·Article·

Dynamic targets searching assistance based on virtual camera priority

Zixiang ZHAO[1], Quanwei ZHOU[1], Xiaoguang HAN[4,5], Lili WANG[1,2,3*]

1. *State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing* 100191*, China*

2. *Beijing Advanced Innovation Center for Biomedical Engineering, Beihang University, Beijing* 100191*, China*

3. *Department of Mathematics and Theories, Peng Cheng Laboratory, Shenzhen* 518052*, China*

4. *The Chinese University of Hong Kong, Shenzhen* 999077*, China*

5. *Shenzhen Research Institute of Big Data, Shenzhen* 999077*, China*

*** Corresponding author,** wanglily@buaa.edu.cn

Received: 12 August 2021 **Accepted:** 18 October 2021

Supported by National Key R&D plan (2019YFC1521102); the National Natural Science Foundation of China (61932003, 61772051, 62172437); the Beijing Natural Science Foundation (L182016); the Beijing Program for International S&T Cooperation (Z191100001619003); the funding of Shenzhen Research Institute of Big Data (Shenzhen 518000).

Citation: Zixiang ZHAO, Quanwei ZHOU, Xiaoguang HAN, Lili WANG. Dynamic targets searching assistance based on virtual camera priority. Virtual Reality & Intelligent Hardware, 2021, 3(6): 484—500
DOI: 10.1016/j.vrih.2021.10.001

Abstract **Background** When a user walks freely in an unknown virtual scene and searches for multiple dynamic targets, the lack of a comprehensive understanding of the environment may have a negative impact on the execution of virtual reality tasks. Previous studies can help users with auxiliary tools, such as top view maps or trails, and exploration guidance, for example, automatically generated paths according to the user location and important static spots in virtual scenes. However, in some virtual reality applications, when the scene has complex occlusions, and the user cannot obtain any real-time position information of the dynamic target, the above assistance cannot help the user complete the task more effectively. **Methods** We design a virtual camera priority-based assistance to help the user search dynamic targets efficiently. Instead of forcing users to go to destinations, we provide an optimized instant path to guide them to places where they are more likely to find dynamic targets when they ask for help. We assume that a certain number of virtual cameras are fixed in virtual scenes to obtain extra depth maps, which capture the depth information of the scene and the locations of the dynamic targets. Our method automatically analyzes the priority of these virtual cameras, chooses the destination, and generates an instant path to assist the user in finding the dynamic targets. Our method is suitable for various virtual reality applications that do not require manual supervision or input. **Results** A user study is designed to evaluate the proposed method. The results indicate that compared with three conventional navigation methods, such as the top-view method, our method can help users find dynamic targets more efficiently. The advantages include reducing the task completion time, reducing the number of resets, increasing the average distance between resets, and reducing user task load. **Conclusions** We presented a method for improving dynamic target searching efficiency in virtual scenes by virtual camera priority-based path

guidance. Compared with three conventional navigation methods, such as the top-view method, this method can help users find dynamic targets more effectively.

Keywords Searching assistance; Virtual environment; Path guidance; Redirection

1 Introduction

Efficient navigation in a virtual environment (VE) is an essential technique for several virtual reality (VR) applications. People are required to determine where they are, where everything else is, and how to get to particular targets or places in the VE. The user interface's insufficient navigation support will cause the user to become disoriented and get lost, impede, and weaken the user's performance in the subsequent search task.

To improve the navigation efficiency, researchers perform much work. Specially designed scenes integrated with some salient features[1,2] or set up landmarks[3,4] are utilized to improve the efficiency of navigation. These methods require special design and processing of virtual scenes, and cannot be utilized in pre-existing virtual or specific virtual scenes required by applications. 2D and 3D maps are also utilized to provide an overview of the entire scene and let users plan where they want to go for exploration[5,6]. Trail methods highlight the part of the environment users had visited, or draw the users' footprints to help users navigate more efficiently[7,8]. With map or trial methods, the user has to switch from the first-person perspective view to the maps or trails in head-mounted displays (HMDs), which interrupts the visual continuity of the 3D scene and increases the time cost and task load to users. The guiding tour method guides users to important locations using automatically computed paths[9,10]. This type of work only analyzes the user's current location and static scenes but is not suitable for dynamic target search tasks.

The task is to search for other users or dynamic targets in multi-user online competition systems or search skill-training systems. In these scenarios, it is usually necessary to hide the location of the dynamic targets, and users cannot directly access the location information of other players or dynamic targets. The search efficiency is low for two reasons. One is that users cannot grasp the global structure of the entire virtual scene and their own position owing to complex occlusions. The other is if the user does not know the location of the target when it is moving, the user can only hope to find a dynamic target. The maps and some additional views captured by fixed surveillance cameras can be provided to users to reference the location information of dynamic targets. However, even if the user has all this information, it is still difficult to find dynamic targets efficiently, especially when the user is not familiar with the virtual scene.

This study designs a virtual camera priority (VCP) -based assistance to help users efficiently search dynamic targets controlled by other independent individuals, but not by the local host. Instead of forcing users to go to destinations, we provide an optimized instant path to guide the user to places where they are more likely to find dynamic targets when they ask for help. A certain number of depth cameras are fixed in virtual scenes to obtain additional views that capture the depth information of the scene and the locations of the dynamic targets. Our method automatically analyzes the priority of extra views, chooses the destination, and generates an instant path to assist the user in finding the dynamic targets. The maintenance of the VCP is based on the analysis of the depth of the extra views. If the depth value varies in an additional view, the user is more likely to encounter the target near the camera. The higher the depth complexity, the more critical the view, and the higher the priority of the corresponding camera. A higher depth complexity means that the corresponding view will make more areas with occlusions visible and have more opportunities to catch dynamic targets.

Our method is suitable for online competition systems or search skill training systems by introducing

manual supervision. We design a user study, using six different strategies to evaluate our method. The results indicate that our method can help the user to find dynamic targets more efficiently than the conventional navigation and top view methods. Figure 1 illustrates two cases in which the user uses assistance based on VCP to search for moving guests in the apartment. The two cases have different target distributions, but the positions of the users in these images are similar. The rank of VCP is indicated by different colors on small triangle icons of virtual cameras in top views Figure 1a and 1c. Red is the first priority, blue is the second priority, and cameras in black are others.

Figure 1 Two pairs of top view (left of pair) and first-person perspective (right of pair). The yellow dots in (a) and (c) represent the user, and the green dots are targets. Red triangle is the first priority camera, blue triangle is the second priority camera, and cameras in black are others. The red line indicates the instant path generated to guide the user to their destination. Because the cameras' priorities are different in these two cases, the auxiliary guiding path is also different.

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

(1) A dynamic target search method based on VCP is proposed to help users search for dynamic targets efficiently in VR scenes when they cannot access the location information of targets.

(2) We introduce the concept of VCP, which represents the order of the virtual camera that needs to be considered in the dynamic target search. We also propose a method for calculating the priority of the view.

(3) We give three strategies for determining the temporary destination, and analyze the respective conditions of use.

