Computational Visual Media
https://doi.org/10.1007/s41095-021-0266-0 Vol. 9, No. 1, March 2023, 177–194

Research Article

AR assistance for efficient dynamic target search

Zixiang Zhao[1], Jian Wu[1], and Lili Wang[1] (✉)

Abstract When searching for a dynamic target in an unknown real world scene, search efficiency is greatly reduced if users lack information about the spatial structure of the scene. Most target search studies, especially in robotics, focus on determining either the shortest path when the target's position is known, or a strategy to find the target as quickly as possible when the target's position is unknown. However, the target's position is often known intermittently in the real world, e.g., in the case of using surveillance cameras. Our goal is to help user find a dynamic target efficiently in the real world when the target's position is intermittently known. In order to achieve this purpose, we have designed an AR guidance assistance system to provide optimal current directional guidance to users, based on searching a prediction graph. We assume that a certain number of depth cameras are fixed in a real scene to obtain dynamic target's position. The system automatically analyzes all possible meetings between the user and the target, and generates optimal directional guidance to help the user catch up with the target. A user study was used to evaluate our method, and its results showed that compared to free search and a top-view method, our method significantly improves target search efficiency.

Keywords augmented reality (AR); search; guidance

1 Introduction

Guided navigation in real world indoor environment is an essential problem for many applications. Typically, the shortest path will be calculated to guide the user to a target's position. When the target is moving, a simple shortest path algorithm may not work well, and the problem is even more difficult when the target's position is only intermittently known. The consequent dynamic target search task is tricky.

Much work has considered improving the efficiency of the search. Specially designed scenes integrating some salient features [1, 2] or preset landmarks [3, 4] may be used to improve navigation efficiency. However, complex preprocessing and lack of robustness make it difficult to use these methods in general scenarios. Some applications offer 2D or 3D maps of the entire scene to the users and let them decide where to search for the target [5, 6], while trail methods highlight the visited parts of the environment or draw users' footprints to help them navigate more efficiently [7, 8]. Both map and trail method require users to switch from a first-person perspective to a third-person perspective to locate themselves and the target, which interrupts visual continuity of the 3D scene, leading to higher time costs and task load. The guiding tour method guides users to important positions, using some automatically computed paths [9, 10]. Such work analyzes only the user's current position and scene structure, and is unsuitable for the dynamic target search task.

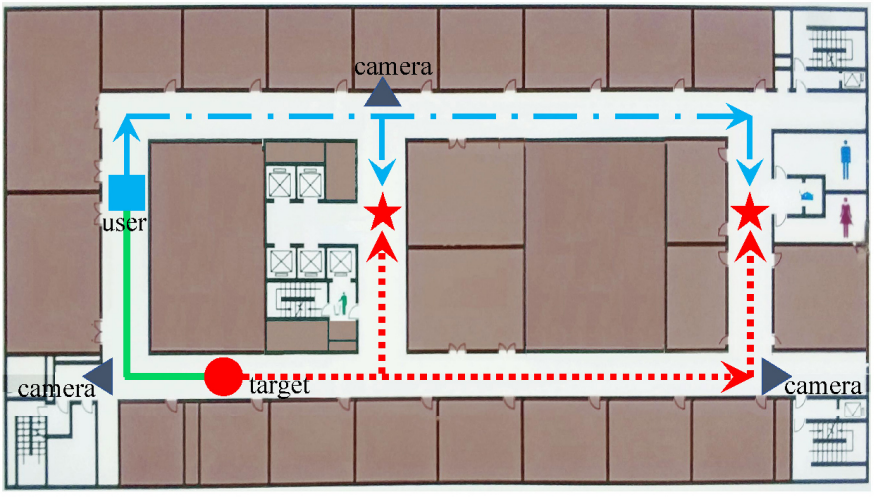
When searching for a dynamic target in a real world scene, using fixed surveillance cameras, the target's position can only be acquired intermittently. The resulting search efficiency is low for two reasons. Firstly, due to complex occlusions, users cannot grasp the global structure of the entire scene and their own position. Secondly, if users do not know the changing position of the moving target, it is difficult to search for such dynamic objects systematically. Even when a scene map and extra fixed surveillance camera views are provided to them, it is still hard for users to find an efficient route to the target, especially when they are not familiar with the scene.

1 State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China. E-mail: Z. Zhao, zhaozixiang@buaa.edu.cn; J. Wu, lanayawj@buaa.edu.cn; L. Wang, wanglily@buaa.edu.cn (✉).

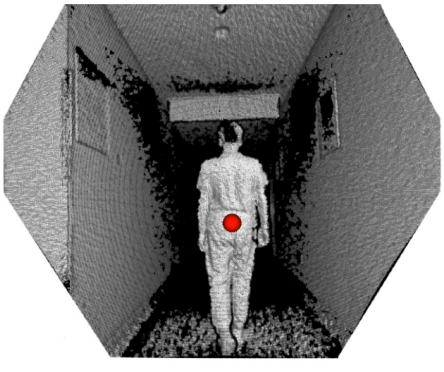
Manuscript received: 2021-07-28; accepted: 2021-12-21

In this paper, we design an augmented reality (AR) guidance assistance system based on a prediction graph search (PGS) algorithm to help users efficiently find dynamic uncontrolled individual targets (see Fig. 1). Unlike a simple shortest path method, we provide an optimized current walking direction to guide users to places where they are more likely to find the target. Recently, depth cameras have become widely used in dynamic target tracking as no additional sensors are required to acquire the target. We assume that a certain number of depth cameras have been positioned, allowing acquisition of the dynamic object's position and motion by analyzing the depth information (see Fig. 1(b)). The problem of covering a bounded environment with the minimal number of cameras is NP-hard; we do not discuss this problem in our paper. Using the acquired target information, our method predicts probabilities of future target's positions using a prediction graph search algorithm. Then, for every allowable walking direction, a user search process is conducted to estimate possible meeting positions and probabilities. Finally, meeting probabilities are accumulated in each direction, and the one with maximum probability is used to determine guidance direction which is displayed in a HoloLens to assist the user searching for the targets (see Fig. 1(c)).

We have tested our AR guidance assistance method in a user study with 16 participants, who tried to find a moving person in three different scenes. The study has three conditions: control condition one (CC_1) is free search with no assistance, control condition two



(a)



(b)

(c)

Fig. 1 (a) Floor plan of a building; the user (blue square) searches for a moving target (red dot) in a circular corridor. The target is captured by multiple depth cameras (gray triangles). The red dotted line shows the target's path as predicted by our method, the red asterisk indicates the predicted meeting position, the blue dash-dotted line shows the user's optimal path calculated by our method, and the green line is the current shortest path from the user to the target. (b) Target depth image captured by the depth camera at the bottom left of (a); the target's position marked with a red dot. (c) Our AR assistant interface as displayed on the Microsoft HoloLens, which efficiently guides the user to the moving target.

(CC_2) uses the conventional approach of displaying a top view map with target's position information, and the experimental condition (EC): our method with PGS assistance. The results show that participants using our method found the target significantly faster than under the other two conditions (CC_1 and CC_2). The experimental condition has significantly lower task load scores than the control conditions, and significantly higher usability scores, in our questionnaire.

In summary, the contributions of our paper are as follows:

- an AR search assistant system using a HoloLens and several Kinect cameras to help users efficiently search for dynamic targets whose movement information is incomplete or intermittent,
- a PGS based dynamic target's position prediction algorithm, used in our search system to help optimize the guidance direction, and
- a test of the efficiency of our method on dynamic target searching tasks using three different scenes, the result being that, compared to free-walking and a top-view method, our method shows a significant improvement in search efficiency.

2 Related work

2.1 AR guidance

Recently, augmented reality (AR) has been applied to many real world interactive applications with excellent results. Some researchers use deep learning methods for virtual content creation and exploration to improve immersion, interactivity, and intelligence [11]. Numerous studies have shown that AR can improve the efficiency of people performing tasks in the real world, such as maintenance, repair, and operation of complex equipment. Using AR systems for maintenance and assembly tasks can improve performance, reduce task time [12] and number of errors [13]. Researchers have used virtual arrows to demonstrate necessary actions in a robot assembly application [14, 15]. Compared to a non-AR system, users of the AR system showed significantly faster speed in locating important items and significantly fewer head movements. In terms of user acceptance, Syberfeldt et al. [16] suggested that AR is most beneficial for tasks that are complex enough to make the AR system worthwhile to the user. The researchers divided the steps involved in creating or maintaining a task into two categories: cognitive and psychomotor [17]. Cognitive activities include directing attention and associating commands with the physical task environment. Psychomotor activities include adjustment and other forms of physical manipulation. AR visualization has been studied to guide and support both types of tasks.

