2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Optimization-based Path Planning for Person Following using Following Field

Heechan Shin[1] and Sung-Eui Yoon[1]

Abstract— **Person following is an essential task for a robot to serve a person. In an indoor environment, however, the following task can be failed due to the occlusion of the target by structures, e.g., walls or pillars. To address this problem, we propose a method that helps the robot follow the target well and rapidly re-detect the target after missing. The proposed method is an optimization-based path planning which uses a *Following Field* that we propose in this paper. The following field consists of two sub-fields: the repulsion field getting the robot out of the occluded area, and the target attraction field pushing the robot toward the target. We introduce how to construct the fields and how to integrate the field into a path optimization process. We show that our method works properly for following the target well in a maze consisting of various in-door features.**

I. INTRODUCTION

The number of collaboration with robots grows in these days [1]. Robots serve people in airports, shopping centers, or even restaurants. While a person collaborates with a robot, he/she expects the robot to perform automatically. Thus, understanding a user's intention well becomes one of the foremost research topics in the robotics field.

In this context, path planning for a robot that follows a user or a target person well is one of the essential tasks for daily cooperation with robots. For example, when a robot supports a person who does his/her job going around in an office or a factory, it is impractical that the user controls every movement of the robot to follow the user. Therefore, we want the robot to follow us automatically.

Main contributions. To achieve the automatic following task, we propose a method that makes the robot follow the target well. We have three contributions to this work. First, we simultaneously consider both how long the robot maintains the following state without missing the target and how fast the robot re-detects the target after missing the target. Second, we merge the person following task into a path planning framework. Finally, we propose a novel concept, *Following Field*, that helps the robot follow the target well. To achieve these, we introduce how to efficiently construct a visibility map and build the following field (Sec. IV).

To evaluate the benefit of our method, we implement the path planner mentioned above and perform the ablation experiment. As a result, the proposed method performs better than the others (Sec. VI).

[1]Heechan Shin (`shin_heechan@kaist.ac.kr`), and Sung-eui Yoon (corresponding author, `sungeui@gmail.com`) are at School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

II. RELATED WORK

Person following can be categorized by various attributes [2]. One of the significant attributes is a medium of operation. There are three operation mediums, including ground scenarios [3], [4], underwater scenarios [5], and aerial scenarios [6], [7]. Among these mediums, this paper focused on the ground scenario, especially with a mobile robot in an indoor scene. We thus mainly review the person following methods with an indoor mobile robot.

One of the main challenges of person following at indoor scenes is missing a target person due to various indoor structures, e.g., doors and pillars. For instance, when a person exits a room or turns around a corner, it is prone for a robot to miss a target. To tackle this problem, there have been two types of prior approaches that we will discuss in later.

A. Reduce Recovery Time

The first approach of handling missing a target is to predict a future target position, for reducing recovery time. Once missing, a robot does not know where the target person is, thus it is desirable for the robot to predict where the target person would be and to re-follow the target candidate location.

These approaches are known as *recovery planners* [2]. Lee et al. [3] tackled this problem by applying Variational Bayesian Linear Regression for trajectory prediction of a missed target. By doing so, they can get a reasonable recovery time. Moreover, they were the only team that passed a person following scenario at RoboCup@Home 2017 [8]. Hoang et al. [9] used the Kalman filter to predict a target person's position. In their work, a robot quickly navigates to the last position where the target is missed, and then uses the Kalman filter if the target is not inside the field of view of the robot. Similarly, Chen et al. [10] replicated a target's trajectory. This approach keeps a recent history of the target positions, and if the target cannot be seen, the robot replicates the trajectory of the missed target. By doing so, the robot has a chance to re-detect the target.

A recovery planner or predicting position of missed target may be helpful to recover the target rapidly. These approaches, however, do nothing for avoiding to miss before missing actually occurs. Instead, these prior methods take actions once the missing occurs. We address this problem in this paper.

B. Maintain Following Time

Maintaining the following is also indispensable. In other words, it is important to delay missing a target as far as possible. To this end, many following strategies have been proposed. Honig et al. [11] reported the following strategies could be categorized as three types. The first type is *direction following*, the most common strategy when a robot follows a target [3], [12]–[14]. Under this strategy, the robot moves toward the target position, simply adjusting velocity. *Path following* strategy is the second type [15], making the robot follow the exact path where the target has walked on. The last strategy is *relative following*, which determines a proper location where the robot should be located at for following the target according to the current target's position. This strategy has various branches since there could be many alternatives on how to determine the appropriate location. Some consider geometric and dynamic properties, e.g., a distance to the target or speed of the target [16], [17]. Others consider social factors [18] etc.

Our work is similar to the third strategy, *relative following*, because we concerned about how to follow a target well. The main difference, however, is that we combine relative following strategy into path planning. In other words, we consider not only where the robot should be to follow the target well, but also how to get there, i.e., planning a path. Furthermore, we have found that considering both low recovery time and long following time was not studied yet. By bridging this gap in the field, we propose a novel path planning strategy, which extends the following time and shortens the recovery time, by using the *Following Field.*

III. Background

In this section, we will briefly introduce the background of our work. First of all, we clarify the general structure of the person following framework and show our main focus in the general structure. Secondly, we explain optimization-based path planning and why we use it as a path planner for our approach. Lastly, the ray casting method as a visibility algorithm is illustrated, which is used at our main method (Sec. IV).