(4) We conduct a user study to evaluate the efficiency of our VCP based searching method.

The remainder of this paper proceeds as follows. In Section 2, we discuss the relevant previous work. Subsequently, our approach is introduced in Section 3. The main user study that we conducted to evaluate the technique is described in Section 4. The study results are reported and discussed in Section 4.2. Finally, we discuss some limitations of this approach outlook on future works in Section 4.3 and give a conclusion in Section 5.

2 Related work

In this section, we briefly introduce widgets or interface-based methods, trail-based methods, and wayfinding methods for improving VE navigation efficiency. To gain a more comprehensive understanding of efficient navigation methods, we recommend readers to read the literature written by Darken et al.[11], which listed several essential technologies in VE navigation, and compared their advantages and disadvantages in detail. Our method focuses on wayfinding methods without a user-related configuration.

2.1 Widget based methods

Some previous works integrated widgets or interfaces in the field of view to help the user explore VE and

ensure immersion during navigation. For example, a virtual two-dimensional map[6] or a miniature three-dimensional world[5] was used to provide the survey knowledge and the user's position in the scene to improve navigation efficiency. However, when these methods are used in VR HMDs, it is usually necessary to switch from the first-person perspective view to the map or miniature world, and then switch back to the first-person perspective, which requires the user to put in some effort, moreover, the user cannot switch too frequently. Highlighting landmarks in the VE[3,11] can be effective in enhancing the navigation efficiency, but it is only suitable for a large-scale environment. Using translucent walls[12] in virtual scenes can help the user to remove the occlusion of walls and improve the pathfinding performance. However, this may confuse the user while reaching their destination. These methods provide intuitive interfaces and are very similar to the target searching method in the real world. They only provide the user with some information about the scene, and the user needs to decide how to search by himself based on this information. These methods do not provide guidance for reaching specific goals. Moreover, maps and landmarks are generated and placed for static scenes, which is not very helpful in finding dynamic targets.

2.2 Trail based methods

Trail-based methods provide users with trails or trails of previous users as visiting cues during navigation. To highlight which parts of the environment the user has visited, Darken et al.[7] allowed the user to specify where to place markers, or automatically drew footprints along the user's path in the VE. Instead of sampling the footprints at a fixed frequency, sampling the footprint at the right time made the footprints more helpful for navigation[8]. A set of master footprints for guidance from the footprints of multiple people was proposed[13]. Providing the footprints of previous users to later users as a reference could improve detection efficiency but would make the environment chaotic in future visits. These methods are suitable for static targets. In the case of searching for a dynamic target, because the location of the dynamic target changes over time, neither the previous user's footprint nor the current user's self-traveled footprint is of reference value for the current user.

2.3 Wayfinding methods

Wayfinding is typically defined as a cognitive aspect of navigation for planning and forming strategies. This is solved by guiding exploration technology. Usually, the guidance method includes two steps: an offline calculation step, which uses scene information as input to derive the entire tour route, and then an online interactive navigation step, which provides users with guided exploration and improved spatial perception.

To automatically identify important areas, methods using viewpoint quality have been proposed[14–17]. These methods usually estimate the amount of information about the scene obtained from that position, and assign a quality value to each position in the scene. Viewpoint quality can be computed in a variety of ways[18–21] and for different types of scene representations, such as geometry-based scenes[17,20], volume data[22], and point clouds[23]. In addition to the methods of automatically determining the viewpoint quality, there are some methods that combine some user input, such as the importance of certain scene entities, to calculate the quality of viewpoint[22,24,25]. In VR navigation, viewpoint quality has been utilized to continuously guide the user toward unseen locations of interest, and determine a set of best or representative positions as waypoints for a camera path[9,10,26,27], to identify the most relevant areas in a scene[28,29], and to automatically control the travel speed in navigation[30]. Although the route is calculated automatically, some methods require a user configuration of the scene, which may be undesirable. The

methods of automatically calculating the position of interest are used for static scenes rather than dynamic targets. Our method does not need to know the location of the target explicitly, but analyzes the movement of the target based on the content seen in the extra views; therefore, our method does not require a configuration related to the users.

There are also some other improvements related to path guidance. A river analogy was proposed, which allowed some deviation from the camera path[31,32]. A system that can perform some interactive control on the direction and speed of movement was introduced in[33], which improved the interaction when following virtual tours. A multi-perspective visualization is automatically constructed to eliminate occlusions and improve VR exploration efficiency[34]. In addition, to further improve the interactivity, some studies only provided suggestions for some essential places that the user may be interested in[35]. Because path generation is the basis of guidance, our work focuses on the path generation method for dynamic targets rather than improving user-friendly interactive guidance.

3 Methods

The VCP-based dynamic target searching method is introduced to provide users with a guide when they ask for help in cases of chasing multiple dynamic targets in the VE. The basic idea of the method is to add virtual cameras to capture the extra views, calculate the priorities of the extra views, and guide the user according to these priorities. Our search method has three main steps: (1) VCP initialization; (2) VCP updating, and (3) guidance generation. The dynamic target-searching pipeline is illustrated in Figure 2.

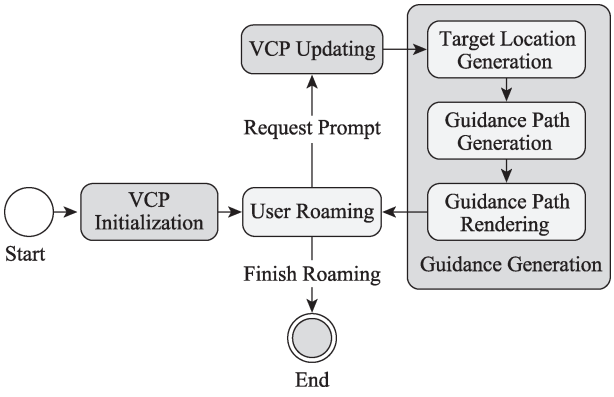


Figure 2 Dynamic targets searching pipeline.

The method's inputs are the scene, dynamic targets, user viewpoint, user orientation, and extra views. The output is the user's view under guidance. In VCP initialization, we compute the importance of the extra view captured by the virtual cameras with the empty virtual scene, that is, no dynamic target is considered. We sort the extra views according to their importance, and the indexes of the sorted extra views are used as priorities. When the user performs a searching task for dynamic targets, the priorities of the extra views are updated instantly if the user asks for guidance. The updating is based on the initial priority and the depth of the extra views with the dynamic targets in the scene. Temporal cues are also considered. Subsequently, we use several strategies to determine the destination according to the updated priorities of the extra views. The navigation path is generated using the A* method from the user's position to the destination, and shown to the user as guidance.

We provide a detailed description of each step in the following section. VCP initialization is described in Section 3.1, and VCP updating is discussed in Section 3.2. The details of the destination determination and guidance path generation are introduced in Section 3.3.

3.1 VCP initialization

For each extra view captured by the virtual camera, we initialize its VCP by analyzing the depth complexity of the view of the empty scene. The higher the depth complexity, the more critical the view.

