Volume 0 (1981), Number 0 pp. 1–11 COMPUTER GRAPHICS *forum*

Automatic Image Checkpoint Selection for Guider-Follower Pedestrian Navigation

K. C. Kwan[1,2] and H. Fu[2†]

[1]University of Konstanz, Germany, [2]City University of Hong Kong, Hong Kong

Abstract

In recent years guider-follower approaches show a promising solution to the challenging problem of last-mile or indoor pedestrian navigation without micro-maps or indoor floor plans for path planning. However, the success of such guider-follower approaches is highly dependent on a set of manually and carefully chosen image or video checkpoints. This selection process is tedious and error-prone. To address this issue, we first conduct a pilot study to understand how users as guiders select critical checkpoints from a video recorded while walking along a route, leading to a set of criteria for automatic checkpoint selection. By using these criteria, including visibility, stairs, and clearness, we then implement this automation process. The key behind our technique is a lightweight, effective algorithm using left-hand-side and right-hand-side objects for path occlusion detection, which benefits both automatic checkpoint selection and occlusion-aware path annotation on selected image checkpoints. Our experimental results show that our automatic checkpoint selection method works well in different navigation scenarios. The quality of automatically selected checkpoints is comparable to that of manually selected ones and higher than that of checkpoints by alternative automatic methods.

CCS Concepts

• ***Computing methodologies*** → ***Video summarization;***

1. Introduction

Pedestrian navigation is vital in our daily life, especially when we visit unfamiliar sites. Outdoor pedestrian navigation is more or less solved since digital maps are often available. However, last-mile (e.g., outdoor to indoor) or indoor pedestrian navigation is still challenging mainly due to the lack of micro-maps for path planning.

To address this issue, a possible solution is guider-follower navigation, for which multiple systems have been proposed [KO10, MS07, SSHC15, YWY*16, WSS*16, RPZ*17, WGL*18]. In such systems, a guider first walks a path and records it for followers to follow. A straightforward way is to rely on automatic positioning and then display the sequence of positions on a map for navigation. However, automatic positioning is not always reliable, especially for indoor scenarios. Existing works also show a positive impact of using image- or video-checkpoints on navigation [Kol04, BS06, HVC*08, MHB11, WARG13, WBRM14, RPZ*17]. However, the guider needs to annotate such visual checkpoints manually. Manual selection of checkpoints is time-consuming: the guide needs to walk along the entire path or watch the entire recorded video (which can be several minutes or even hours long for navigation), possibly multiple times for selecting checkpoints. Thus, an automatic checkpoint selection system is preferred. Recently, Roy et al. [RPZ*17] studied a set of criteria on how to select the checkpoints manually. It motivates us to implement an automatic system using such criteria.

In this paper, we study the problem of automatic image checkpoint selection from a single video recorded while a guider is walking along a route, i.e., walking video (Figure 1). This system can help the guider automatically select checkpoints for the existing guider-and-follower systems [RPZ*17]. On the other hand, it also allows the guider to create easy-to-access and printable guidance information using an ordinary smartphone. Followers can use the generated guidance information without specific display devices (e.g., smartglasses). This is particularly beneficial when the guidance needs to be provided to multiple followers, such as attendees for an event, since there is no guarantee that every follower has the required equipment.

To some extent, our goal is analogous to the works of keyframe extraction from videos [DKD98, CSJ15, ZCSG16, KVGUH18, DM18, CZDZ18, MJC95, Sha95, HK17]. A key difference is that we need to extract the "keyframes" based on the path of walking, instead of video content. To understand how users as guiders man-

† Corresponding author.

Figure 1: *Given a video recorded by a guider while walking a path, our algorithm leverages multiple user-elicited criteria (including visibility, path occlusion, distance) to select a compact set of critical image checkpoints automatically. The selected checkpoints with occlusion-aware path annotation can be used to guide followers to navigate the same path.*

ually select critical checkpoints from videos, we study the criteria provided by Roy et al. [RPZ*17]. However, their criteria are somewhat subjective and not well-defined for a computational method. We thus conduct a small-scale pilot study to elicit additional criteria for checkpoint selection and the reasoning behinds from users, and finally refine a set of computational criteria.