A. Person Following

The main purpose of person following is to make a robot follow a target person keeping a proper distance. To complete the mission, the person following is usually done by two steps, perception and action [3].

In the perception step, the robot identifies and tracks the target. To do so, most of the previous studies applied vision techniques. Though few works [12] apply non-neural network based techniques, most of the recent works [3], [10] use neural networks, especially deep neural network based approaches, e.g., YOLO family [19].

After perception, the robot needs to control itself to follow the tracked target. In this control step, there are two concerns: where to go and how to get there. As we described at Sec. II-B, many studies tackled these problems by proposing several strategies. 'Where to go' pays more attention to suggest a goal position. On the other hand, 'how to get there' has more interest in controlling or path planning.

In our work, we focus on the action step, in particular, combining 'where to go' and 'how to get there'. Specifically, we plan a path considering proper positions of the robot to maximize an opportunity to accomplish the person following.

B. Visibility Check

To make a robot do not miss a target person while following, we use visibility information. In this paper, we use the term 'visibility' as a point of view of the robot. For example, if the robot can see the target person directly, we say the target is visible. On the other hand, if the robot cannot see the target, the target is invisible. We use the visibility information to enhance the robot to keep following the target.

Among various visibility techniques [20], we specifically choose a ray-casting algorithm for computing the visibility, thanks to its simplicity and efficiency. Using the ray-casting algorithm, we know the visibility of the target person by casting an imaginary ray from the position of the robot to the position of the target person. In Sec. IV, we will use the visibility information to build a following field that helps the robot follow the target well.

C. Optimization based Path Planning

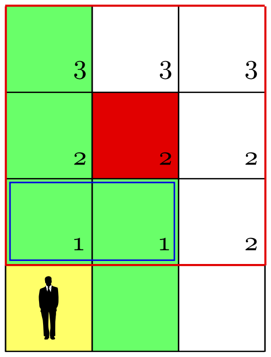
The proposed path planner in this paper is a kind of optimization-based path planners. An optimization-based planner transforms a path planning problem into an optimization problem to plan a path by optimizing objective functions. It divides the path into nodes, which contain their own configurations, and convert them as optimization variables for processing the optimization problem.

In the optimization process, gradient-based approaches are the most common way to solve the problem. It iteratively optimizes the objective function according to the gradient of the problem. Thus, to find a path using an optimization-based method, we need to know gradients of objective functions. For example, obstacle avoidance of optimization-based path planning is performed by pushing the path outward of the obstacle boundary iteratively [21]. To do so, we should know the gradient of an obstacle objective function, which is a set of vectors that directs toward to the obstacle boundaries. With this mechanism, we will introduce a new objective function and its gradient for better following (Sec. V-B).

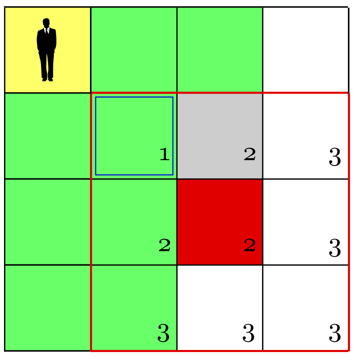
IV. Following Field

Maintaining visibility of the target is necessary to follow the target well. In other words, if we can keep the robot within a region where the robot can see the target, it is highly likely to follow the target well. Inspired by this intuition, we propose a *Following Field.* At a high level, the following field represents regions where the robot can follow the target better, and it is described by visibility information and distance from the target.

The following field consists of two components of repulsion and target attraction fields. The repulsion field is to repulse a robot from the invisible region, and the target attraction field is to direct the robot to the target. We first address how to build the visibility map that is the basis for



(a)



(b)

Fig. 1. Two sample cases of how to choose $C_{analysis}$, which are marked with blue line. Red cell is a C_{check} that we want to check visibility. The target is at the yellow cell, and the green cells are currently constructed visible region. C_{neigh} is marked by red line. Chebyshev distances from C_{neigh} and C_{check} to C_{target} are denoted on the bottom right corner of each cell. Gray cell indicates an invisible cell.

other fields, and then explain two main components of the following field.

A. Generating Visibility Map

We construct visibility map, which is simply a grid map that contains visibility information, i.e., visible or invisible. For example, if the robot can see the target directly, we mark the position where the robot stands to be visible. By determining this visibility information on every grid cell of the map, we can generate a visibility map. The visibility map is then decomposed into the visible region where the robot can see the target directly, and the invisible region where the robot cannot see the target.

When checking the visibility at every cell on the map, a ray-casting method is used. We shoot a ray from every cell on the map toward the target and obtain a visibility information of each cell. This indicates that if a robot is placed on a visible cell, the robot can see the target directly from that cell. Though it is easy to understand how to construct a visibility map, it can consume a lot of time to check the visibility information at every cell on the map. Therefore, we introduce a fast visibility map construction method based on a culling technique inspired by the concept of ‘culling region’ [22].