This is because a higher depth complexity means that the corresponding view will make more areas with different occlusions visible, and have more opportunities to catch dynamic targets. The initialized VCP is used as the weight of the dynamic target footprint area during VCP updating.

To initialize VCP, we first introduce a concept called depth complexity. The depth complexity, C, of the extra view is used to describe the depth discontinuity area caused by other occlusions, which is obtained according to two variables: N_d and V_d. N_d is the number of local deep segments, and V_d is the variance of the length of the local deep segment in the extra view. Figure 3 illustrates the views with low (above row) and high (following row) depth complexities. The original views are presented in column 1, the depth buffers are visualized in column 2, and the depth of the centerline is visualized in column 3. The blue line represents the average value of the depth in the centerline used in Algorithm 1 as variable m. The local deep segments are marked in yellow, starting from the red points and ending with green points.

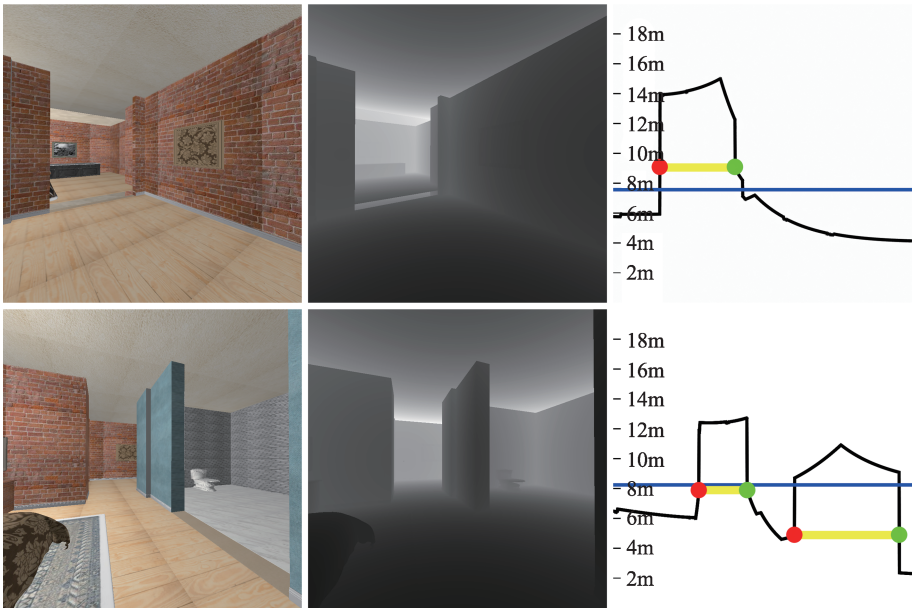


Figure 3 Comparison of the depth map captured by the virtual camera with low depth complexity (left) and the virtual camera with high depth complexity (right). Column 1 shows extra views and column 2 shows depth buffer visualization. At last, depth in the centerline is visualized in column 3. The blue line in column 3 is the average value of depth.

The inputs of VCP initialization are depth maps for all extra views without target objects (denoted by D), the viewpoint location of user (denoted by V), and the ratio of the minimum width of the local depth segment to the width of the entire depth map (denoted by w), which is set to one-tenth of the user study to ensure the lower bound of the segment's width. The outputs are the indexes of the sorted extra views (denoted by C). The initialization comprises two steps. First, we compute the depth complexity for each extra view with an empty scene. Second, we sort the extra views according to their depth complexity in descending order to generate their initial priority.

The depth complexities of all the extra views are computed using Algorithm 1. First, we created an empty set S to store all the local deep segments (line 2). For each depth map in the list, we considered its horizontal centerline depth (line 3). We stipulated the minimum width of the local depth segment (line 4). Then, we determined the left endpoints and the corresponding right endpoints (lines 5−10). To find the left endpoint, we searched from the first pixel of the centerline. If the current pixel had a local maximum gradient value, or if its subsequent w' pixels gradually increased evenly, it was used as the left endpoint of

the segment (line 6). The pixel with the minimum local gradient or with the previous w' pixel gradually decreasing evenly was regarded as the right endpoint of the segment (line 9). Before adding the local deep segment (l, r) into S, it was necessary to check whether the deep segment in the view is at the appropriate observation angle (line 11−12). If the angle between vector (V, L_l) + vector (V, L_r) and vector (L_l, L_r) in 3D space was too small, the segment (L_l, L_r) was too oblique for viewpoint V, that is, (L_l, L_r) was not under the proper view; hence, (l, r) will not be added to S. Once we obtained all local deep segments in this depth map, we recorded the number of segments and the variance of these segment lengths (lines 14−15).

Algorithm 1 Depth complexity computation

Input: depth maps list of extra views D, view point V, the ratio of the minimum width of the local depth segment to the width of the entire depth map w
Output: depth complexity list of extra views C

```
1  for id=0; id < length(D); id++ do
2      S = ∅ ;
3      L = GetCenterline(D[id]) ;
4      w' = w * length(L) ;
5      for l=0; l < length(L) - w'; l++ do
6          if L[l] suddenly increases or the w' elements following l in L are gradually increase evenly then
7              portalFlag = 0;
8              for r=l + w'; r < length(L); r++ do
9                  if L[r] suddenly decreases or the w' elements before r in L are gradually decrease evenly then
10                     portalFlag = 1; break;
11             if portalFlag == 1 and angle(V⃗L_l + V⃗L_r, L_l⃗L_r)>Π / 9 then
12                 insert (l,r) to S ;
13             l = r ;
14     Num[id] = Number(S) ;
15     Var[id] = Variance(S) ;
16 Num = Normalization(Num) ;
17 Var = Normalization(Var) ;
18 for id=0; id < length(D); id++ do
19     C[id] = Num[id] + Var[id] ;
20 return C
```

After processing all extra views' depth maps, we normalized the number and variance lists of the segments using the L_2 norm (lines 16−17). Finally, the depth complexity is calculated by adding the normalized number and normalized variance (lines 18−19), then the depth complexity list is returned (line 20). After obtaining the depth complexity of each extra view, we sort the views according to the depth complexity and initialize the VCP with the sorted view indexes. The reason for considering both number and variance is that a large number means that additional views can cover more channels, so there is a better chance to capture targets from different channels, and a large variance means that the more complex the scene, the greater the probability of targets passing through different types of regions. Both will give higher priority.

In Figure 3, the VCP of the left image is smaller than that of the right image because there is one segment in the left image and two segments in the right image.

3.2 VCP updating

When a target object moves freely in a virtual scene, the target object may appear in the field of additional view. If the user needs to be prompted, navigating to the virtual camera's location where it can capture the moving targets in its view may help the user find the target objects more efficiently. Therefore, we are required to find an extra view with a higher VCP to guide navigation in real time. When a dynamic target moves in a scene, the dynamic target's size and movement trend in the extra view will change. Therefore, we are required to consider the static scene and projection of the dynamic target in additional views. We

update the VCP when users need a prompt. Here, we do not distinguish different targets because we consider the movement trends of targets together and calculate a position with the highest probability of catching targets.