Common localization tasks place virtual icons or text labels on the physical object to be located [18]. If the enhancement is outside the field of view, it is helpful to provide an interface prompt to direct the user's attention to the position of the target component. To do this, Biocca et al. [19] developed the idea of the attention funnel, similar to tunnel-in-the-sky aviation cockpit head-up displays. This allows the user to notice objects that are completely invisible (including behind them), or that are obscured by other objects or walls. Schwerdtfeger et al. [20] compared several guides, including an attention funnel, 3D arrows, and compass-like 2D arrows. They suggested that the attention funnel should fade away as the user's gaze approaches the target so as not to block it.

Static arrows are used to point the user in the direction of the target in many methods. To guide the user, small arrows may be used to the upper right of known position markers in the room [21]. A solution with more precise tracking and thus more flexibility of guidance is proposed in Ref. [22], based on WiFi triangulation. The user interface for this work displays a 3D arrow at a fixed point on the screen. The red arrow rotates according to the relationship between the direction of the phone and the target's positions. Mulloni et al. [23] focused on the user interface design and information presented to the user. In addition to arrows, they also showed walking instructions, target steps, and turn information. Kim and Jun [24] linked the concepts of AR and indoor navigation and adopted a self-developed tracking algorithm. While tracking is based on markers and image sequence matching, which allows 3D presentation of AR data, users can only do so under the guidance of a 2D miniature map in the upper left corner of the AR display.

In terms of path guiding, many AR applications calculate the path between any pair of connected positions based on a building graph using some shortest path algorithm, e.g., Dijkstra's algorithm [23,

25, 26]. Such modules can dynamically recalculate the path to any target destination whenever the user reaches an arbitrary info point. However, when it comes to a dynamic target, the shortest path is no longer the optimal path, since the meeting position will not be the current target's position. To solve the dynamic target search problem, reliable target movement prediction is indispensable.

2.2 Probabilistic search

In navigation or guidance systems, target positioning is a major problem. GPS is already widely used outdoors, but buildings and walls can block satellite signals, making it inadequate for indoor positioning and navigation systems [27]. Early indoor navigation systems used sensors, consisting of a transmitter and a receiver, for tracking and positioning [28–30]. Either the transmitter or receiver can be fixed, while the other is on the moving user. Similarly, recent indoor navigation systems use mobile phones for positioning and navigation [31, 32]. Some studies use image information as markers, making it possible to integrate visual features of the environment without the need to place reference markers on walls or floors [21, 33].

In a guidance system, under normal circumstances, there is no position sensor installed on the target, and in general, the target's position is acquired by sensors, e.g., surveillance cameras, distributed in the real world environment. The problem of positioning a minimal set of fixed sensors in a known bounded environment in order to cover the whole area has been proven to be NP-hard in its general form [34]. Even if the number of sensors is predetermined, finding their optimal positions is still very complicated. In this article, we manually placed several Kinect DK cameras to obtain depth information, and used a body tracking application to identify the position of the target. The widespread use of depth cameras is a future development trend.

In robotics search tasks, the robot typically analyzes its environment and its own state in order to move sensors to obtain further useful information under certain constraints [35]. There is often significant uncertainty about the characteristics of the target, such as its position, direction, intent, and possibly even its existence. In this uncertain environment, most robotics target search research aims to optimize the trajectory of a given searcher so as to maximize the probability of detecting the target; the searcher is subject to certain constraints [36]. An effective planning method is needed to apply search theory to practical search tasks. Sato and Royset [37] proposed the optimal search path planning problem under resource constraints. In this paper, we assume that users will not become fatigued in the search process, allowing this paper to focus on the accuracy of navigation without considering how to allocate limited resources.

Assuming that the target moves independently of the searcher, the target model consists of a set of possible target tracks, each associated with a probability of the target taking that track. The number of possible tracks grows exponentially in time and space [38, 39]. Initiated by Koopman [40], several branch and bound methods have been developed, e.g., Martins [41] using a mean bound, Lau et al. [42] using a discounted mean bound, and Sato [43] using a static bound. Morin et al. [44] introduced a mixed-integer linear program with visibility constraint, in which agents are allowed to observe adjacent cells. Other research assumes that target moves to one of the cells adjacent to the current cell with equal probability [45, 46]. Firstly, however, these searches all consider detecting objects moving between ideal finite units in discrete time according to a Markov process. We solve the problem of searching for a walking target in a real corridor scene. Secondly, these methods do not dynamically update information about the target object or update the transfer probabilities according to the position of the target while going from the unknown target's position to the discovery of the target. Our method is that target object information is used to update the search with variable transition probabilities for the target's position over time.

In a cooperative system with multiple search teams, some strategies solve an overall optimization problem for all teams [39, 47], while some strategies exchange predicted observations on the optimized search path between platforms [48, 49]. As this paper does not realize a multi-camera and multi-target tracking system, and for the sake of user safety, this paper focuses on how to provide reliable direction indication for users in the task of seeking a single target, rather than a multi-user system. In the pursuit mission, the target may be antagonistic. As for adversarial pursuit-evasion games, if the user and the target have equal

maximum speeds, a single pursuer cannot capture an evader in fully connected maps. Bhattacharya and Hutchinson [50] examined the presence of Nash equilibria for the case of a single pursuer maintaining sight of an evader with bounded speed. They presented necessary and sufficient conditions for cases in which the target can escape the pursuer. These methods provide strategies for continuous tracking in surveillance applications. However, few algorithms for maintaining visibility have been implemented on real systems. For the above reasons and for safety reasons, we assume that the target does not the avoid user.

Most motion planning either considers how to search for the target object when user does not know the target object's position, or performs path planning when the target's position is known. In the real world, the target often appears intermittently in a fixed monitoring system. This paper studies the problem of how to guide the target object accurately when the position of the target object is updated intermittently. Therefore, we propose a target guidance algorithm based on a probabilistic search.

3 Method

3.1 Set up

The user searches in the augmented real world scene wearing a Microsoft HoloLens display (see Fig. 2(b)), which is connected to a local host. The user's positional information is updated in real time. Several Kinect depth cameras are distributed in the scene to capture views and for monitoring (see Fig. 2(a)). The target (without sensors) moves freely in the scene. When captured by a Kinect camera, the target's position and movement information is updated and sent to the local host. The real world scene is pre-calibrated using the fixed Kinect cameras and HoloLens display. Our method is implemented in a local area network environment.

When all hardware devices are ready, our AR guidance assistance method provides the user with a recommended search direction that is most likely to meet the target, and presents it in the HoloLens display (see Fig. 1(c)). The search direction is computed by our prediction graph search algorithm in real time. The PGS pipeline of the search algorithm is given in Section 3.2, and then we introduce in detail two crucial algorithms used in the pipeline, for *target prediction* and *probability calculation* in Sections 3.3 and 3.4, respectively.

(a) (b)

Fig. 2 (a) A Kinect camera and its affiliated devices are placed in the scene. (b) User wearing a HoloLens display, searching for the target.

3.2 PGS pipeline

Our method aims to provide users with continuous optimized direction guidance towards a dynamic target in real time. During the whole guidance process, the target's position P_t, target's direction D_t, target's velocity v_t, and target captured time stamp t_{target} are updated when the target is captured by the Kinect cameras, or remain unchanged otherwise. V_t is the average target velocity in the recorded period, at most 30 s. Meanwhile, the user's position P_u, and user's direction D_u are obtained through the HoloLens API, and the user's velocity V_u is calculated in the same way as V_t. We already know the real world scene layout and all walkable paths are extracted during a prepossessing stage. These paths are organized into a graph G, whose nodes represent corners and edges represent passageways. The total length of the walkable paths is L. The current system time stamp is t.

A target prediction state list structure is introduced to store all predicted target moving paths (Fig. 3). When the target passes through a passageway, the related information is packed as a passing state and stored in the list. The passageway is used as a key in the hash table to index the corresponding passing states. One passageway may index multiple states since the predicted target moving paths can loop. There is a pointer in the passing state points to the previous state to indicate where the target came from. By storing the search paths in this list,

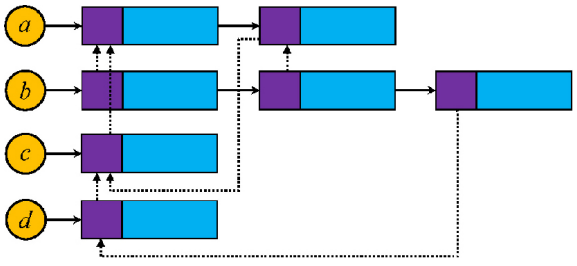


Fig. 3 Target prediction state list structure used in our PGS algorithm. The predicted target paths are organized and indexed for every passageway (a–d). Passing state data (rectangles) are attached to the corresponding passageways and contain a pointer (purple) to the previous passing state.

we can traverse all predicted passing states for any passageway quickly. The best user guidance direction D_{best} is calculated using Algorithm 1 and then shown on the HoloLens display. Starting from the current target's position, all possible paths are found in graph G, and the probability of each searched node is recorded in our target prediction state list (lines 1–3). Subsequently, all possible directions from the user's position are searched and the one with maximum probability is chosen as the guidance direction (lines 4–11).