Constructing a visibility map is achieved by extending a visible region from a cell where the target stands, C_{target}, and extends the visible region by determining the visibility information of its neighbors. Let C_{check} be a cell that we want to check the visibility and C_{neigh} be the neighbor cells of C_{check}. Then, C_{neigh} is always visible by C_{check}, because they are adjacent. Thus, if all cells among C_{neigh} and laid between C_{check} and C_{target} are visible, then C_{check} is also visible. With this intuition, instead of checking the visibility by ray casting, we can determine the visibility of a cell by analysing its neighbors. Then, we can determine the visibility of C_{check} by analysing some cells among C_{neigh}, whose chebyshev distance to C_{target} is less than the distance between C_{check} and C_{target} (See Fig. 1), and we named these cells as $C_{analysis}$.

Algorithm 1 Calculating distance field from invisible region

```
Require: Boundary cells of invisible region, $C_{boundary}$,
         resolution of the field, $\rho$
  $Q$.push($\forall c \in C_{boundary}$)                 ▷ $Q$ is queue.
  $S_{close} \leftarrow \emptyset$                       ▷ $S_{close}$ is set.
  $S_{tmp} \leftarrow \emptyset$                         ▷ $S_{tmp}$ is set.
  $\phi(\forall c \in C_{boundary}) = 0$
                 ▷ $\phi(\cdot)$ is distance function from the boundary
  while $Q$ is not empty do
      $S_{close}$.insert($\forall q \in Q$)
      while $Q$ is not empty do
          $q \leftarrow Q$.pop()
          $\Psi \leftarrow FindNeighbors(q)$
          for $\psi$ in $\Psi$ do
              if $\psi \notin S_{close} \wedge \psi \notin Obstacles$ then
                  $S_{tmp}$.insert($\psi$)
      for $\psi$ in $S_{tmp}$ do
          if $\psi \in InvisibleRegion$ then
              $\phi(\psi) = \phi(C) + \rho$
          else
              $\phi(\psi) = \phi(C) - \rho$
      $Q$.push($\forall s \in S_{tmp}$)
      $S_{tmp}$.clear()
  return $\phi(\cdot)$
```

$$C_{analysis} = \{C \mid CD(C, C_{target}) < CD(C_{check}, C_{target}), C \in C_{neibgh}\}$$

Where $CD(\cdot,\cdot)$ is chebyshev distance for the given cells.

After determine $C_{analysis}$, we can easily check the visibility of C_{check} as follows.

$$\text{Visibility of } C_{check} = \begin{cases} visible, & \text{if } \forall C \in C_{analysis} \text{ is } visible \\ invisible, & \text{if } \forall C \in C_{analysis} \text{ is } invisible \\ Raycasting(), & \text{otherwise} \end{cases}$$

Only except $\forall C \in C_{analysis}$ is either *visible* or *invisible*, we cast a ray from C_{check} to C_{target} to determine visibility.

As a result, we construct the visibility map consisted of visible region and invisible region. This map is used for recognizing where the invisible region is to escape from there.

B. Repulsion Field

After building the visibility map, we can figure out where the visible region is and where it is not. To use this information while planning a path to follow a target, we need to know how close each cell is from the invisible region. Intuitively, the robot should be away from those invisible regions and get closer to the visible regions. This intuition is captured in the repulsion field, $\mathfrak{R}$, in a form of a signed distance field (SDF).

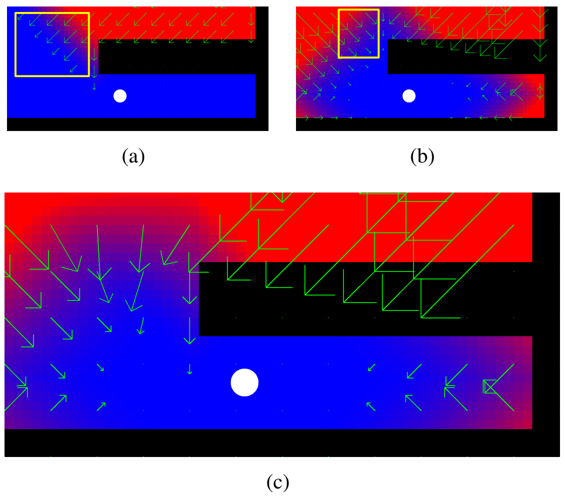


Fig. 3. (a) is the repulsion field, (b) is the target attraction field, and (c) is the following field. Blue region represent low cost and red region represent high cost. The white circle is the position of the target.

Specifically, we construct the SDF based on the boundaries of the invisible region. Though the concept is similar, the approach of constructing the SDF of the invisible region is different from the general approach of constructing the SDF of obstacles. That is because, when we construct the SDF of the invisible region, we need to consider not only the boundaries of the invisible region, but also to handle the obstacles. For example, in the Fig. 3-(a), distance from the boundary of invisible region, which is between red and blue region, is calculated detouring the obstacle, which is colored as black. This means we need to calculate not 'displacement' to the boundary of invisible region, yet 'length of the shortest path' to the boundary of invisible region to detour the obstacles. Thus, to calculate it, we propose a method of identifying the boundaries of invisible region and calculating the distance.

We identify the boundaries of invisible region by finding a boundary of visible region. Note that the visibility map can have only one visible region, which is a topologically connected region; the visible region is star-shaped according to the target position. On the other hand, the visibility map can have multiple invisible regions. For finding the boundary of the visible region, we adopt wavefront algorithm [23]. The wavefront algorithm is an optimal navigation algorithm that propagates 'wavefront' to find the shortest path from the start to every cell on the map. By propagating the wavefront from C_{target}, we can find the boundary of the visible region, while avoiding to investigate every cell on the map.