VCP updating is based on an analysis of the depth of the extra views with the target object. In this procedure, the depth map list for all extra views (initial and current), the dynamic target area ratio of the previous frame, and the initial depth complexity list for the extra views are used as inputs. The output is the updated depth complexities of the additional views.

The updating process comprises two steps. First, we computed the depth complexity for each view with the target object. Second, we sort the extra view according to the new depth complexity in descending order to update the VCP. The greater the complexity, the higher is the priority. The view-depth complexity of the extra views is updated using Algorithm 2.

Algorithm 2 View depth complexity updating

Input: current depth maps list of the extra view D,
initial depth maps list of the extra view D_{init},
dynamic target area ratio of the previous frame P_{pre},
initial depth complexity list C
Output: depth complexity list C'

1 **for** $id=0;\ id < length(D);\ id{+}{+}$ **do**
2 $P[id]$ = ComputeRatio($D[id]$, $D_{init}[id]$) ;
3 **if** $(P[id] == 0)$ **then** $C'[id] = C[id]$;
4 **else**
5 **if** $(P_{pre}[id] != 0)$ **then** $v = P[id]/P_{pre}[id]$;
6 **else** $v = 1.0$;
7 $C'[id] = C[id] * P[id] * v * CONST$
8 **return** C'

Although the static scene covered by a fixed camera does not change, the target captured by the camera changes in real time. We calculate the number of pixels whose depth was greater than the initial depth. The ratio of this number to the entire image size is the projected area of the targets in the extra view (line 2). To reduce complexity, the size of the depth map is set to 100pixels×100pixels; hence, it can satisfy the performance requirement of 60fps. If there is no target in the current extra view, the updating complexity is similar to the initialized complexity (line 3). If the target is visible in both the current frame and previous frames of the extra view, we calculate the change rate of the projected area of the dynamic target in the current frame relative to the previous frame, and record it in v (lines 5−7). An increase in the current frame ratio to the previous frame indicates that the targets are closer to the extra views, and the corresponding catch possibility increases. If the ratio is small, it means that the targets are far away from the extra views, and the corresponding catch possibility is small (line 5). If the target is solely visible in the current frame of the extra views but not visible in the previous frame, we set V to 1.0 (line 6). Finally, we update the depth complexity by multiplying the updated complexity, the dynamic target area ratio of the previous frame, and the change rate together (line 7). We adopt a constant to adjust the product's value, which ensures that the depth complexity of the extra view with the visible target is greater than the complexity when the target is invisible.

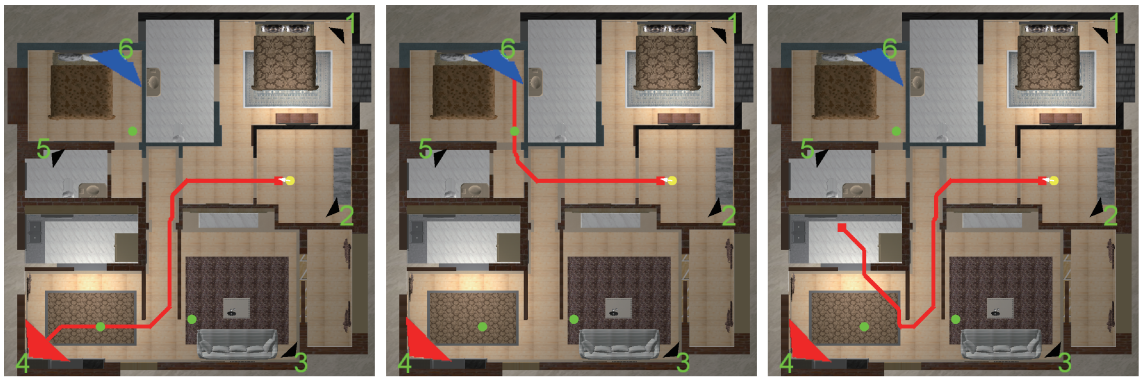
With Algorithm 2, the greater the depth complexity of the extra view with the target object, the higher the probability of encountering the target when the target guide position is set to the camera position capturing the extra view. Then, we sort this complexity and update the priority of the extra views according to their ranking in the complexity list.

3.3 Guidance generation

When the user asks for a prompt in the search navigation, a virtual path is rendered to guide the user. To generate this virtual path, the destination must be set. The setting strategy of the destination based on VCP considers the information of the scene and dynamic targets; hence, the next step is to use the shortest path as the guide path from the user to the destination, rather than the complex guide path generation strategy. Here, the A* algorithm was used to generate the guidance path from the current user location to the

destination.

We provide three strategies based on the VCP to set the destination (Figure 4).



(a) (b) (c)

Figure 4 Comparison of three strategies used in our method by the top view. There are three dynamic targets in this case, and the user and dynamic targets are at the same position in these three images. (a), (b) and (c) are corresponding to strategies 1, 2 and 3, respectively.

Strategy 1: We set the position of the virtual camera that captured the highest priority view as the target position. Intuitively, we usually think that the target object is most likely to be captured near the camera position corresponding to the highest priority view. Therefore, we guide the user to the location of the camera with the highest VCP in Strategy 1.

Strategy 2: We set the position of the virtual camera that captured the second-highest priority view as the target position, and explored whether the position of the camera with the second-highest priority is better than the highest-priority position. This is because, on the one hand, the extra view at the highest priority position cannot always maintain its condition, and the second highest priority position is likely to become the highest priority position next. On the other hand, sometimes the position of the camera with the second highest priority is not far away from the position of the camera with the highest priority value.

Strategy 3: We set the midpoint of the highest priority camera and the second highest priority camera as the target position. This strategy is not required to indicate the exact camera position to navigate, but provides a direction with a high probability of finding the target object. Owing to its adaptability, we want to explore whether participants are more likely to find moving targets following this navigation prompt. This destination setting method is a compromise between strategies 1 and 2.

4 User study

Six conditions are designed for each scene, namely the control conditions (CC_1, CC_2, *and* CC_3) and the experimental conditions (EC_1, EC_2, EC_3). These conditions use different guidance strategies.

CC_1: Conventional space roaming without guidance, the user moves around to find dynamic targets in virtual scenes.

CC_2: Conventional space roaming with a mini-map, which includes the top view of the VE. The position and direction of the user is marked on the map. The map is a translucent interface (with an alpha value of 0.6, Figure 5a), which is a north-up configuration, where the map is displayed and closed at the press of a button.

CC_3: Conventional space roaming with a north-up mini-map. Moreover, a certain number of views captured by additional virtual cameras are displayed in the first-person view translucently (with an alpha value of 0.6; Figure 5b).

(a) (b) (c) (d)

Figure 5 First-person view in CC_2 (a) and CC_3 (c) when the user asks for prompt. In addition to the VE layout, the top view in CC_2 solely has the position and direction of the user. A yellow point marks the position and a white arrow marks the orientation, see (b), the partial enlargement of (a). In CC_3, the position and orientation of all extra virtual cameras are marked in the mini-map besides all the information in CC_2's mini-map, and the views of all cameras are attached under the top view. The cameras' icons in CC_3's mini-map are in black, see (d), the partial enlargement of (c).