For simplicity, we set the start position of the search to the position of the next intersection of the target P'_t (lines 1 and 2). Then we start searching from P'_t and update the target prediction state list (line 3). When the target arrives at an intersection in the scene, a node storing this state information about the target is inserted into the target prediction state list. Each node representing a state consists of four values: a pointer to its parent node, the target's position of the state, the time stamp when the target passed by, and the probability of going from the root node to the current node. The root node state S_t is initialized with its parent node pointer set to null, its position is set to P'_t, the time stamp is set to $t_{\text{target}} + \text{Distance}(P'_t, P_t)/v_t$ (the current time minus the time it takes the target to go from P'_t to P_t) and the probability is set to 1.0 (line 2). Starting from the root node, our method constructs the corresponding target prediction state list by calling function the TargetPrediction function discussed in Algorithm 2 (line 3). The parameters for this function are the root node state of the target S_t, target's velocity V_t, time stamp upper bound $t + L/v_t$, and the scene graph G. The time stamp upper bound limits the maximum search distance so as to not exceed the total path length L.

Algorithm 1 Guidance direction determination

Input: target's position P_t, target's direction D_t, target's velocity v_t, target available time stamp t_{target}, user's position P_u, user's direction D_u, user's velocity v_u, scene graph G, total length of walkable paths L, current system time stamp t

Output: guidance direction D_{best}

1: $P'_t \leftarrow \text{NextIntersection}(P_t, D_t, G)$
2: $S_t \leftarrow (\text{null}, P'_t, t_{\text{target}}+\text{Distance}(P'_t, P_t)/v_t, 1.0)$
3: $\text{TargetPrediction}(S_t, v_t, t + L/v_t, G)$
4: $pr_{\text{best}} \leftarrow 0$; $D_{\text{best}} \leftarrow \text{ShortestPathToCenter}(P_u, D_u)$
5: **for each** $P'_u \in \text{AdjacentIntersection}(P_u, G)$ **do**
6: $\quad S_u = (\text{null}, P'_u, t + \text{Distance}(P'_u, P_u)/v_u, 1.0)$
7: $\quad pr \leftarrow \text{ProbabilityCalculation}(S_u, v_u, t + L/v_u, G)$
8: $\quad$ **if** $pr > pr_{\text{best}}$ **then**
9: $\quad\quad pr_{\text{best}} \leftarrow pr$; $D_{\text{best}} \leftarrow \text{GetDirection}(P_u, P'_u)$
10: $\quad$ **end if**
11: **end for**
12: **return** D_{best}

Our method accumulates the probabilities of each path to its corresponding start direction, and then the direction with maximum probability pr_{best} is recorded in D_{best} and shown in the HoloLens as the guidance direction. Both values are initialized before starting optimization (line 4). Note that D_{best} is initialized to guide the user to take the shortest path to the center of the scene along the current user's direction, unless the current user's direction leads to a dead end. Naturally, this initial direction will be returned if the probability of capturing the target object in every direction is zero.

Starting from the current user's position P_u, we traverse all of its adjacent intersections (line 5). Then we initialize the user search root node state S_u (line 6), similarly to the target root node S_t. Next, the probability of the user meeting the target using the starting state S_u is calculated by the ProbabilityCalculation function detailed in Algorithm 3 (line 7). If the calculated probability is greater than the previous best, we update pr_{best} to pr and D_{best} to D (lines 8–10).

Finally, when all possible user's directions have been checked, our method returns the estimated best user walking direction (line 12), which is rendered in the HoloLens to guide the user to the target efficiently.

3.3 Target prediction

When a target moves freely in a real world scene, we can only obtain its position when it appears in

some camera view. Simply guiding the user to this position is futile: the future trajectory of the target is uncertain, and the target is likely to be far away from this position before the user arrives. Instead, our method provides a target prediction algorithm, from which possible future movements of the target can be predicted, along with corresponding probabilities. Subsequently, the most likely search direction to meet the target can be calculated and provided to the user.

The target prediction algorithm is shown in Algorithm 2. The inputs to the algorithm are the current target root node state S, target's velocity v, time stamp upper bound t_{limit}, and scene graph G. The algorithm searches all predicted possible target paths and updates them in the target prediction state list.

The algorithm is based on an iterative depth first search algorithm and stops when the time stamp in the current search state exceeds the time stamp upper bound t_{limit}, or the probability of the current state $S.pr$ is less than 0.001 (line 1). We search all possible adjacent intersections of $S.P$ in G and get the next node position P' (line 2), and then the corresponding node state S' of P' is updated by setting its previous state (parent node) to S, setting its position to P', setting its time stamp to $S.t$ plus the time for the target to move from P to P', and setting its probability to $S.pr$ times the probability of the target moving in direction D at position $S.P$ (line 3).

Our target prediction state list is maintained in a hash table, in which the key is a passageway represented by a position pair like $(S.P,\ P')$, and the value is a list of all possible target states that pass through the passageway $(S.P,\ P')$. As shown in Algorithm 2 (line 4), S' is inserted into the list marked with $(S.P,\ P')$. If the target is captured by a camera at position P' and the predicted state time is earlier than the current time (line 5), the node is deleted and the search skipped (line 6). Finally, we go to deeper branches for the new state S' (line 7).

In Fig. 4(c), the search paths are shown as a tree, in which nodes represent corners in the path and corresponding arrival time, and edges indicate the probability that the target passes through the corresponding passageway. For example, the target (shown as red spot) passes node B at time $t = 8$ s and continues moving, arriving at C at time $t = 18$ s with probability 0.5. When it arrives C, it may move to D with local probability of 0.5, so the overall probability of moving from C to D is 0.25. Note that we assume that the target never turns back unless it encounters a dead end.

Algorithm 2 Target prediction

Input: target search state S, target's velocity v, time stamp upper bound t_{limit}, scene graph G

1: **if** $S.t > t_{\text{limit}}$ **or** $S.pr < 0.001$ **then return**
2: **for each** $P' \in$ AdjacentIntersection$(S.P,\ G)$ **do**
3: $S' \leftarrow (S, P', S.t + \text{Distance}(S.P, P')/v, \text{GetProb}(S.P, P', G) * S.pr)$
4: InsertIntoList(S', GetStateList($S.P,\ P'$))
5: **if** $S'.t <$ GetTime() **and** KinectVisible(P') **then**
6: continue
7: TargePrediction($S',\ v,\ t_{\text{limit}},\ G$)
8: **end for**
9: **return**

3.4 Probability calculation

As in target prediction, the probability calculation process is also based on depth first search. A valid intersection between the target prediction status list and the user search path represent possible interception locations. The probabilities of all these

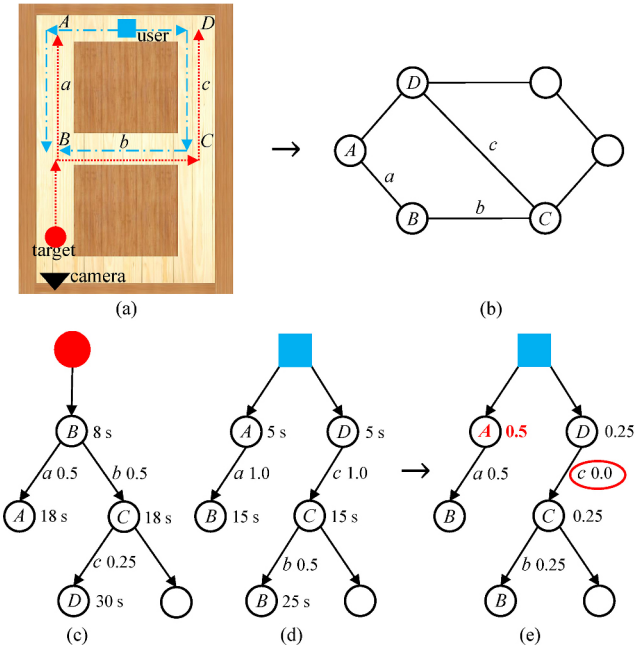


Fig. 4 A simple corridor scene layout is given (a) with its topological graph (b). The target prediction process is shown in a tree (c). Search time and probabilities are attached to nodes and edges of the tree. (d) shows the predicted user search process and (e) shows the probability backtracking process. Note that the probability of edge c is set to 0 as it is invalid (highlighted as a red circle).

intersection events are accumulated along their corresponding user's movement directions, and then the direction with the maximum probability is set to the final guidance direction. The probability of meeting the target is computed using Algorithm 3. Its inputs are the current user search state S_u, defined in Section 3.2, user's velocity v, time stamp upper bound t_{limit}, and the scene graph G. The output is the estimated probability of the user meeting the target face-to-face if starting in state S_u.