We observe that a walking path's visibility is one of the most critical criteria to automate the checkpoint selection process. To this end, we first estimate the walking path by using existing simultaneous localization and mapping (SLAM) methods [LM13, MUS15, YYR17, QLS18]. We then detect path occlusions by a novel, lightweight algorithm. The key idea of our algorithm is to determine left-hand-side (LHS) objects and right-hand-side (RHS) objects concerning the path in the images. We believe that this path occlusion detection idea can provide an exciting insight as a new feature of geometry clue for the future works. The path occlusion also helps generate occlusion-aware path annotation (e.g., arrows), which can greatly improve the fidelity of the path information inside the checkpoint images. Our selected image checkpoints with path annotation can be presented in a printed format or used in an interactive guider-and-follower navigation system [RPZ*17].

For evaluation, we have asked users as guiders to interactively confirm our automatically selected checkpoints. Our experimental results show that our method works well in different navigation scenarios. The quality of automatically selected checkpoints is comparable to that of manually selected ones, and better than that of alternative automatic solutions including video summarization [YL95].

Here, we summarize our contributions in this paper: 1) Refining a set of checkpoint selection criteria from HCI and implementing them for a CG problem; 2) The idea of using LHS and RHS objects in a walking video for a lightweight and effective path occlusion detection algorithm; 3) The first algorithm for automatic image checkpoint selection for guider-follower pedestrian navigation.

After reviewing the literature in Section 2, we describe our pilot study for our refined checkpoint selection criteria in Section 3 and our checkpoint section system in Section 4. Section 4.2 is one of our major contributions, the path occlusion detection algorithm. Finally, we present the results in Section 5. Figure 2 shows the flow of our system and their corresponding sections in this paper.

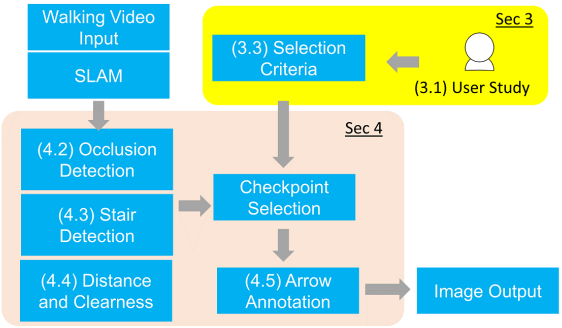


Figure 2: *Our system and their corresponding sections (Numbers in the blanket).*

2. Related Work

Our work is related to three different areas, including pedestrian navigation, keyframe extraction, and occlusion detection.

Pedestrian Navigation. The pedestrian navigation problems have been extensively studied in the research communities. A full review of this topic is beyond the scope of this paper. Please refer to insightful surveys [FABF13, BGVGT*17, DP17] for more details. Many navigation systems use image checkpoints for navigation [BS06, HVC*08, HGL*09, CW10] and many of them require reasonably accurate localization and the availability of complete digital maps. However, while multiple research communities have extensively studied indoor positioning [XYZ*15, YWL12], effective indoor navigation is still mostly unavailable in practice. We are interested in more challenging pedestrian navigation scenarios, for both indoor and outdoor, without accurate positioning or digital maps.

Guider-follower or leader-follower approaches aim to address the navigation problem with insufficient or no map information. Kameda and Ohta [KO10] allow guiders to use head-mounted cameras to capture walking videos, which are then used by followers to detect their relative position for navigation through image-based retrieval. The follow-up works propose to either use additional information (e.g., IMU data [ZSL*17, WGL*18], GPS data [WGL*18], WiFi-fingerprints [YWY*16], location-specific features like going

upstairs/downstairs [SSHC15]), or explore different methods for followers to interact with the recorded data (e.g., a scrolling method on smartwatches [WSS*16], interactive videos with fast video transitions between image/video checkpoints [RPZ*17]).