EC_1: Select the highest priority camera's viewpoint as the destination (Strategy 1).

EC_2: Select the second-priority camera's viewpoint as the destination (Strategy 2).

EC_3: Select the midpoint of the highest priority and the second priority camera's viewpoint as the destination (Strategy 3).

The user roams in the scene and looks for moving targets. When the user needs a prompt, the control conditions provide a top view (CC_2) or extra virtual cameras' views (CC_3). The mini-map in CC_3 not only has the top view of VE (like CC_2) but also has the extra virtual cameras' position and orientation marked on the map (Figure 4). The experimental conditions select different navigation destinations according to different strategies and provide route prompts. Users can freely choose whether to follow a route.

4.1 Design

4.1.1 Participants

We recruited 16 participants (2 females, 19−28 years old, mean=23.9, SD=2.1). Seven of our participants have experiences using HMD VR applications. The participants had normal and corrected visions, and none reported preexisting balance disorders. Our study adopted a within-subject design, with each participant serving once in each of the six conditions of the two scenes. The order of the conditions of the participants was random in two scenarios. If an experiment fails, the experiment was repeated. This means that there were 16 pieces of data for each condition of each scene.

4.1.2 Hardware and software implementation

We utilized an HTC Cosmos VR system with two hand-held controllers, which allowed the user to ask for a prompt. Each HMD was connected to its own workstation with a 3.6GHz Intel(R) Core (TM) i7-9900KF CPU, 32GB of RAM, and an NVIDIA GeForce GTX 1060 graphics card. The VE was rendered at 60fps for each eye. The tracked physical space for the user was a 4m×4m empty space in our laboratory. Whenever a participant reached the boundary of the tracked area, a yellow grid was rendered in the HMD. The user could use the controller to trigger a reset operation that rotated the VE by 180°, then redirect and return to his or her previous orientation.

4.1.3 Virtual environments and scenarios

Two scenes were used in this experiment, which were designed to explore the influence of different space sizes on the method. Each user performed each scene under all the conditions.

Scene 1. The first scene was to meet the walking guests in an indoor apartment. The apartment covers an area of 30m×30m (Figure 1). There were ten dynamic guests roaming in the apartment. The user was asked to catch the guests as much as possible after a 60s countdown. The guests' speed was 0.4m/s. The guests' moving path was randomly generated in the available area, ensuring that it is connected to the user's position. When the participant is looking for a moving guest, if the distance to the moving guest is less than 0.6m, it is considered that the current guest is found, and the guest will disappear from the virtual scene. The participant will continue to search for the remaining moving guests until all of them are found. Six extra virtual cameras are fixed in this scene, and their distribution is illustrated in Figure 4.

Scene 2. In the second scene, the user is required to search for moving robots in a warehouse. This scene has an area of 15m×25m (Figure 6). The robot's movement is also randomly generated, and its rules of action are similar to those in *Scene 1*. Considering that the size of *Scene 2* is half of that in *Scene 1*, we added additional three virtual cameras in *Scene 2*.

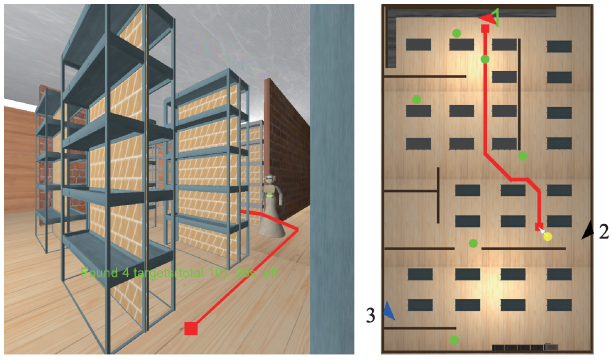


Figure 6 User study in *Scene 2*. There are ten robots in *Scene 2* initially, the user found 4 targets already.

4.1.4 Procedure

Our study uses a within-subject design, with each participant serving in all conditions for all the scenes. The conditions and trial order are randomized. The scene for each condition is completed within one minute. Before each scene, participants are given a three-minute tutorial to understand the task and practice catching the target. During the task, the system displays the current number of caught targets and the remaining time for participants.

4.2 Results

The testing field is illustrated in Figure 7, where the physical space is 4m×4m. Task performance is measured using the following four metrics for each scene and for each condition:

(1) Target caught (*TAR*): the number of target users catch during the experiment;

(2) Distance per target (*DPT*): average distance between catching each target;

(3) Reset per target (*RPT*): average number of reset users take between catching each target;

(4) Distance per reset (*DPR*): average distance users walk between each reset.

For the first two indicators, the larger the value, the more effective our method. For the latter two indicators, the smaller the value, the more effective our method. The

Figure 7 Testing field.

best results are indicated in bold for the six conditions of the four indicators. Refer to Tables 1, 2 and 3 for further details.

Table 1 Number of targets found and average distance users walk per target

Scene	Condition	TAR_{AVG}± TAR_{SD}	$(EC_1\text{-}CC_i (EC_i))/CC_i (EC_i)$	p	Cohen's d	Effect size	DPT_{AVG}± DPT_{SD}	$(CC_i(EC_i)\text{-}EC_1)/CC_i (EC_i)$	p	Cohen's d	Effect size
Scene 1	EC_1	**7.75±0.83**	–	–	–	–	4.27±1.09	–	–	–	–
	EC_2	7.38±0.86	−5.1%	0.43	0.44	S	**4.23±0.94**	−0.8%	0.80	0.03	VS
	EC_3	7.13±1.05	8.8%	0.40	0.66	M	4.30±1.35	0.8%	0.82	0.04	VS
	CC_1	5.25±0.64	47.6%	<0.01*	3.38	H	5.60±1.15	23.8%	<0.01*	1.54	VL
	CC_2	5.75±0.91	34.8%	<0.01*	2.30	H	4.98±0.97	14.3%	0.03*	0.69	M
	CC_3	5.75±1.74	34.8%	<0.01*	1.47	VL	5.06±1.70	15.7%	0.16	0.56	M
Scene 2	EC_1	**8.38±0.98**	–	–	–	–	**3.64±0.71**	–	–	–	–
	EC_2	8.13±0.86	3.1%	0.45	0.36	S	3.73±0.76	2.4%	0.75	0.12	VS
	EC_3	8.00±1.32	4.7%	0.57	0.38	S	3.81±1.40	4.3%	0.69	0.15	VS
	CC_1	5.38±0.70	55.8%	<0.01*	5.00	H	4.87±1.45	25.2%	0.01*	1.08	L
	CC_2	6.00±1.32	39.6%	<0.01*	2.38	H	4.56±1.75	20.2%	0.02*	0.69	M
	CC_3	5.88±1.54	42.6%	<0.01*	2.19	H	5.03±2.01	27.6%	0.06	0.92	L

Table 2 average number of reset per target and average distance users walk per reset