The stopping criteria for Algorithm 3 are defined similarly to Algorithm 2 (line 1) and the output probability is initialized to 0 (line 2). Then, we traverse all possible adjacent intersections of $S_u.P$, get the next intersection P', and create the corresponding next user search state S'_u (lines 3 and 4). We next check the target prediction state list to find all valid meetings in the passageway P' to $S_u.P$, and add the corresponding probability into the current search node (lines 5–8). Finally, we set state S'_u as the next user search state and iterate. The iteratively returned probability and the current branch probability are multiplied and added to pr (line 9). An example of the probability calculation process is shown in Figs. 4(d) and 4(e). Here, the iterative process of determining the search path and probability is similar to that in Algorithm 2. An accompanying accumulation backtracking process is performed to calculate the probabilities of all valid meetings and contributes them to the corresponding user search starting direction: see Fig. 4(e). Finally, the starting search direction with maximum probability (0.5 here) is returned as the user guidance direction.

Algorithm 3 Probability calculation

Input: user search state S_u, user's velocity v, time stamp upper bound t_{limit}, scene graph G
Output: probability from current user state S_u to target

1: **if** $S_u.t > t_{\text{limit}}$ **or** $S_u.pr < 0.001$ **then return** 0
2: $pr \leftarrow 0$
3: **for each** $P' \in$ AdjacentIntersection($S_u.P$, G) **do**
4: $S'_u \leftarrow (S_u, P', S_u.t + \text{Distance}(S_u.P, P')/v, \text{GetProb}(S_u.P, P', G) * S_u.pr)$
5: **for** $S'_t \in$ GetStateList(P', $S_u.P$) **do**
6: **if** CheckContributionValidity(S'_u, S'_t) **then**
7: $pr \leftarrow pr +$ GetProb($S_u.P$, P', G) * $S'_t.pr$
8: **end for**
9: $pr \leftarrow pr + \text{GetProb}(S_u.P, P', G) * \text{ProbabilityCalculation}(S'_u, v, t_{\text{limit}}, G)$
10: **end for**
11: **return** pr

The CheckContributionValidity function is used in Algorithm 3 to judge whether a meeting will happen or not. The first consideration is time: if a meeting occurs in a passageway, there should be an intersection between the passing time periods of the user and the target. As shown in Fig. 4(e), the probability on edge c changed from 1.0 in Fig. 4(d) to 0, because the time when the user searches from D to C is $t \in$ 5–15 s, while the time period when the target moves from C to D is $t \in$ 18–30 s, so there is no physical intersection in passageway c. A further consideration is that there should not be more than one meeting between the same target and user search path pair. See Fig. 5, the user and the target will meet in passageway a and again later in c if they keep moving along their own paths. Our method discards c in such cases.

Regarding speed, firstly, when the probability of reaching an edge in the search process is less than one in a thousand, we perform a pruning operation, so the number of edges in the search tree does not exceed one thousand. Secondly, our target prediction state list is maintained in a hash table, allowing direct retrieval when querying. Therefore, the method fully supports calculation of the guidance direction at 60 frames/s.

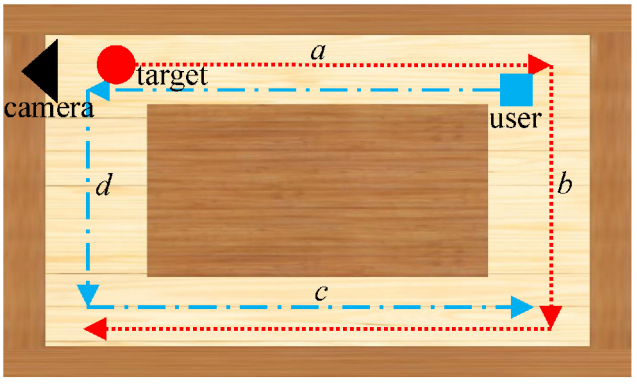


Fig. 5 Example of an invalid meeting state in passageway c: the corresponding paths of the user and the target already have an earlier intersection in passageway a.

4 User study

4.1 Outline

We have evaluated our guidance method in a controlled user study of the dynamic person searching task under one experimental condition (EC) and two

control conditions (CC_1, CC_2) detailed below. The mini-map condition is used as the second control condition because there are many way-finding studies using maps [5, 51, 52]. Our study is aimed to show that our guidance method is more efficient, requires less mental effort, and has better usability.

The three conditions were:

- *EC*: A series of arrows indicating a moving direction, calculated by our method, are shown to guide the user. Users are advised to move in the direction of the arrows. See Fig. 6(a).
- CC_1: No guidance is provided to users. They can move freely and should try to find the dynamic target by themselves.
- CC_2: A mini top-view map is provided to the users on the HoloLens screen. The user's position and orientation are marked in the map at all time. The target's position and movement direction are displayed in the map by an arrow when captured by the Kinect cameras. See Fig. 6(b).

4.2 Design

4.2.1 Participants

We recruited 16 participants, 12 males and 4 females, between 20 and 40 years old. Five of our participants had prior experience with AR applications. Participants had normal or corrected vision and none reported preexisting balance disorders.

4.2.2 Implementation

The AR application was developed using Unity v2019.4 and displayed on a Microsoft HoloLens. Human motion is captured using a skeleton tracking technique with the Microsoft Azure Kinect depth camera. Microsoft provides a skeleton tracking software development kit (SDK) [53], which was used to determine the target's position and movement. Each Kinect was connected to a laptop computer for this purpose. A server workstation with 3.6 GHz Intel i7-9900KF CPU, 32 GB RAM, and an NVIDIA RTX 2080 graphics card received the tracking results, and ran our method, sending instruction to the HoloLens 50 times/s. The AR content in the HoloLens is rendered at 60 frames/s for each eye.

We used a HoloLens 2 as the augmented reality device for reliable tracking of users. Through a one-time localization, the real world indoor environment is aligned with our building model.

The optimal number and placement of cameras were not considered here. This paper aims to explore whether the search algorithm can accelerate the search for the target, rather than focusing on how to place cameras. We manually placed the cameras in position that covered multiple passageways. Principles followed included trying to put them at corners instead of the middles of passageways, and on corners where the target has a high probability of passing instead of remote ones.

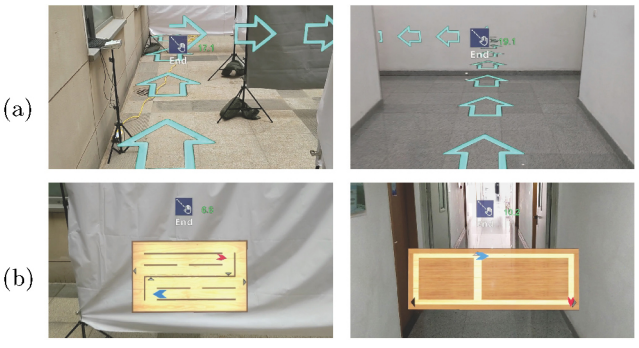


Fig. 6 (a) First-person view in *EC*. A series of arrows indicates the guidance direction provided by our method. (b) First-person view in CC_2. A mini top-view map is provided to show the positions of the user (blue arrow) and the target (red arrow). The depth cameras are marked by black triangles.

4.2.3 Environments and tasks

Three scenes were used in our user study, which were designed to explore the influence of different space dimensions on our method. The walkable regions of these scenes were initially extracted manually.

- *Scene 1.* Here, the user needs to search for a moving person in a hand-built maze which covers a 6 m × 9 m area (see Fig. 7(a)). Two cameras are placed in the scene, which has 10 branching nodes, including right-angled corners that connect two sections of the passageway.
- *Scene 2.* This is also a hand-built maze, covering a 10 m × 16 m area (see Fig. 7(b)). Four cameras are placed in the scene, which has 18 branching nodes. It is 3 times the area of Scene 1 and more complicated.
- *Scene 3.* The third scene is an indoor corridor. From real buildings, 2D walkable areas were extracted from the building plan. The area of the passageway, including the enclosed area, is 20 m × 64 m (see Fig. 1(a)). It is 23 times as big as Scene 1, but only has 6 branching nodes—its structure is relatively simple. Three cameras are placed in this scene.

