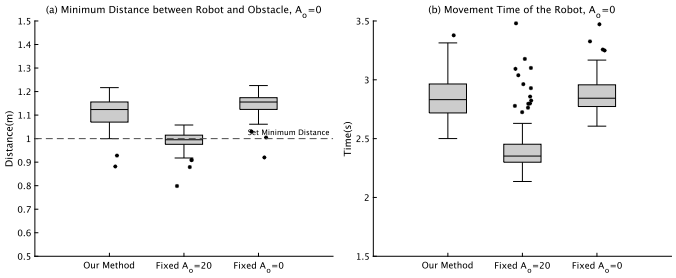

Fig. 6. Comparison with the scenario without parameter estimation. The results for the case where the actual value of $A_o = 0$, indicate that the obstacle will not attempt to avoid the robot.

change its trajectory.

4) Discussion: From our simulations, it is evident that by estimating the obstacle's intention to avoid the robot, our method consistently calculates an appropriate desired velocity even if the observation includes errors. This ensures that the robot not only moves efficiently towards its target, but also maintains a safe distance from moving obstacles, even in the presence of observational errors. The robustness of our approach was demonstrated across 100 iterations with varying noise levels. Although inevitable path intersections test the capabilities of our method, it is worth noting that potential prediction inaccuracies can compromise either the safety or efficiency of our method. Our method demonstrated a strong balance between these two crucial aspects in most situations.

V. Conclusion

In this paper, we utilized the Social Force Model to understand and predict obstacle avoidance behaviors, then applied these insights to calibrate the robot's control input, striking a balance between efficiency and safety. In the future, we will delve deeper into the practical application of our method, encompassing deployment on real robots and testing in real-world environments. Additionally, we will consider the volume of obstacles and extend the obstacles to humans, individuals who might exhibit irrational behavior.

Acknowledgment

These research results were partly obtained from the commissioned research (JPJ012368C00701) by National Institute of Information and Communications Technology (NICT), Japan.

References

[1] Milos Vasic and Aude Billard. Safety issues in human-robot interactions. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 197–204, 2013.

[2] H.J.S. Feder and J.-J.E. Slotine. Real-time path planning using harmonic potentials in dynamic environments. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, volume 1, pages 874–881 vol.1, 1997.

[3] Omur Arslan and Daniel E. Koditschek. Exact robot navigation using power diagrams. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2016.

[4] Marcell Missura and Maren Bennewitz. Predictive collision avoidance for the dynamic window approach. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8620–8626, 2019.

[5] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 797–803, 2010.

[6] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[7] Dirk Helbing and Anders Johansson. *Pedestrian, Crowd and Evacuation Dynamics*, volume 16, pages 697–716. Springer, 04 2010.

[8] Gonzalo Ferrer, Anaís Garrell, and Alberto Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1688–1694, 2013.

[9] Masahiro Shiomi, Francesco Zanlungo, Kotaro Hayashi, and Takayuki Kanda. Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model. *International Journal of Social Robotics*, 6(3):443–455, 2014.

[10] Mitsuhiro Kamezaki, Yusuke Tsuburaya, Taichi Kanada, Michiaki Hirayama, and Shigeki Sugano. Reactive, proactive, and inducible proximal crowd robot navigation method based on inducible social force model. *IEEE Robotics and Automation Letters*, 7(2):3922–3929, 2022.

[11] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350, 2017.

[12] Tetsushi Ikeda, Yoshihiro Chigodo, Daniel Rea, Francesco Zanlungo, Masahiro Shiomi, and Takayuki Kanda. Modeling and prediction of pedestrian behavior based on the sub-goal concept. *Robotics*, 10:137–144, 2013.

[13] Graeme Best and Robert Fitch. Bayesian intention inference for trajectory prediction with an unknown goal destination. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5817–5823. IEEE, 2015.