Scene	Condition	RPT_{AVG}± RPT_{SD}	$(CC_i(EC_i)\text{-}EC_1)/CC_i (EC_i)$	p	Cohen's d	Effect size	DPR_{AVG}± DPR_{SD}	$(EC_1\text{-}CC_i (EC_i))/CC_i (EC_i)$	p	Cohen's d	Effect size
Scene 1	EC_1	**1.04±0.28**	–	–	–	–	4.16±0.62	–	–	–	–
	EC_2	1.05±0.18	1.3%	0.88	0.06	VS	**4.19±0.71**	−0.7%	0.96	0.05	VS
	EC_3	1.05±0.27	1.4%	0.78	0.07	VS	4.07±0.92	2.1%	0.71	0.12	S
	CC_1	1.52±0.46	31.6%	<0.01*	1.25	VL	3.92±0.98	6.1%	0.45	0.35	S
	CC_2	1.30±0.22	20.3%	0.01*	1.23	VL	3.84±0.79	8.1%	0.12	0.56	M
	CC_3	1.23±0.35	15.6%	0.04*	0.60	M	4.08±1.02	1.9%	0.52	0.09	VS
Scene 2	EC_1	**0.94±0.14**	–	–	–	–	3.87±0.62	–	–	–	–
	EC_2	0.98±0.20	4.3%	0.42	0.25	S	3.83±0.49	1.2%	0.56	0.12	VS
	EC_3	0.96±0.25	2.2%	0.62	0.11	VS	**3.96±0.76**	−2.3%	0.88	0.10	VS
	CC_1	1.38±0.28	32.3%	<0.01*	2.76	H	3.46±0.66	11.9%	0.35	0.67	M
	CC_2	1.25±0.12	24.9%	<0.01*	2.39	H	3.58±0.91	8.0%	0.72	0.34	S
	CC_3	1.39±0.50	32.8%	<0.01*	1.05	L	3.71±0.61	4.4%	0.57	0.31	S

Table 1 reports the average and standard deviations of the first two metrics *TAR* and *DPT*. Because there are ten dynamic targets generated initially in VE in every scene, the *TAR* for each user is an integer between 0 and 10. Table 2 reports the two metrics, *RPT*, and *DPR*. Both tables presents results for each scene and condition.

The *TAR* in EC_1 was compared to the other five conditions. Table 1 reports the relative difference in the average number of *TAR* values (column 4). A Shapiro-Wilk test[36] revealed that the *TAR* metric is not normally distributed; hence, we performed a statistical analysis of the differences between conditions using the Wilcoxon signed-rank test (see p values in column 5). We also reported Cohen's d (column 6) along with the conventional effect size quantification (column 7)[37,38] . The d values were translated to qualitative effect size estimates of 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). They are abbreviated as H, VL, L, M, S, and VS in the

Table 3 Raw Task Load Index score and Simulator Sickness Questionnaire score

Scene	Condition	TLX_{AVG}± TLX_{SD}	$(CC_i(EC_i)\text{-}EC_1)/CC_i(EC_i)$	p	Cohen's d	Effect size	PRE_{AVG} ± PRE_{SD}	$POST_{AVG}$ ± $POST_{SD}$	p
Scene 1	EC_1	**28.02±11.97**	–	–	–	–	**3.06±3.22**	**3.70±3.86**	0.21
	EC_2	29.13±13.64	3.8%	0.78	0.09	VS	3.28±3.48	3.96±4.04	0.35
	EC_3	30.75±13.34	8.9%	0.47	0.22	S	3.10±3.82	4.12±4.82	0.51
	CC_1	29.50±16.93	5.0%	0.75	0.10	VS	3.52±4.34	5.48±5.34	0.36
	CC_2	34.96±13.54	19.8%	0.12	0.54	M	3.26±3.86	4.18±3.78	0.26
	CC_3	41.23±14.50	32.0%	0.02*	1.07	L	4.18±4.78	7.18±6.24	0.71
Scene 2	EC_1	**26.65±10.21**	–	–	–	–	2.88±3.58	**3.66±4.18**	0.26
	EC_2	28.00±9.65	4.8%	0.65	0.10	VS	**2.70±3.36**	3.84±4.74	0.31
	EC_3	29.88±15.56	10.8%	0.27	0.33	S	3.04±3.06	4.30±5.22	0.35
	CC_1	27.88±14.36	4.4%	0.42	0.09	VS	3.18±4.18	4.86±4.26	0.42
	CC_2	30.46±13.37	12.5%	0.37	0.32	S	2.50±3.06	4.10±3.98	0.38
	CC_3	39.23±15.86	32.1%	0.01*	0.94	L	3.64±3.88	6.20±5.76	0.52

tables.

For both scenes, data show that the EC_1 has the highest average *TAR* in all conditions. The user could choose to walk along the guiding path, which had the highest probability of catching the targets. Therefore, the user caught more targets in EC_1 on average, and the advantage of EC_1 over three *CC*s on this metric was statistically significant ($p<0.05$). The condition with the lowest *TAR* was CC_1. This is because the user could not receive any assistance in this condition and would walk to useless places repeatedly. CC_3 had a higher standard deviation of the *TAR*. The user needed to analyze the virtual camera's view to recognize the number of targets, switch from the user's egocentric perspective to the exocentric perspective provided by the map, and plan a path guiding himself to the camera's position. This often requires a mental load. Very few users could benefit from the aid of the camera view. The rest of the users abandoned this analysis and only used the top view in CC_3, and this assistance degenerated to CC_2. The *TAR* in EC_1 was slightly higher than that in EC_2 and EC_3. There was no significant difference between the two conditions. In EC_3, if the first priority camera was far away from the second priority camera, the midpoint strategy may set the destination in a useless place. Therefore, the *TAR* in EC_3 was slightly less than that in the other two experimental conditions. The result of *Scene 2* is similar to that of *Scene 1* on this metric.

Participants committed more moving distance in EC_1, thereby indicating that they moved faster in EC_1 than in *CC*s. We use a relative proportion to further analyze the relationship between the distance the users walked, number of targets the users caught, and reset number the users taken in these six conditions.

We averaged the distance users walked and the number of reset users taken during the whole task to each target, denoted by *DPT* (Table 1) and RPT (Table 2), respectively. The *DPT* in the three *EC*s is smaller than that in the three *CC*s in both scenarios, which means that our method requires less walking to catch a target. The average decrease in *DPT* in EC_1 is 18% in *Scene 1* and 24% in *Scene 2* compared with three *CC*s. There is a statistically significant ($p<0.05$) difference between EC_1 and *CC*s except CC_3 on this metric. If the user has good spatial cognition, he or she can benefit from CC_3. The value of *DPT* also shows advantages in *Scene 2*, and the extent of its advantages is deeper than that of *Scene 1*. Our method can provide a more timely and accurate priority for extra views and help the user perform better. The extent of the *EC*s' advantages over *CC*s in *Scene 2* is greater than that in *Scene 1*.