There were also two tasks:

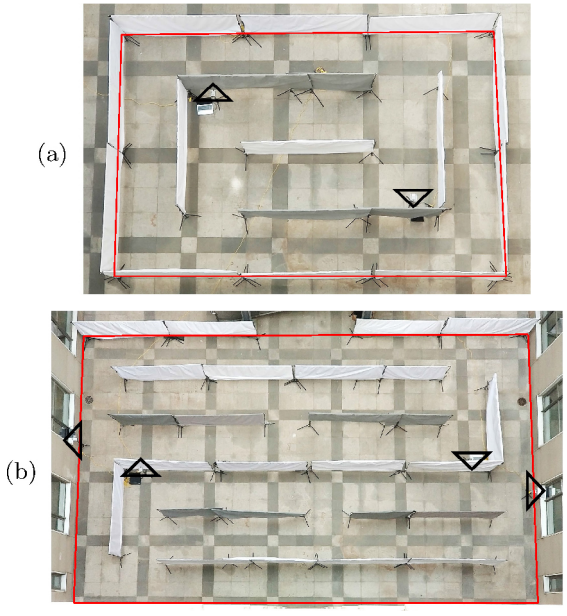


Fig. 7 Top views of (a) Scene 1 and (b) Scene 2. The walkable area in each lies within the red rectangle. The Kinect cameras' positions and orientations are marked with black triangles.

- *Task 1.* The target person roams freely in the scene along a random walking route. The user is asked to meet with the target as soon as possible. This task adopts a within-subject design, with each participant serving once in each of three conditions for each scene. The order of the conditions of each participant is random for each scene. We have no requirements for the speed of users or targets. They walk at normal speed according to their habits. Sixteen participants were used in this task. For each participant, the target person was randomly selected from the other participants and remained unchanged in all three conditions of this task. If an experiment failed, it was repeated. This means that there are sixteen pieces of data of each condition for each scene.
- *Task 2.* The target person is required to walk along a predefined route in the scene. The user is asked to meet with the target as soon as possible. This task adopts a between-subject design. One participant was randomly selected as the target and remained unchanged in all experiments of this task. The other 15 participants were randomly assigned to one experimental group (EG) and two control groups (CG_1, CG_2), each having 5 participants. Each participant served in one group for each scene. The EG has the same AR interface as the EC. The two CGs have the same AR interface as the CCs. Each scene has three predefined loop paths illustrated in red, green, purple in Fig. 8. Each group performs each predefined path for each scene. The starting positions of the user and the target are marked in Fig. 8. The target repeatedly walks clockwise along the route. Each group of each route for each scene has five pieces of data.

4.3 Procedure

Before experimenting with each scene, we provided participants with a three-minute tutorial. We showed users the top view of the scene and let them walk around the scene to familiarize themselves with it. Then we showed the user how to control the HoloLens used in the three conditions. During execution of the tasks, the system displays different information under

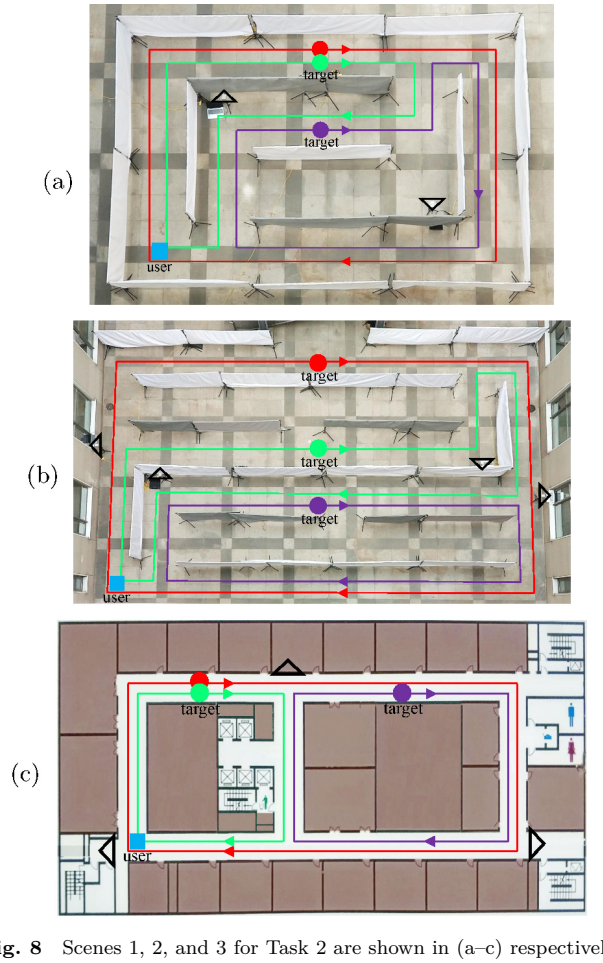


Fig. 8 Scenes 1, 2, and 3 for Task 2 are shown in (a–c) respectively. Each scene has three predefined loop paths shown in red, green, purple. The starting positions of target and user are marked as circle and square respectively. The direction of the target's movement is clockwise, as marked by arrows. Targets walk along the path repeatedly until they meet the user.

different conditions. The user can choose to take a break after each experiment. Once the user tapped the start button on the HoloLens screen, the program drew different prompts on the screen according to the situation. When the user meets the target, the user taps the end button in the HoloLens screen to stop the experiment. Light background music was played to avoid the influence of the sounds of footsteps.

4.4 Results

We quantified the results of our user study using task performance metrics and perceptual metrics.

4.4.1 Task performance metrics

The performance was measured with the following metrics for each task, and for each condition.

- *Completion distance:* The distance the user travels to intercept the target, measured in meters.
- *Completion time:* Elapsed time to complete the task, measured in seconds.
- *Time in place:* The total amount of time the user stays in place while performing the task, measured in seconds.

For these three indicators, the smaller the value, the more effective our method. The best results are indicated in bold for the three conditions of these indicators. The performance metrics of Task 1 are shown in Tables 1, 2, and 3 for details.

Table 1, column three reports the average and standard deviation of completion distance in Task 1 over all participants, for each scene and each condition. Column four gives the relative decrease of the completion distance from CC to EC. Column five reports the relative difference between the EC and CC. We use p-value as statistical test and mark the upper corner of statistical significant results with an asterisk. We also report Cohen's d in column six, along with the conventional effect size quantification in column seven [54, 55]. The d values were translated to qualitative effect size estimates according to Huge ($d > 2.0$), Very Large ($2.0 > d > 1.2$), Large ($1.2 > d > 0.8$), Medium ($0.8 > d > 0.5$), Small ($0.5 > d > 0.2$), and Very Small ($0.2 > d > 0.01$).

The results in Table 1 show that, although our optimization goal is to maximize the probability of meeting the target, our method can also reduce the

Table 1 Completion distance (m) for Task 1

	Condition	Avg ± std.dev.	$(CC_i - EC)/CC_i$	p	Cohen's d	Effect size
Scene 1	EC	**16.1±2.9**	—	—	—	—
	CC_1	23.5±6.1	31.5%	$< 0.001^*$	1.55	Very Large
	CC_2	20.1±4.3	20.2%	0.002^*	1.10	Large
Scene 2	EC	**23.8±5.8**	—	—	—	—
	CC_1	38.1±9.0	37.5%	$< 0.001^*$	1.88	Very Large
	CC_2	32.7±8.1	27.4%	$< 0.001^*$	1.27	Very Large
Scene 3	EC	**66.3±16**	—	—	—	—
	CC_1	106±27	37.5%	$< 0.001^*$	1.77	Very Large
	CC_2	85.2±19	22.2%	0.002^*	1.05	Large

Table 2 Completion time (s) for Task 1

	Condition	Avg ± std.dev.	$(CC_i - EC)/CC_i$	p	Cohen's d	Effect size
Scene 1	EC	**22.6±4.9**	—	—	—	—
	CC_1	34.4±9.6	34.3%	$< 0.001^*$	1.55	Very Large
	CC_2	28.9±6.0	21.7%	0.001^*	1.14	Large
Scene 2	EC	**32.0±8.2**	—	—	—	—
	CC_1	52.5±13	39.0%	$< 0.001^*$	1.87	Very Large
	CC_2	46.3±9.6	30.8%	$< 0.001^*$	1.59	Very Large
Scene 3	EC	**75.6±22**	—	—	—	—
	CC_1	121±37	37.7%	$< 0.001^*$	1.49	Very Large
	CC_2	97.8±30	22.7%	0.013^*	0.84	Large

distance that users need to walk to complete the task. The advantage of EC over the two CCs using this metric is always statistically significant (i.e., $p < 0.05$) and the effect size is mostly Very Large.