After identifying the boundaries of invisible region, we calculate the shortest path distance from the boundaries. Similar to the wavefront algorithm, the calculation of distance is performed in a breath-first manner. We store all the boundary cells to the queue, find neighbors of the stored cells, and calculate the distances. By repeating this, we can get the distance field of the invisible region. Its detailed process is elaborated at Algorithm 1. Since what we finally want to build is the 'signed' distance field for the repulsion field, we assign a positive distance to the invisible region and a negative distance to the visible region.

C. Target Attraction Field

Although the repulsion field helps the path planner repulse the invisible region, it is not enough to make a robot follow a target well. This is mainly because the repulsion field contains only information about how to escape the invisible region. What we need to additionally know for following a target person well is how to get closer to the target.

To realize such a goal, we use a target attraction field, $\mathfrak{T}$, in addition to the repulsion field. The target attraction field is another field that represents how close to the target. We can simply construct the target attraction field, again using the wavefront algorithm. Similar to using the wavefront algorithm for calculating the distance from the boundaries of the invisible region in Sec. IV-B, we use the same algorithm to measure a distance from the target to any cell on the map. It can be done by iteratively executing the wavefront algorithm until all the distances of the every cell on the map are measured. This results in a distance field around the target, which is the target attraction field. By calculating its gradient, we can forces the robot to get closer to the target; this will be discussed in Sec.V-B.

D. Following Field

Using the fields we proposed above, we finally construct a following field, $\mathfrak{F}$, which is used in the path optimization process to make the robot follow the target well. The following field is a weighted sum of the repulsion field and the target attraction field. Note that the repulsion field is used for guiding the robot only towards repulsive directions from the invisible region; see the yellow box of Fig. 3-(a). Also, the target attraction field does not consider visibility information, which sometimes makes the robot get closer to the invisible region, see yellow box of Fig. 3-(b). The following field addresses these problems by summing them up. Consequently, it helps the robot get closer to the target, while detouring the invisible region; see Fig. 3-(c).

In the next section, we consider the following field into the path optimization process to plan a path that follows the target well.

V. Path Optimization with Following Field

In this section, we describe how the basic optimization-based path planning works and how we integrate the following field into the optimization process. Notations used in this section are described in Table I.

A. Basic Components of Optimization based Path Planning

While optimizing the path, we interpolate the nodes to satisfy the connectivity of the path. In addition to the connectivity, obstacle avoidance, and kinematic feasibility of the path are also crucial. We will explain how we deal with those components.

Obstacle Objective. We use Bidirectional Obstacle Estimation(BOE) of Shin et al.'s work [24] for avoiding obstacles.

TABLE I
NOTATIONS

Notation	Description
i	node index on the path
N	number of nodes on the path
x_i, y_i	coordinate value of the robot at i^{th} node
v_i	speed of the robot at i^{th} node
$v_{x,i}$, $v_{y,i}$	x-axis and y-axis value of normalized velocity at i^{th} node
t_f	travel time of the path
dt	time differences between adjacent nodes($=\dfrac{t_f}{N-1}$)

It detects the obstacles by casting perpendicular rays from each node of the path. Thus, this method is not dependent on the size of an obstacle map, but it is dependent on the number of nodes. Thanks to the small planning area of the person following task, about one or two meters, the number of nodes, N, does not need to be large.

$$f_o(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N} BOE(x_i, y_i)$$

Smooth Path Objective. For comfort maneuvering, the smooth path is preferred in many cases [25]–[27]. In general, optimization-based planners smooth the path by minimizing the sum of derivatives of configurations, i.e., $\sum_{d=1}^{\infty} q^{(d)}$, where q is a configuration in the path [25]. It is a burden, however, to consider all derivatives. Instead, we minimize differences in speed and orientation vector of the robot between adjacent nodes to smooth the path:

$$f_{s1}(\mathbf{v}) = \sum_{i=1}^{N-1} (v_{i+1} - v_i)^2,$$

$$f_{s2}(\mathbf{v}_x, \mathbf{v}_y) = \sum_{i=1}^{N-1} (v_{x,i+1} - v_{x,i})^2 + (v_{y,i+1} - v_{y,i})^2.$$

Where v_i is a speed of node i. $v_{x,i}$ and $v_{y,i}$ are components of direction vector of velocity at node i. $\mathbf{v}$, $\mathbf{v}_x$, and $\mathbf{v}_y$ are vectors which contain v_i, $v_{x,i}$, and $v_{y,i}$ $\forall i = 1 \ldots N-1$.

Kinematic Constraints. To make a robot go through the planned path, the path should respect the kinetic constraints of the robot. Unlike other 'objectives' of the optimization problem, kinematic 'constraints' have to be satisfied. Therefore, we assign equality constraints between each node and its adjacent node in the path:

$$x_{i+1} - (x_i + v_i\, v_{x,i}\, dt) = 0, \quad i = 1, \ldots, N-1$$
$$y_{i+1} - (y_i + v_i\, v_{y,i}\, dt) = 0, \quad i = 1, \ldots, N-1$$

B. Following Field Objective

We now propose the following field objective. The following field that we construct in Sec. IV represents positions where a robot can follow the target well. The following field is a the summation of two sub-fields: the repulsion field and the target attraction field. The repulsion field shows how far each cell is from the invisible region, and the target attraction field shows how close from the target.