In both scenes, *RPT* had a statistically significant ($p<0.05$) difference between EC_1 and *CC*s. The extent of the effect size on *RPT* between EC_1 and *CC*s was greater than that on *DPT* in both tasks. One reason is

that there are four participants in these experiments who are senior game players, and are familiar with the operation of VR devices. They used reset as a tool to assist in exploring the VE more effectively. For example, if they notice that there is no target in front of them, they will implement a reset to rotate 180° immediately. They prefer to use this convenient interface instead of rotating their bodies artificially. Another reason is that when they encounter a target at their side back, they will use a reset to pick up the rotation speed as they believe that this can increase the capture efficiency. In contrast, our method in EC_1 provides a guiding path, and these four participants tend to walk along this path safely instead of considering whether to use reset as a tool to enhance their exploration efficiency. Eventually, this causes a few increments of *DPR* (Table 2) in EC_1 compared with *CCs*, but is not a statistically significant difference.

We measured the task load using Raw TLX[39,40] (Table 3, middle five columns). We averaged the scores of the six raw TLX task load questions, which were between 5 and 100, with 5-point increments. The task load values did not follow a normal distribution; hence, the differences were analyzed using the Wilcoxon signed-rank test. The task load of EC_1 was not significantly different from that of CC_1 and CC_2. However, the task load was significantly higher for CC_3 than for EC_1. We attribute this expected difference to the additional increment of the user's cognitive load required by CC_3, compared with EC_1. This is because the user needs to recognize the target's figure in the virtual camera's view and then plan a path to the corresponding camera's position. The load of EC_3 is larger than that of EC_1 and EC_2, and the task load of EC_2 was similar to that of EC_1. This is because there is sometimes no target at the midpoint of the first-and second-priority cameras. If the user goes to that point and sees nothing, the time is wasted, and they may feel frustrated and flustered. The task load of EC_1 is similar to that of CC_1, indicating that the EC_1 strategy will not bring much more load to users compared with the conventional method. In contrast, the EC_1 strategy will probably increase the user's confidence.

Simulator sickness was measured using the standard simulator sickness questionnaire (Table 3, rightmost three columns). The SSQ was administered pre-and post-experiment for each task and condition. We then analyzed these questionnaires using the total severity (TS) score[41]. All conditions exhibited pre-to post-TS increases below the 70 thresholds, which is typically used to detect onset simulator sickness[42]. Furthermore, the participants were instructed to stop the experiment at any time if they feel uncomfortable, but all participants completed their experiments. The SSQ scores were not normally distributed, and the pre-to post-differences were analyzed using the Wilcoxon signed-rank test. None of the differences were statistically significant. We conclude that in these experiments, simulator sickness is not more of a concern for our method than it is for conventional VR exploration. This is expected because our searching assistance is also an immersive first-person exploration. The auxiliary guiding path will not interfere with users significantly and is then viewed conventionally.

4.3 Limitations

One limitation of our method is that the virtual camera used to obtain the extra view containing the scene and dynamic target information is manually fixed in the virtual scene based on experience, which may not be the best practice. In future studies, we will design automatic camera layout algorithms based on the analysis of different scene structures.

Another limitation of our method is that it is based on an extra view centerline depth analysis to detect deeper segments, but if the scene has several small geometric details, the depth discontinuity will reduce the detection robustness. In the future, for complex scenes, we should integrate smart-filtering algorithms to detect segments and enhance the robustness of detection.

Finally, we utilized a straightforward strategy to set the destination of the guide path. However, the actual situation is that the positions of the virtual camera and the dynamic target are not similar, and inaccurate target positions may affect the search efficiency. A more accurate dynamic target position prediction method or a more reasonable destination-setting strategy should be considered.

5 Conclusion

We presented a method for improving the dynamic target searching efficiency in a VE using VCP-based path guidance. In essence, when a user asked for help, our method provided the user with an optimized instant path, making them more likely to find dynamic targets. Our method automatically analyzed the extra views containing static scenes and dynamic target information captured by a certain number of depth cameras fixed in the virtual scene, calculated the priority of these extra views, selected the destination, and generated an instant path to help the user find dynamic targets. We designed a user study to evaluate the proposed method. The results indicate that compared with the conventional navigation and the top view methods, our method helped the user find dynamic targets more effectively, reduced task completion time, reduced the number of resets, increased the distance between two resets, and reduced user task load. The benefits of our approach did not come at the cost of increasing simulator sickness. Our method was suitable for various VR dynamic target search applications that did not require complicated manual supervision or input. To further expand the application scenarios of our VCP-based dynamic target efficient search method, we will migrate the method from VR application scenarios to augmented reality application scenarios and make the system more practical with the help of computer vision technology in the future.

Declaration of competing interest

We declare that we have no conflict of interest.

References

1 Kruijff E. 3D user interfaces: theory and practice. IEEE Transactions on Visualization and Computer Graphics, 2012, 18 (4): 565−572

2 Lynch K. Reconsidering the image of the City. In: Cities of the Mind. Boston, MA, Springer US, 1984, 151−161 DOI:10.1007/978-1-4757-9697-1_9

3 Chen C H, Chen M X. Wayfinding in virtual environments with landmarks on overview maps. Interacting with Computers, 2020, 32(3): 316−329 DOI:10.1093/iwc/iwaa022

4 Keil J, Edler D, Kuchinke L, Dickmann F. Effects of visual map complexity on the attentional processing of landmarks. PLoS One, 2020, 15(3): e0229575 DOI:10.1371/journal.pone.0229575

5 Danyluk K, Ens B, Jenny B, Willett W. A design space exploration of worlds in miniature. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. Yokohama Japan, New York, NY, USA, ACM, 2021, 1−15 DOI:10.1145/3411764.3445098

6 Zhao H T, Thrash T, Grossrieder A, Kapadia M, Moussaïd M, Hölscher C, Schinazi V R. The interaction between map complexity and crowd movement on navigation decisions in virtual reality. Royal Society Open Science, 2020, 7(3): 191523 DOI:10.1098/rsos.191523

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. 1993, 157−165 DOI:10.1145/168642.168658

8 Grammenos D, Filou M, Papadakos P, Stephanidis C. Virtual prints: leaving trails in virtual environments. EGVE, 2002,

2, 131−138

9 Freitag S, Weyers B, Kuhlen T W. Interactive exploration assistance for immersive virtual environments based on object visibility and viewpoint quality. In: 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). Tuebingen/Reutlingen, Germany, IEEE, 2018, 355−362
DOI:10.1109/vr.2018.8447553

10 Wu F, Thomas J, Chinnola S, Rosenberg E S. Exploring communication modalities to support collaborative guidance in virtual reality. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW). Atlanta, GA, USA, IEEE, 2020, 79−86
DOI:10.1109/vrw50115.2020.00021

11 Darken R P, Peterson B. Spatial orientation, wayfinding, and representation. Handbook of virtual environments. CRC Press, 2002, 533−558

12 Sun J, Stuerzlinger W. Selecting and sliding hidden objects in 3D desktop environments. Graphics Interface, 2019, 8, 1−8