Table 2 reports the average and standard deviation of completion time for Task 1 over all participants, for each scene and each condition. Both completion time and completion distance of Task 1 are normally distributed as confirmed by a Shapiro–Wilk test [56], so the conditions were compared using ANOVA. We used a one-way repeated measure ANOVA and Bonferroni correction for pair-wise comparisons. We also used a Greenhouse–Geisser adjustment to correct violations of the spherical hypothesis. The following are the results of the data analysis.

For all three scenes, participants had the shortest average of completion time in EC in which the user walks along the guiding path with higher probability of catching the target. The advantage of EC over two CCs on this metric is always statistically significant (i.e., $p < 0.05$) and the effect size is mostly Very Large. The condition which has the longest average completion time is CC_1: in this case the user does not get any assistance and may walk to useless places repeatedly.

Table 3 gives the time spent in place. The Shapiro–Wilk test reveals that this is not normally distributed, so we performed a statistical analysis of the differences between EC and CCs using a Wilcoxon signed-rank test [57]; p values are in column five. Participants had the shortest average time of staying in place for all three scenes and there is a statistically significant (i.e., $p < 0.05$) difference between EC and the two CCs.

In EC, when the cameras capture the target, the best direction for the user to follow can be immediately calculated and presented, while CC_2 requires the user to recognize the target's position in the top view map, to switch from the user's egocentric perspective to the exocentric perspective provided by the map, to predict the movement path of the target, and to plan their own walking path—so the reaction time increases. Furthermore, when the target has disappeared from the cameras for a certain period of time, in EC, the historical data can be used to calculate the average velocity of the target for path prediction. However, in CC_2, users need to infer the position of the object from their own memory, which can sometimes be unreliable. It is hard for them to deduce the expected position and make the correct decision.

To further verify the results, we choose three predefined paths to control the movement of the target and test the performance of our method. Table 4 reports the average and the standard deviation of the completion time for Task 2 for each scene and each group. Columns three–five report the completion time for red, green, and blue loop paths respectively. Column six gives the average and standard deviation over all three predefined paths. Column seven gives the relative decrease of the completion time of Task 2 from CG to EG. For all three scenes and all three predefined paths, participants used the shortest average time to intercept the target in the experimental group EG. The overall average of completion time in each group in Task 2 is longer than that in each condition in Task 1, because the red and purple paths require the user to walk a longer distance around the scene rather than cutting across the middle of the scene. In Task 2 the average decrease in completion time from CG_1 to EG over the three scenes is 33.3%. This average decrease ratio from CG_2 to EG among three scenes is

Table 3 Time (s) that user stays in place for Task 1

	Condition	Avg ± std.dev.	$(CC_i - EC)/CC_i$	p	Cohen's d	Effect size
Scene 1	EC	**1.43±0.73**	—	—	—	—
	CC_1	2.12±1.2	32.5%	0.039*	0.70	Medium
	CC_2	2.32±0.98	38.5%	0.003*	1.04	Large
Scene 2	EC	**2.57±1.0**	—	—	—	—
	CC_1	4.18±1.7	38.5%	0.001*	1.16	Large
	CC_2	4.77±1.7	46.0%	< 0.001*	1.56	Very Large
Scene 3	EC	**3.73±1.2**	—	—	—	—
	CC_1	6.96±2.6	46.4%	< 0.001*	1.58	Very Large
	CC_2	7.65±3.1	51.3%	< 0.001*	1.69	Very Large

Table 4 Completion time (s) for Task 2

	Condition	Red path Avg ± std.dev.	Green path Avg ± std.dev.	Blue path Avg ± std.dev.	Overall Avg ± std.dev.	$(CC_i - EC)/CC_i$
Scene 1	*EG*	**24.2±2.6**	**19.5±1.8**	**25.1±2.4**	**23.0±2.3**	—
	CG_1	36.2±5.2	35.7±4.7	37.4±5.9	36.4±5.3	37.0%
	CG_2	30.4±3.6	26.7±2.8	29.1±3.3	28.8±3.2	20.1%
Scene 2	*EG*	**38.4±4.3**	**32.3±3.7**	**34.4±4.0**	**35.0±4.0**	—
	CG_1	58.2±7.0	51.4±6.3	47.7±6.0	52.4±6.4	33.2%
	CG_2	51.3±5.3	47.8±4.8	42.2±4.4	47.1±4.8	25.6%
Scene 3	*EG*	**92.1±12**	**57.5±7.4**	**110±14**	**86.5±11**	—
	CG_1	138±23	97.3±16	134±22	123±20	30.1%
	CG_2	107±14	70.5±8.8	124±16	101±13	14.5%

20.1%, less than that for Task 1 because it is easier for the user to plan a path using the top-view map when the target walks around the edge of the scene than when the target passes through the intersection in the middle of the scene. In this situation the guidance path of our method is similar to the path analyzed by the user through the top-view map, especially for Scene 3 which has a relatively simple structure. The results show that compared to free searching and the top-view method, our method can help users intercept the target more effectively when the target moves along various common paths.

4.4.2 Perceptual metrics

We also employed task load and usability questionnaires to evaluate our method's practicality.

We measured task load using the Raw Task Load Index (TLX) questionnaire, a simplified version of the NASA-TLX questionnaire [58]. The results are given in Table 5. We averaged the scores of the six Raw TLX task load questions, which are between 5 and 100, with 5 point increments. The task load values do not follow a normal distribution, so the differences were analyzed with a Wilcoxon signed-rank test.

In all three scenarios, the task load of the *EC* was smaller than that of the *CC*. The task load of *EC* has no significant difference compared to CC_1 and CC_2 for Scene 1, which is the smallest of the three scenes and users do not need to explore for a long time.

The task load is significantly higher for CC_1 compared to *EC* for Scene 2 ($p = 0.021$) and Scene 3 ($p = 0.003$). Since there is no target information in CC_1, the users need to analyze the scene structure and search for the target object by themselves, which takes more effort and increases frustration. In Scene 3,

Table 5 Raw Task Load Index score

	Condition	Avg ± std.dev.	p	Cohen's d	Effect size
Scene 1	*EC*	**19.8±3.2**	—	—	—
	CC_1	21.9±5.8	0.245	0.45	Small
	CC_2	23.2±6.9	0.087	0.64	Medium
Scene 2	*EC*	**21.3±4.9**	—	—	—
	CC_1	29.0±9.8	0.021*	0.99	Large
	CC_2	27.4±10	0.076	0.74	Medium
Scene 3	*EC*	**30.3±9.1**	—	—	—
	CC_1	48.9±20	0.003*	1.21	Very Large
	CC_2	45.8±21	0.015*	0.95	Large

there is a statistically significant difference between *EC* and CC_2 ($p = 0.015$): the passageway is long, and the cost of wrong user judgment increases, making users more cautious and placing them under more pressure when making judgments. This increases mental demands and frustration.

Overall, the *EC* strategy does not bring more load to users compared to two *CC*s. The advantage of our method over the two conventional methods confirms that users can find the target more easily when using our AR path guidance assistance.

Each participant was asked to fill out a usability questionnaire after each condition. It has eight questions pertaining to the experience of using the AR assistance. The questions were to be answered using a numerical score from 1 to 10, and were

Q1. Was the interface confusing?
Q2. Will you be able to use this guidance in the future?
Q3. Was it convenient to operate the device?
Q4. Was it difficult to utilize the guidance information?
Q5. Do you think others will soon adapt to this guidance mode?

Q6. Was the guidance mode reasonable?
Q7. Was the guidance mode tiresome?
Q8. Did the guidance help you find the target quickly?

Three questions are negative, and five are positive, to reduce the risk of mechanical answers.

Results for the individual questions as well as the average over all questions are given in Fig. 9. A negative score x is replaced by its complement $11 - x$ so that larger scores are always more favorable. The scores are not normally distributed and differences were analyzed with a Wilcoxon signed-rank test; significant differences are indicated by an asterisk. Participants indicated that they would like and will be able to use our AR assistance in the future (Q2). EC was judged as providing significantly easier to utilize guidance information than CC_1 and CC_2 (Q4), confirming that our path guidance provides intuitive help when searching for a target. Since EC does not require the user to think too much, while CC_1 does not have any hints that will make the user feel confused or bored, EC scores significantly higher than CC_1 in Q6 and Q7. CC_2 provides users with map information and partial target information, improving the user's participation and reducing anxiety, and while EC rates higher than CC_2 in both scores, the results are not significant. Participants rate EC as significantly more useful for guiding them to intercept the target quickly (Q8). Finally, over all questions, EC received significantly higher scores than CC_1 and CC_2, confirming that participants note the benefits of using our AR assistance in the target search task.