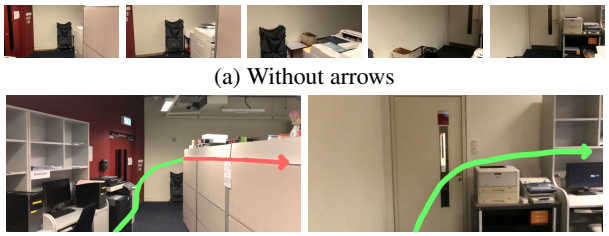
While image/video checkpoints or landmark images play an important role in many of the above works, guiders need to author the checkpoints manually. Manual checkpoint specification is tedious and time-consuming [RPZ*17]. Our focus is on automatic image checkpoint selection, which is beneficial to the existing guider-follower systems.

Keyframe Extraction. Our goal is analogous to the existing works on keyframe extraction from videos. The problem of keyframe extraction has been extensively studied in the context of video summarization [DKD98, CSJ15, ZCSG16, KVGUH18, DM18, CZDZ18] and scene change detection [MJC95, Sha95, HK17]. The goal of video summarization, however, is to find a compact set of images that can well represent as much video content as possible. Thus they usually perform clustering on the color histograms or semantic information of video frames, and find the longest distance in the resulting clusters. Scene change detection aims to detect abrupt or gradual transitions between shots by measuring the significance of a change in video content. In contrast, our video analysis is more dependent on the walking path (e.g., its turning points), instead of the video content itself.

Occlusion Detection. The visibility or occlusion of the walking path plays an important role in automatic checkpoint selection. A straightforward solution to detect the occlusion of a path is first to estimate the depth map of a scene or even perform 3D scene reconstruction, and then check the visibility of the path against the estimated depth map or 3D scene. However, 3D reconstruction and depth map reconstruction [CXG*16, FSG17, DNZ*17, SSHP17, HK18, VKB*18] usually heavy, hard-to-implement, and have various restrictions (e.g., camera movements or maximum distance). Since our goal is to detect the occlusion of the walking path in the video, which is a particular case for occlusion detection, we design a lightweight, easy-to-implement algorithm tailor-made for path occlusion detection, without explicitly recovering the 3D information of a scene.

3. Criteria for Checkpoint Selection

A naïve method for selecting checkpoints is to uniformly and densely sample images along the walking path in the video. It might help followers to reach their destination, but easily generate tons of duplicated frames. The redundant checkpoints require unnecessary attention from followers, making it difficult for them to maintain proper situational awareness [RPZ*17]. On the other hand, when the guider chooses checkpoints too sparsely, followers might miss some critical information (e.g., turning points) and thus feel difficult to follow the path. The authors in Follow-My-Lead (FML) [RPZ*17] define four rules for manually positioning checkpoints: 1) with at least one important feature, 2) able to see the next checkpoint, 3) one checkpoint before and after each turn, staircase or door, and 4) no more than 30 seconds between checkpoints. However, these rules are somewhat subjective and some of them are not easy to be implemented into computational systems.



(a) Without arrows

(b) With arrows

Figure 3: *An example of turning right. Annotated arrows represent the motion direction, greatly reducing the number of required checkpoints. With arrows, the next checkpoint (b-Right) can be simply selected when the path starts to get occluded in (b-Left).*

It is thus necessary to refine these rules for automation systems. To further study the rules for checkpoint selection, we conduct a pilot study.

3.1. Pilot Study

We first recorded seven walking videos, while holding a camera (iPhone in our case) roughly at the height of the shoulders and facing the camera towards the walking direction. These videos include different navigation scenarios and have various characteristics, including indoor/outdoor, short/long, with/without stairs, with/without crowds. Different scenarios may appear in the same videos at different moments. The combination of these scenarios forms a large variety of environments in our problem.