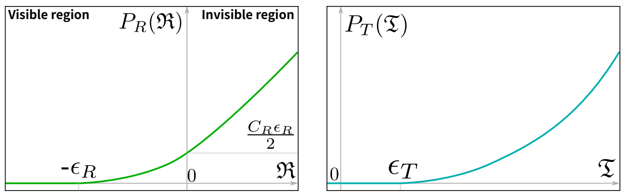


Fig. 4. Penalize functions for $\mathfrak{R}$ and $\mathfrak{T}$. Each function penalizes the field where the value of the field is useless, e.g., too far from the obstacle boundaries or too close to the target, according to ε_R and ε_T respectively.

Both fields are related to distance, which increases or decreases linearly according to their own references. For efficient converging, we define penalize functions $P_{R \text{ or } T}(\cdot)$ for each field inspired by [25].

For the repulsion field, $\mathfrak{R}$, we do not need to consider the repulsion value which is far away from the invisible boundaries and inside of the visible region, specifically higher distance than ε_R. We penalize the values for the cells that are far from the invisible boundaries; see Fig. 4-(a) The penalize function for the repulsion field, $P_R(\cdot)$, is defined as follows:

$$P_R(\mathfrak{R}) = \begin{cases} 0, & \text{if } \mathfrak{R} < -\varepsilon_R \\ C_R \frac{(\mathfrak{R}+\varepsilon_R)^2}{2\varepsilon_R}, & \text{if } -\varepsilon_R \le \mathfrak{R} < 0 \\ C_R(\mathfrak{R} + \frac{\varepsilon_R}{2}), & \text{otherwise.} \end{cases}$$

Where C_R is a tangential coefficient.

Likewise, for the target attraction field, cells that are too close to the target are ignorable. The penalize function for the target attraction field, $P_T(\cdot)$, is defined as follows:

$$P_T(\mathfrak{T}) = \begin{cases} 0, & \text{if } \mathfrak{T} < \varepsilon_T \\ C_T(\mathfrak{T} - \varepsilon_T)^2, & \text{otherwise.} \end{cases}$$

Where C_T is a tangential coefficient.

With the penalize functions, the following field, $\mathfrak{F}$, becomes not just a simple summation of $\mathfrak{R}$ and $\mathfrak{T}$, but the summation of the penalized $\mathfrak{R}$ and $\mathfrak{T}$. Then, it becomes following field objective for the optimization problem.

$$f_f(\mathbf{x},\mathbf{y}) = \mathfrak{F} = P_R(\mathfrak{R}) + P_T(\mathfrak{T})$$

Both $\mathfrak{R}$ and $\mathfrak{T}$ are only affected by configurations of locations, $\mathbf{x}$ and $\mathbf{y}$. The gradients along the x-axis and y-axis of each penalized field can be obtained by calculating the gradient of each penalized field.

$$\nabla f_f(\mathbf{x},\mathbf{y}) = \begin{bmatrix} \nabla_x f_f \\ \nabla_y f_f \end{bmatrix} = \begin{bmatrix} \nabla_x P_R(\mathfrak{R}) + \nabla_x P_T(\mathfrak{T}) \\ \nabla_y P_R(\mathfrak{R}) + \nabla_y P_T(\mathfrak{T}) \end{bmatrix}.$$

C. Optimization Problem for Person Following

So far, we discussed various objectives and constraints. We put them together as the overall formulation of the

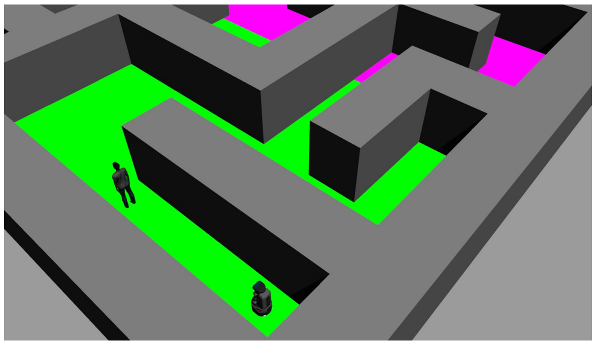


Fig. 5. The maze scene is constructed on Gazebo robot simulator.

optimization process to make a robot follow the target well:

$$\begin{aligned}
\underset{t_f,\mathbf{x},\mathbf{y},\mathbf{v},\mathbf{v}_x,\mathbf{v}_y}{\text{minimize}} \quad & \lambda_t t_f + \lambda_{s1} f_{s1} + \lambda_{s2} f_{s2} + \lambda_o f_o + \lambda_f f_f \\
\text{subject to} \quad & x_{j+1} - (x_j + v_j v_{x,j} dt) = 0, \quad j = 1, \dots, N-1, \\
& y_{j+1} - (y_j + v_j v_{y,j} dt) = 0, \quad j = 1, \dots, N-1, \\
& v_{x,j}^2 + v_{y,j}^2 = 1, \quad j = 1, \dots, N, \\
& g(\mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{v}_x, \mathbf{v}_y) \le 0,
\end{aligned}$$

where λs are weight parameters for each objective, and we set λ_t=1e^{-4}, λ_{s1}=1, λ_{s2}=1, λ_o=100, and λ_f=1.5. $g(\mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{v}_x, \mathbf{v}_y)$ is additional constraints including start and goal restriction, or bounds of the variables. Moreover, in addition to the objectives that we explain in Sec. V-A and Sec. V-B, we also consider t_f, which is the total travel time of the path, as an objective for identifying a shorter path.