5 Conclusions

We have presented a method for improving dynamic target search efficiency in the real world through PGS based AR assistance. In essence, our method provides the user with an optimized instantaneous direction cue to lead the user in a direction most likely to intercept the dynamic target. Our method obtains dynamic target's positions captured by depth cameras fixed in the real scene, predicts possible interception locations, and generates an instantaneous direction to help users find the dynamic target. We also designed a user study to evaluate our method. Its results show that compared to free search and a top-view method, our method can help users find the dynamic target more effectively, and significantly reduce task completion time, walking distance, and time that the user stays in place. The method also shows an obvious task load reducing and usability score promoting, some of the improvements are significant.

One limitation of our method is that if the target does not appear in any camera for a long time, or does not move, our method may not work well. In the worst case, our method may degenerate into a non-guidance method. One solution is to increase the number of cameras so that even if the target is lost from the cameras, we can guide users to search in a relatively small surveillance blind area. Another limitation is that Kinect's recognition distance is 6 m, which limits the use scenario. Huang et al. [59] provided a practical backend for stereo visual simultaneous localization and mapping (SLAM) which can discover individual rigid bodies and compute their

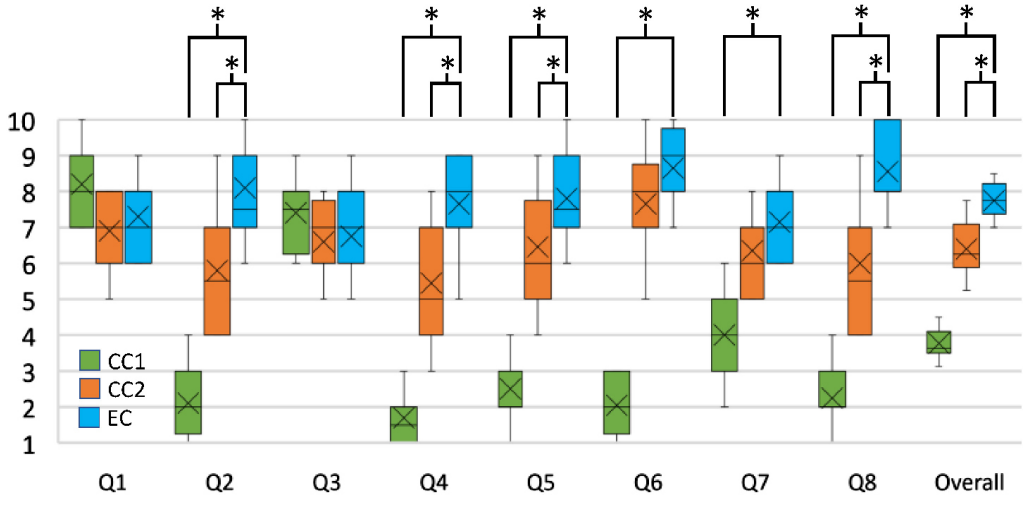


Fig. 9 Usability questionnaire scores, the higher, the better. Significant differences are indicated by asterisks.

motions in dynamic environments; it is not limited to indoor scenes. Such motion estimation methods can expand the application scenarios of our paper.

The current work sets the branch probabilities equally, although the target may be biased when choosing a branch passageway. In future work, we hope to combine data-driven trajectory analysis and attention distribution models into the algorithm to more accurately predict the likely trajectory of the target, so that our algorithm can deal with more complex environments and movement patterns. Other future work will focus on extending our method to multiple targets. Since multi-target multi-camera tracking remains a very complex and difficult problem [60, 61], we have not yet carried out experiments with multiple targets. Once there is a lightweight solution to track and differentiate multiple targets, we hope to apply our method to multi-target search tasks.

Acknowledgements

We sincerely thank the reviewers for their their constructive suggestions and comments. This work was supported by National Key R&D Program of China (2019YFC1521102) and the National Natural Science Foundation of China (61932003 and 61772051).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Electronic Supplementary Material

Supplementary material is available in the online version of this article at `https://doi.org/10.1007/s41095-021-0266-0`.

References

[1] Lynch, K. Reconsidering the image of the city. In: *Cities of the Mind. Environment, Development, and Public Policy*. Rodwin, L.; Hollister, R. M. Eds. Springer Boston MA, 151–161, 1984.

[2] LaViola Jr., J. J.; Kruijff, E.; McMahan, R. P.; Bowman, D.; Poupyrev, I. P. *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2017.

[3] Pierce, J. S.; Pausch, R. Navigation with place representations and visible landmarks. In: Proceedings of the IEEE Virtual Reality, 173–288, 2004.

[4] Steck, S. D.; Mallot, H. A. The role of global and local landmarks in virtual environment navigation. *Presence: Teleoperators and Virtual Environments* Vol. 9, No. 1, 69–83, 2000.

[5] Darken, R. P.; Cevik, H. Map usage in virtual environments: Orientation issues. *Proceedings IEEE Virtual Reality* 133–140, 1999.

[6] Stoakley, R.; Conway, M. J.; Pausch, R. Virtual reality on a WIM: Interactive worlds in miniature. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 265–272, 1995.

[7] Darken, R. P.; Sibert, J. L. A toolset for navigation in virtual environments. In: Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology, 157–165, 1993.

[8] Grammenos, D.; Filou, M.; Papadakos, P.; Stephanidis, C. Virtual prints: Leaving trails in virtual environments. In: Proceedings of the Workshop on Virtual Environments, 131–138, 2002.

[9] Chittaro, L.; Ranon, R.; Ieronutti, L. Guiding visitors of Web3D worlds through automatically generated Tours. In: Proceedings of the 8th International Conference on 3D Web Technology, 27–38, 2003.

[10] Elmqvist, N.; Tudoreanu, M. E.; Tsigas, P. Tour generation for exploration of 3D virtual environments. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 207–210, 2007.

[11] Wang, M.; Lyu, X. Q.; Li, Y. J.; Zhang, F. L. VR content creation and exploration with deep learning: A survey. *Computational Visual Media* Vol. 6, No. 1, 3–28, 2020.

[12] Henderson, S.; Feiner, S. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics* Vol. 17, No. 10, 1355–1368, 2011.

[13] Webel, S.; Bockholt, U.; Engelke, T.; Gavish, N.; Olbrich, M.; Preusche, C. An augmented reality training platform for assembly and maintenance skills. *Robotics and Autonomous Systems* Vol. 61, No. 4, 398–403, 2013.

[14] Barakonyi, I.; Schmalstieg, D. Ubiquitous animated agents for augmented reality. In: Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality, 145–154, 2006.

[15] Zauner, J.; Haller, M.; Brandl, A.; Hartman, W. Authoring of a mixed reality assembly instructor for hierarchical structures. In: Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, 237–246, 2003.

[16] Syberfeldt, A.; Danielsson, O.; Holm, M.; Wang, L. H. Visual assembling guidance using augmented reality. *Procedia Manufacturing* Vol. 1, 98–109, 2015.

[17] Neumann, U.; Majoros, A. Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance. In: Proceedings of the IEEE Virtual Reality Annual International Symposium, 4–11, 1998.

[18] Nassani, A.; Bai, H. D.; Lee, G.; Billinghurst, M. Tag it!: AR annotation using wearable sensors. In: Proceedings of the SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications, 1–4, 2015.

[19] Biocca, F.; Tang, A.; Owen, C.; Xiao, F. Attention funnel: Omnidirectional 3D cursor for mobile augmented reality platforms. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1115–1122, 2006.

[20] Schwerdtfeger, B.; Reif, R.; Günthner, W. A.; Klinker, G. Pick-by-vision: There is something to pick at the end of the augmented tunnel. *Virtual Reality* Vol. 15, Nos. 2–3, 213–223, 2011.

[21] Kasprzak, S.; Komninos, A.; Barrie, P. Feature-based indoor navigation using augmented reality. In: Proceedings of the 9th International Conference on Intelligent Environments, 100–107, 2013.

[22] Alnabhan, A.; Tomaszewski, B. INSAR: Indoor navigation system using augmented reality. In: Proceedings of the 6th ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness, 36–43, 2014.

[23] Mulloni, A.; Seichter, H.; Schmalstieg, D. Handheld augmented reality indoor navigation with activity-based instructions. In: Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, 211–220, 2011.

[24] Kim, J.; Jun, H. Vision-based location positioning using augmented reality for indoor navigation. *IEEE Transactions on Consumer Electronics* Vol. 54, No. 3, 954–962, 2008.