Then, we invited five university students to do the study. We asked the participants to use a simple interface (i.e., for video browsing, addition and deletion of checkpoints) to interactively extract a minimum number of checkpoint images which they think are sufficient for others to navigate along the same paths in the videos. Once a participant completed a set of checkpoints for a video, we asked them the justification for each choice.

Unlike FML [RPZ*17], which required participants to make instant decisions to place checkpoints when following a path only once, we allowed our participants to familiarize themselves by browsing the videos back and forth to make a globally more optimal set of checkpoints.

Another difference is that the videos shown to the participants were annotated with directional arrows (e.g., Figure 3 (b)), since, a previous study [CB05] has shown that directional arrows have a positive impact on navigation. Therefore, we believed that annotations would influence the choice of checkpoints (Figure 3), and we decided to include such annotations in the study as we will have similar annotations in our final results.

Our goal here is to ask the participants to confirm the rules from the FML work [RPZ*17], replenish the common reasons behind the selection, and provide new criteria if any, instead of performing quantitative analysis. Thus, a large number of participants is not required.

3.2. Preliminary Study Results

All the participants decided to place a checkpoint directly at the starting point (possibly with small offset adjustment to place the

Figure 4: *Examples of turns which do not require checkpoints.*

projected path in the middle of the frame). They also selected a checkpoint at the destination, depending on its visibility in its previous checkpoint image (e.g., not being occluded). It means that a clear view of the starting direction (*Starting) and destination (*Destination) is essential for navigation. The FML rules do not include these two rules.

One interesting finding of our study is all the participants claimed that they would place one checkpoint before (to indicate the turning motion) and one after (to show what a follower will see next) every "turn" to indicate the turning of the path. This is similar to one of the rules presented in FML [RPZ*17]. However, we found that the actual set of selected checkpoints was not always consistent with this claim by the participants. In the interview period, when we asked the participants to explain why they did not place checkpoints before or after certain "turns" (e.g., Figure 4), the participants reconsidered their reasons. Further, they explained that it was due to visibility. As long as a turn is fully visible in the previous checkpoint image, it is unnecessary to place a new checkpoint before and after this turn. Thus, one possible rule for selecting checkpoints is after the path becomes invisible (*Visibility). This* often occurs after a turn (out of the image) or moving behind another object (occlusion) like a building wall (Figure 3 (b)). This finding is different from the turning rule of FML [RPZ*17].

If the path after a turn was straight but too long, 80% of the participants tended to put an additional checkpoint (*Long-distance)* after the turn to indicate go straight. The distance threshold is roughly 50m for indoor and 100m for outdoor, which is more or less similar to the 30s rules from FML (for mainly indoor) when the guider walk in average walking speed (~1.5m/s). Note that our values may not be accurate here due to the limited number of subjects, but the values work well for our results. Since a walking video when climbing stairs was unclear due to the small FOV, 80% of the participants tended to put checkpoints at the entry and exit of stairs (*Stair)* to indicate the action. All the participants carefully selected clear frames (*Clearness) instead of blurry frames, which are less informative for navigation, as checkpoints.*

Another finding from our study is that, although all of the participants selected almost similar checkpoint images in our provided videos, 40% of the participants placed extra checkpoints at different positions with apparent features. For instance, P2 and P5 placed an extra checkpoint within a straight road when there was a visually salient object (e.g., yellow bar, red fire extinguisher) in that frame. However, they also mentioned that some of such checkpoints might be used by followers for navigation verification but not vital. Besides, P3 and P4 said that they preferred to place checkpoints when there were text symbols such as letters and numbers in the image frames. If there was a one-way route segment (i.e., no other exits), 40% of the participants placed checkpoints inside this segment. Other participants only set one checkpoint respectively at the entry and the exit of the one-way segment. Two participants (P3 and P4) tended to place certain checkpoints according to the semantic events, such as "enter the door" and "go into the building." Similar to the FML rules, some of the criteria from the participants are still somewhat subjective, not clearly defined, or hard to be implemented for automation. The implementation of such rules is beyond the scope of this paper.