Finally, the optimization-based path planner pushes the path to make a robot follow a target well by optimizing the aforementioned optimization objective.

VI. Experiments

The experiments are executed on Intel i7-8700K CPU with 64GB RAM. To verify the effectiveness of the proposed method, we execute the experiment on a robot simulator system, Gazebo [28]. It is because, for a fair comparison, the trajectory of a target person has to be the same while the experiments are performed. In this experiment, we use IPOPT [29], a non-linear optimization solver; we can use any non-linear optimization solver that can solve the optimization problem with constraints, to solve the optimization problem we discussed in Sec. V.

Experimental setting: To plan a path using the optimization approach, we use several parameters. Most of the parameters are fixed for the fair comparison. Only a few parameters are modified according to the test for the ablation study. We set N=20, ρ=0.05(m), ε_R=1.5(m), C_R=0.1, ε_T=3(m), and C_T=0.01.

We evaluate the performance of our method on a maze-like scene, Fig. 5. We use the maze because it consists of various aspects that are parts of indoor environments, e.g., office or school. We construct the maze to have different sections with various characteristics. The first section is colored as green. This section mimics a wall that the target turns around and a narrow aisle. The second section shown in the magenta has a long hallway with two corners at the head and the tail of the way. The third, yellow section has a S-shaped corner. It is difficult for the robot to follow the target at the S-shaped way because the target can hide multiple times, passing through the way. The last section with the cyan has a standard corner where the target turns 90 degrees. We set the maximum speed of the robot as 1m/s.

TABLE II
The results of the experiment.

Type		Θ	n_{miss}	ω	Σt_{recov}	t_{tot}	$\mathcal{L}_{invis}$
ξ	*Ours*	**0.29**	5	0.06	13.8	234.85	9.54
	Ours w/o $\mathfrak{T}$	0.93	9	0.10	24.45	235.65	14.33
	Ours w/o $\mathfrak{R}$	2.67	12	0.22	52.4	235.5	19.61
	w/o *Ours*	3.75	~~12~~	0.31	58.9	188.75	29.93
η_G	*Ours*	**0**	0	0	0	38.8	0
	Ours w/o $\mathfrak{T}$	0.03	1	0.03	1.0	39.15	0.87
	Ours w/o $\mathfrak{R}$	1.32	3	0.44	17.45	39.6	1.8
	w/o *Ours*	1.76	4	0.44	18.1	41.2	8.95
η_M	*Ours*	**0.14**	2	0.07	5.75	80.8	3.91
	Ours w/o $\mathfrak{T}$	0.42	3	0.14	11.25	80.9	7.1
	Ours w/o $\mathfrak{R}$	0.64	4	0.16	12.95	80.5	7.08
	w/o *Ours*	1.43	5	0.29	22.9	80.25	11.54
η_Y	*Ours*	**0.16**	2	0.08	5.0	62.45	3.68
	Ours w/o $\mathfrak{T}$	0.47	4	0.12	7.35	62.15	3.89
	Ours w/o $\mathfrak{R}$	1.13	4	0.28	17.85	63.45	8.13
	w/o *Ours*	1.29	~~3~~	0.43	17.9	41.5	9.44
η_C	*Ours*	**0.11**	1	0.11	3.05	27.05	1.96
	Ours w/o $\mathfrak{T}$	0.18	1	0.18	4.85	26.85	2.47
	Ours w/o $\mathfrak{R}$	0.16	1	0.16	4.15	26.65	2.59
	w/o *Ours*	·	·	·	·	·	·

ξ represents the entire path. η_G, η_M, η_Y, and η_C are the path of inside of green, magenta, yellow, and cyan sections, respectively. $\mathcal{L}_{invis}$ is a path length of the invisible path. Units of Σt_{recov}, t_{tot}, and $\mathcal{L}_{invis}$ are seconds, seconds, and meters respectively. Strikeout means failure case.

Since the path of the target has to be reproducible for a fair comparison, we use a reproducible but simple path planner, the A^* path planner. Any path planner, however, can be used if the output path is reproducible. Moreover, during the entire experiment, the robot plans the path only when the target is visible. Otherwise, it follows the previously planned path.