[25] Rehman, U.; Cao, S. Augmented-reality-based indoor navigation: A comparative analysis of handheld devices versus google glass. *IEEE Transactions on Human-Machine Systems* Vol. 47, No. 1, 140–151, 2017.

[26] Subakti, H.; Jiang, J. R. A marker-based cyber-physical augmented-reality indoor guidance system for smart campuses. In: Proceedings of the IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, 1373–1379, 2016.

[27] Van Diggelen, F. S. T. *A-GPS: Assisted GPS, GNSS, and SBAS*. Artech House, 2009.

[28] Want, R.; Hopper, A.; Falcão, V.; Gibbons, J. The active badge location system. *ACM Transactions on Information Systems* Vol. 10, No. 1, 91–102, 1992.

[29] Fukuju, Y.; Minami, M.; Morikawa, H.; Aoyama, T. DOLPHIN: An autonomous indoor positioning system in ubiquitous computing environment. In: Proceedings IEEE Workshop on Software Technologies for Future Embedded Systems, 53–56, 2003.

[30] Minami, M.; Fukuju, Y.; Hirasawa, K.; Yokoyama, S.; Mizumachi, M.; Morikawa, H.; Aoyama, T. DOLPHIN: A practical approach for implementing a fully distributed indoor ultrasonic positioning system. In: *UbiComp 2004: Ubiquitous Computing. Lecture Notes in Computer Science, Vol. 3205*. Davies, N.; Mynatt, E. D.; Siio, I. Eds. Springer Berlin Heidelberg, 347–365, 2004.

[31] Liu, M. Y.; Liu, K.; Yang, P. P.; Lei, X. K.; Li, H. Bio-inspired navigation based on geomagnetic. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics, 2339–2344, 2013.

[32] Rubino, I.; Barberis, C.; Di Chio, L.; Xhembulla, J.; Malnati, G. Enhancing a museum mobile application through user experience design: A comparative analysis. *Recent Advances in Electrical & Electronic Engineering* 295–300, 2014.

[33] Delail, B. A.; Weruaga, L.; Zemerly, M. J. CAViAR: Context aware visual indoor augmented reality for a university campus. In: Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, 286–290, 2012.

[34] O'rourke, J. *Art Gallery Theorems and Algorithms, Vol. 57*. Oxford University Press, 1987.

[35] Zeng, R.; Wen, Y. H.; Zhao, W.; Liu, Y. J. View planning in robot active vision: A survey of systems, algorithms, and applications. *Computational Visual Media* Vol. 6, No. 3, 225–245, 2020.

[36] Foraker, J.; Royset, J. O.; Kaminer, I. Search-trajectory optimization: Part I, formulation and theory. *Journal of Optimization Theory and Applications* Vol. 169, No. 2, 530–549, 2016.

[37] Sato, H.; Royset, J. O. Path optimization for the resource-constrained searcher. *Naval Research Logistics* Vol. 57, No. 5, 422–440, 2010.

[38] Kratzke, T. M.; Stone, L. D.; Frost, J. R. Search and rescue optimal planning system. In: Proceedings of the 13th International Conference on Information Fusion, 1–8, 2010.

[39] Royset, J. O.; Sato, H. Route optimization for multiple searchers. *Naval Research Logistics* Vol. 57, No. 8, 701–717, 2010.

[40] Koopman, B. O. *Search and Screening: General Principles with Historical Applications*. Pergamon Press, 1980.

[41] Martins, G. H. A new branch-and-bound procedure for computing optimal search paths. Technical Report. Naval Postgraduate School Monterey CA, 1993. Available at https://apps.dtic.mil/sti/citations/ADA265276.

[42] Lau, H.; Huang, S. D.; Dissanayake, G. Discounted MEAN bound for the optimal searcher path problem with non-uniform travel times. *European Journal of Operational Research* Vol. 190, No. 2, 383–397, 2008.

[43] Sato, H. Path optimization for single and multiple searchers: Models and algorithms. Technical Report. Naval Postgraduate School Monterey CA, 2008. Available at https://apps.dtic.mil/sti/citations/ADA488991.

[44] Morin, M.; Abi-Zeid, I.; Lang, P.; Lamontagne, L.; Maupin, P. The optimal searcher path problem with a visibility criterion in discrete time and space. In: Proceedings of the 12th International Conference on Information Fusion, 2217–2224, 2009.

[45] Peng, H.; Huo, M. L.; Liu, Z. Z.; Xu, W. Simulation analysis of cooperative target search strategies for multiple UAVs. In: Proceedings of the 27th Chinese Control and Decision Conference, 4855–4859, 2015.

[46] Hu, J. W.; Xie, L. H.; Xu, J. Vision-based multi-agent cooperative target search. In: Proceedings of the 12th International Conference on Control Automation Robotics & Vision, 895–900, 2012.

[47] Perez-Carabaza, S.; Bermudez-Ortega, J.; Besada-Portas, E.; Lopez-Orozco, J. A.; de la Cruz, J. M. A multi-UAV minimum time search planner based on ACO_R. In: Proceedings of the Genetic and Evolutionary Computation Conference, 35–42, 2017.

[48] Meng, W.; He, Z. R.; Su, R.; Yadav, P. K.; Teo, R.; Xie, L. H. Decentralized multi-UAV flight autonomy for moving convoys search and track. *IEEE Transactions on Control Systems Technology* Vol. 25, No. 4, 1480–1487, 2017.

[49] Hu, J. W.; Xie, L. H.; Xu, J.; Xu, Z. Multi-agent cooperative target search. *Sensors* Vol. 14, No. 6, 9408–9428, 2014.

[50] Bhattacharya, S.; Hutchinson, S. On the existence of Nash equilibrium for a two player pursuit-evasion game with visibility constraints. In: *Algorithmic Foundation of Robotics VIII. Springer Tracts in Advanced Robotics, Vol. 57*. Chirikjian, G. S.; Choset, H.; Morales, M.; Murphey, T. Eds. Springer Berlin Heidelberg, 251–265, 2009.

[51] Darken, R. P.; Peterson, B. Spatial orientation, wayfinding, and representation. In: *Handbook of Virtual Environment Technology*. Stanney, K. Ed. CRC Press, 533–558, 2002.

[52] Kraus, M.; Schäfer, H.; Meschenmoser, P.; Schweitzer, D.; Keim, D. A.; Sedlmair, M.; Fuchs, J. A comparative study of orientation support tools in virtual reality environments with virtual teleportation. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 227–238, 2020.

[53] Microsoft Azure Kinect DK documentation. 2020. Available at https://docs.microsoft.com/en-us/azure/kinect-dk/.

[54] Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*. Academic Press, 2013.

[55] Sawilowsky, S. S. New effect size rules of thumb. *Journal of Modern Applied Statistical Methods* Vol. 8, No. 2, 597–599, 2009.

[56] Shapiro, S. S.; Wilk, M. B. An analysis of variance test for normality (complete samples). *Biometrika* Vol. 52, Nos. 3–4, 591–611, 1965.

[57] Rey, D.; Neuhäuser, M. Wilcoxon-signed-rank test. In: *International Encyclopedia of Statistical Science*. Lovric, M. Ed. Springer Berlin Heidelberg, 1658–1659, 2011.

[58] Hart, S. G. Nasa-task load index (NASA-TLX); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* Vol. 50, No. 9, 904–908, 2006.

[59] Huang, J. H.; Yang, S.; Zhao, Z. S.; Lai, Y. K.; Hu, S. M. ClusterSLAM: A SLAM backend for simultaneous rigid body clustering and motion estimation. *Computational Visual Media* Vol. 7, No. 1, 87–101, 2021.

[60] He, Y. H.; Wei, X.; Hong, X. P.; Shi, W. W.; Gong, Y. H. Multi-target multi-camera tracking by tracklet-to-target assignment. *IEEE Transactions on Image Processing* Vol. 29, 5191–5205, 2020.

[61] Li, P.; Zhang, J. B.; Zhu, Z.; Li, Y. W.; Jiang, L.; Huang, G. State-aware re-identification feature for multi-target multi-camera tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 1506–1516, 2019.

Zixiang Zhao is an M.A. student in the School of Computer Science and Engineering of Beihang University. His current research focuses on virtual reality, augmented reality, and visualization.

Jian Wu is a Ph.D. student in the School of Computer Science and Engineering of Beihang University. His current research focuses on virtual reality, augmented reality, and visualization.

Lili Wang received her Ph.D. degree from Beihang University where she is now a professor with the School of Computer Science and Engineering, and a researcher with the State Key Laboratory of Virtual Reality Technology and Systems. Her interests include virtual reality, augmented reality, mixed reality, real-time rendering, and realistic rendering.