13 Ruddle R A. The effect of trails on first-time and subsequent navigation in a virtual environment. In: IEEE Proceedings. VR 2005. Bonn, Germany, IEEE, 2005, 115−122
DOI:10.1109/vr.2005.1492761

14 Freitag S, Löbbert C, Weyers B, Kuhlen T W. Approximating optimal sets of views in virtual scenes. In: 2017 IEEE Virtual Reality (VR). Los Angeles, CA, USA, IEEE, 2017, 311−312
DOI:10.1109/vr.2017.7892301

15 Freitag S, Weyers B, Kuhlen T W. Efficient approximate computation of scene visibility based on navigation meshes and applications for navigation and scene analysis. In: 2017 IEEE Symposium on 3D User Interfaces (3DUI). Los Angeles, CA, USA, IEEE, 2017, 134−143
DOI:10.1109/3dui.2017.7893330

16 Neuville R, Pouliot J, Poux F, Billen R. 3D viewpoint management and navigation in urban planning: application to the exploratory phase. Remote Sensing, 2019, 11(3), 236
DOI:10.3390/rs11030236

17 Zhang Y, Fei G Z, Yang G. 3D viewpoint estimation based on aesthetics. IEEE Access, 2020, 8, 108602−108621
DOI:10.1109/access.2020.3001230

18 Dutagaci H, Cheung C P, Godil A. A benchmark for best view selection of 3D objects. In: Proceedings of the ACM Workshop on 3D Object Retrieva. Firenze, Italy, New York, ACM Press, 2010, 45−50
DOI:10.1145/1877808.1877819

19 Freitag S, Weyers B, Bönsch A, Kuhlen T W. Comparison and evaluation of viewpoint quality estimation algorithms for immersive virtual environments. ICAT-EGVE, 2015, 15, 53−60

20 Neumann L, Sbert M, Gooch B, Purgathofer W. Viewpoint quality: Measures and applications. In: Proceedings of the 1st Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging. Aire-la-Vile: The Eurographics Association Press, 2005, 185−192

21 Polonsky O, Patané G, Biasotti S, Gotsman C, Spagnuolo M. What's in an image? The Visual Computer, 2005, 21(8/9/10): 840−847
DOI:10.1007/s00371-005-0326-y

22 Viola I, Feixas M, Sbert M, Groller M E. Importance-driven focus of attention. IEEE Transactions on Visualization and Computer Graphics, 2006, 12(5): 933−940
DOI:10.1109/tvcg.2006.152

23 Habibi Z, Caron G, Mouaddib E M. 3D model automatic exploration: smooth and intelligent virtual camera control. Computer Vision-ACCV 2014 Workshops, 2015, 612−626
DOI:10.1007/978-3-319-16631-5_45

24 Sokolov D, Plemenos D. High level methods for scene exploration. Journal of Virtual Reality and Broadcasting, 2007, 3(12)
DOI: 10.20385/1860-2037/3.2006.12

25 Sokolov D, Plemenos D. Virtual world explorations by using topological and semantic knowledge. The Visual Computer, 2008, 24(3): 173−185

DOI:10.1007/s00371-007-0182-z

26 Tao J, Ma J, Wang C L, Shene C K. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. IEEE Transactions on Visualization and Computer Graphics, 2013, 19(3): 393−406
DOI:10.1109/tvcg.2012.143

27 Zhao J, Song H C, Xu J B, Wu L D. Automatic exploration path planning for 3D object observation. In: 2009 Fifth International Joint Conference on INC, IMS and IDC. Seoul, Korea (South), IEEE, 2009, 1289−1294
DOI:10.1109/ncm.2009.109

28 Yang S, Xu J, Chen K, Fu H B. View suggestion for interactive segmentation of indoor scenes. Computational Visual Media, 2017, 3(2): 131−146
DOI:10.1007/s41095-017-0078-4

29 Zhang C, Zeng W, Liu L G. UrbanVR: an immersive analytics system for context-aware urban design. Computers & Graphics, 2021, 99, 128−138
DOI: 10.1016/j.cag.2021.07.006

30 Freitag S, Weyers B, Kuhlen T W. Automatic speed adjustment for travel through immersive virtual environments based on viewpoint quality. In: 2016 IEEE Symposium on 3D User Interfaces (3DUI). Greenville, SC, USA, IEEE, 2016, 67−70
DOI:10.1109/3dui.2016.7460033

31 Elmqvist N, Tudoreanu M E, Tsigas P. Evaluating motion constraints for 3D wayfinding in immersive and desktop virtual environments. In: Proceeding of the Twenty-sixth Annual CHI Conference on Human Factors in Computing systems. Florence, Italy, New York, ACM Press, 2008, 1769−1778
DOI:10.1145/1357054.1357330

32 Liu W Y, D'Oliveira R L, Beaudouin-Lafon M, Rioul O. BIGnav: Bayesian information gain for guiding multiscale navigation. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Denver Colorado USA, New York, NY, USA, ACM, 2017, 5869−5880
DOI:10.1145/3025453.3025524

33 Mirhosseini S, Gutenko I, Ojal S, Marino J, Kaufman A E. Automatic speed and direction control along constrained navigation paths. In: 2017 IEEE Virtual Reality (VR). Los Angeles, CA, USA, IEEE, 2017, 29−36
DOI:10.1109/vr.2017.7892228

34 Wang L L, Wu J, Yang X F, Popescu V. VR exploration assistance through automatic occlusion removal. IEEE Transactions on Visualization and Computer Graphics, 2019, 25(5): 2083−2092
DOI:10.1109/tvcg.2019.2898782

35 Zhang Y R, Ladevèze N, Fleury C, Bourdot P. Switch techniques to recover spatial consistency between virtual and real world for navigation with teleportation. Virtual Reality and Augmented Reality, 2019, 3−23
DOI:10.1007/978-3-030-31908-3_1

36 Razali N M, Wah Y B. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. Journal of Statistical Modeling and Analytics, 2011, 2(1): 21−33

37 Cohen J. Statistical power analysis for the behavioral sciences. Academic Press, 2013

38 Sawilowsky S S. New effect size rules of thumb. Journal of Modern Applied Statistical Methods, 2009, 8(2): 597−599
DOI:10.22237/jmasm/1257035100

39 Hart S G. Nasa-task load index (NASA-TLX); 20 years later. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 2006, 50(9): 904−908
DOI:10.1177/154193120605000909

40 Hart S G, Staveland L E. Development of NASA-TLX (task load index): results of empirical and theoretical research. Advances in Psychology, 1988, 52: 139−183
DOI:10.1016/s0166-4115(08)62386-9

41 Kennedy R S, Lane N E, Berbaum K S, Lilienthal M G. Simulator sickness questionnaire: an enhanced method for quantifying simulator sickness. The International Journal of Aviation Psychology, 1993, 3(3): 203−220
DOI:10.1207/s15327108ijap0303_3

42 Ehrlich J A, Kolasinski E M. A comparison of sickness symptoms between dropout and finishing participants in virtual environment studies. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 1998, 42(21): 1466−1470
DOI:10.1177/154193129804202101