A. Difficulty of Following Task

To evaluate whether the robot follows the target well or not, we measure two quantities: recovery time and following time. The recovery time is a duration of recovering the missed target. On the other hand, the following time is how

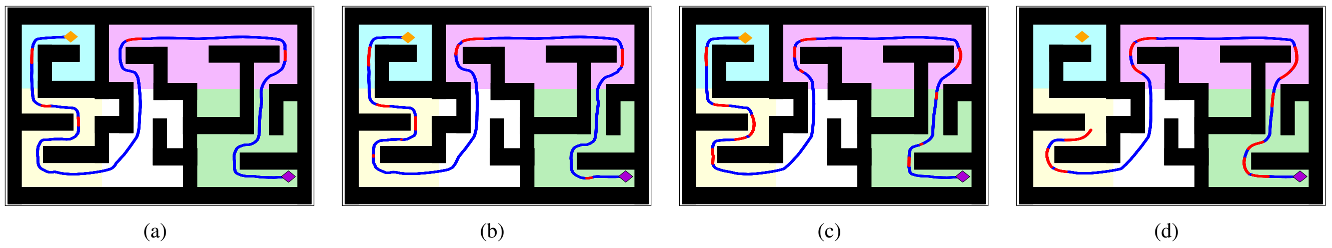


Fig. 6. The results of the planned path for each objective. (a) is the result of the objective using following field, *Ours*. (b), (c), and (d) are the results of the objective using only the repulsion field, *Ours* w/o $\mathfrak{T}$, the objective using only the target attraction field *Ours* w/o $\mathfrak{R}$, and the objective using only the basic components, w/o *Ours*. The blue lines represent visible path, and the red lines represent invisible path. The purple diamond is the starting point and the orange diamond is the goal point of the target.

long the robot maintains to follow the target without missing the target.

The red lines, invisible path segments, on the path, depicted in Fig. 6, show path segments where the robot cannot see the target, and the blue lines, visible path segments, are where it can see the target. In this context, the recovery time, t_{recov}, can be measured by calculating the duration of going through the invisible path. The following time can be calculated by subtracting t_{recov} from the total travel time, t_{tot}, of the path.

Since the recovery time and the following time are dependent on each other, the ratio, ω, of Σt_{recov} to the total time t_{tot} represents the difficulty, Θ, of the following task. For example, if a robot follows the target by 10 seconds, the recovery time of 2 seconds makes the robot harder to follow the target than the recovery time of 1 second. It is, however, not enough for determining Θ, because even with the same Σt_{recov}, frequently missing the target is undesirable than rarely missing the target. We reflect how often the robot misses the target, n_{miss}, in measuring Θ. Therefore, we finally measure the difficulty of the following task, Θ, as follows.

$$\Theta = n_{miss} \times \omega$$

B. Performance Evaluation

Comparing our work with previous person following systems, e.g., [4], [8], is almost impossible since implementing the same system is unfeasible. Instead, to prove the utility of our work, we conduct ablation study with our method. We test four distinct objectives to verify the performance of path optimization using the following field. The first objective, *Ours*, is what we proposed in this paper that considers the following field to optimize. The second objective, *Ours* w/o $\mathfrak{T}$, considers only the repulsion field, i.e., without the target attraction field. Likewise, the third objective, *Ours* w/o $\mathfrak{R}$, considers only the target attraction field. The other objective, w/o *Ours*, considers only the basic components of optimization-based path planning explained in Sec. V-A. The test is conducted in the maze scene, and its result is reported in Table II.

We report various results across all the trajectory, a type of ξ, or only in a particular section; η_G is only for the path of the robot inside of the green section. Likewise, η_M, η_Y, and η_C are only for the path in the magenta, yellow, and cyan sections, respectively.

The result shows that considering the following field, which includes the repulsion field and the target attraction field, performs better than the others.

First of all, it rarely misses the target in comparison to the others. Even the robot does not miss the target in the green section, η_G, while others miss the target several times. Moreover, it has the smallest value of ω, which is ratio of Σt_{recov} to t_{tot}. This means the robot, that plans a path with *Ours*, spends less time than the others on the invisible path compared to the entire travel time. Therefore, planning a path with *Ours* achieves the best performance, the lowest Θ, compare to others from 3.1769 times to 9.08 times, except for the failure cases.

Additionally, we also can analyze the performance in other ways. For the average recovery time, sorely dividing Σt_{recov} by the number of missing, n_{miss}, does not reflect the performance properly. Because, a short occlusion drastically reduces the average of the recovery time. To address it, we calculate the average speed of escaping the invisible path since the faster the robot escapes the invisible path, the opportunity of seeing the target is increased. In ξ, the robot, which plans the path with *Ours*, escapes the invisible path with 0.69 m/s on average, which is faster than the others. The average escaping speeds with *Ours* w/o $\mathfrak{T}$, *Ours* w/o $\mathfrak{R}$ and w/o *Ours* are 0.59m/s, 0.37m/s and 0.51m/s, respectively.

VII. Conclusion

In this paper, we address the main problem of the person following task, which is the robot misses the target person before reaching the goal, by considering Following Field when planning a path. The following field consists of the subfields, the repulsion field, and the target attraction field. We introduce an efficient way of constructing a visibility map for building the repulsion field. Besides, by applying wavefront algorithm, we explain how to construct the repulsion field and the target attraction field. Lastly we merge the following field concept into optimization-based path planning.

We analyze the performance of our method by ablation experiment in the maze-like scene. The result shows that the performance of the following task with the proposed method is better than the others. Additionally, it also shows that the person following with the proposed method has the fastest average speed of escaping the invisible path.

In the future work, we can predict the future position of the target person and use the future following field information for better following.

Acknowledgment

This research was supported by SW Starlab program (IITP-2015-0-00199) and NRF/MSIT (No.2019R1A2C3002833).

References

[1] Kerstin Dautenhahn, "Methodology & themes of human-robot interaction: A growing research field", *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, pp. 15, 2007.

[2] Md Jahidul Islam, Jungseok Hong, and Junaed Sattar, "Person following by autonomous robots: A categorical overview", *The International Journal of Robotics Research*, 03 2018.

[3] Beom-Jin Lee, Jinyoung Choi, Christina Baek, and Byoung-Tak Zhang, "Robust human following by deep bayesian trajectory prediction for home service robots", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7189–7195.

[4] Goran Huskić, Sebastian Buck, Luis Azareel Ibargüen González, and Andreas Zell, "Outdoor person following at higher speeds using a skid-steered mobile robot", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3433–3438.

[5] Md Jahidul Islam, Marc Ho, and Junaed Sattar, "Understanding human motion and gestures for underwater human–robot collaboration", *Journal of Field Robotics*, vol. 36, no. 5, pp. 851–873, 2019.

[6] M. Le and M. Le, "Human detection and tracking for autonomous human-following quadcopter", in *2019 International Conference on System Science and Engineering (ICSSE)*, July 2019, pp. 6–11.

[7] Ningshi Yao, Emily Anaya, Qiuyang Tao, Sungjin Cho, Hongrui Zheng, and Fumin Zhang, "Monocular vision-based human following on miniature robotic blimp", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3244–3249.

[8] B. Loy van, H. Dirk, M. Mauricio, R. Caleb, and W. Sven., *Robocuphome 2017 Rules and regulations*, [Online]. Available: http://www.robocupathome.org/rules/2017_rulebook.pdf, 2017.

[9] Minh-Do Hoang, Sang-Seok Yun, and Jong-Suk Choi, "The reliable recovery mechanism for person-following robot in case of missing target", in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2017, pp. 800–803.

[10] Bao Xin Chen, Raghavender Sahdev, and John K Tsotsos, "Integrating stereo vision with a cnn tracker for a person-following robot", in *International Conference on Computer Vision Systems*. Springer, 2017, pp. 300–313.

[11] Shanee S Honig, Tal Oron-Gilad, Hanan Zaichyk, Vardit Sarne-Fleischmann, Samuel Olatunji, and Yael Edan, "Toward socially aware person-following robots", *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 4, pp. 936–954, 2018.

[12] Andrés Echeverri Guevara, Anthony Hoak, Juan Tapiero Bernal, and Henry Medeiros, "Vision-based self-contained target following robot using bayesian data fusion", in *ISVC*, 2016.

[13] Jongho Han and Taeseok Jin, "Sound source based mobile robot control for human following in a networked intelligent space", *International Journal of Control and Automation*, vol. 8, no. 7, pp. 67–74, 2015.

[14] Mengmeng Wang, Yong Liu, Daobilige Su, Yufan Liao, Lei Shi, Jinhong Xu, and Jaime Valls Miro, "Accurate and real-time 3-d tracking for the following robots by fusing vision and ultrasonar information", *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 997–1006, 2018.

[15] Dror Katz, "Development of algorithms for a human-following robot equipped with kinect vision and laser sensors in an unknown indoor environment with obstacles and corners", Master's thesis, Ben-Gurion University of the Negev, 2016.

[16] Hyukseong Kwon, Youngrock Yoon, Jae Byung Park, and Avinash C Kak, "Person tracking with a mobile robot using two uncalibrated independently moving cameras", in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2877–2883.

[17] Kazuyuki Morioka, Joo-Ho Lee, and Hideki Hashimoto, "Human-following mobile robot in a distributed intelligent sensor network", *IEEE Transactions on Industrial Electronics*, vol. 51, no. 1, pp. 229–237, 2004.

[18] Gonzalo Ferrer, Anaís Garrell Zulueta, Fernando Herrero Cotarelo, and Alberto Sanfeliu, "Robot social-aware navigation framework to accompany people walking side-by-side", *Autonomous robots*, vol. 41, no. 4, pp. 775–793, 2017.

[19] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement", *arXiv*, 2018.

[20] S. Yoon, *Rendering*, Freely available on the internet, 1 edition, 2018, https://sglab.kaist.ac.kr/ sungeui/render/.

[21] Helen Oleynikova, Alexander Millane, Zachary Taylor, Enric Galceran, Juan Nieto, and Roland Siegwart, "Signed distance fields: A natural representation for both mapping and planning", in *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan, 2016.

[22] Youngsun Kwon, Donghyuk Kim, Inkyu An, and Sung-eui Yoon, "Super rays and culling region for real-time updates on grid-based occupancy maps", *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 482–497, 2019.

[23] Steven M LaValle, *Planning algorithms*, Cambridge university press, 2006.

[24] Heechan Shin, Donghyuk Kim, and Sung-Eui Yoon, "Kinodynamic comfort trajectory planning for car-like robots", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6532–6539.

[25] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[26] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal, "Stomp: Stochastic trajectory optimization for motion planning", in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[27] M. Fu, K. Zhang, Y. Yang, H. Zhu, and M. Wang, "Collision-free and kinematically feasible path planning along a reference path for autonomous vehicle", in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[28] Nathan Koenig and Andrew Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator", in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. IEEE, 2004, vol. 3, pp. 2149–2154.

[29] Andreas Wächter and Lorenz T Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.