3.3. Summary of Criteria

We conclude a few rules that can be implemented into automation for checkpoint selection, as summarized below.

Starting. A checkpoint is needed near the starting point with navigation annotation (e.g., the projected path) being visible (e.g., centered in the frame).

Destination. If the destination cannot be clearly seen from the previous checkpoint, we need to add a checkpoint close to the destination.

Visibility. If the path starts to become invisible due to either occlusion or out-of-image, a new checkpoint at the occluded point of the path is needed to show the path next.

Long-distance. An extra checkpoint is needed if the distance between the current checkpoint and the next turn is too far away.

Stair. Checkpoints before and after stairs are needed at the entry and exit of stairs.

Clearness. A checkpoint image should not be blurry and should contain sufficient visual features for recognition.

4. Automatic Checkpoint Selection

With the refined set of criteria for selecting checkpoints in walking videos, we have developed our automatic system. We first extract a walking path in an input video. We then extract the required features such as occlusion and stairs. Finally, the checkpoints images are annotated by direction arrows to enhance the information for navigation.

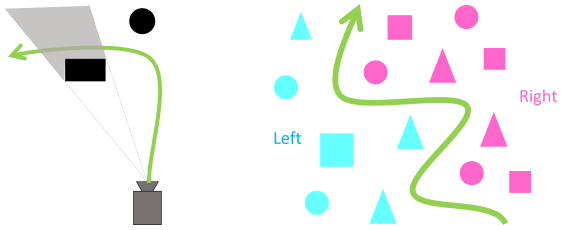
4.1. Path Extraction

The first step of our system is to extract the walking path from a walking video. Since guider-follower approaches do not rely on any absolute positioning technique, we do not have a walking path as input. Instead, we have to calculate it as the trajectory of the camera.

This vision-based positioning task can be done easily using existing SLAM-like methods such as visual inertial odometry (VIO) [LM13,LLB*15,FCDS17,QLS18] and concurrent odometry and mapping (COM) [NLZ17]. Since we use an iPhone to capture the input videos, we directly employ the SLAM-based ARKit for iPhone (or ARCore for Android devices) to obtain the relative walking path. With the extracted path, we can analyze the path and select image checkpoints along the path according to the rules discussed in Section 3.3. Specifically, we first create the first checkpoint at the starting position at the beginning of the path because of the "Starting" criteria. We then fine-tune this checkpoint until

(a) Video frame (b) Optical flow (c) Left-right map (d) Overlaid path (e) Occlusion result

Figure 5: *Illustration of our algorithm for occlusion detection of a path on a specific video frame.*



(a) Problem (b) Our key idea

Figure 6: *Our key idea for path occlusion detection. (a) Occlusion occurs if there are objects on its left (right) hand side when it turns left (right). (b) Our idea is to find left-hand-side objects (cyan) and right-hand-side objects (pink).*

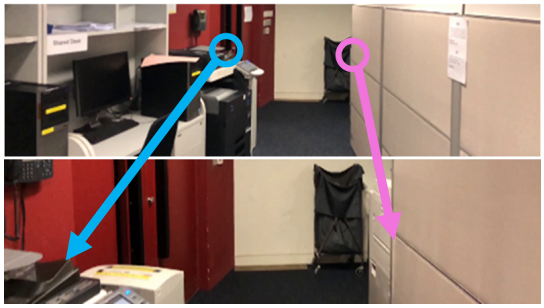


Figure 7: *If the objects are on the left (right) hand side of a moving path, they will move to the left (right) of the image when walking along the path.*

the path afterward is visible in the center region of the frame. After that, we search along the path and choose the next checkpoint image (Figure 3 (b)) if it satisfies with other criteria, as discussed below.

4.2. Path Occlusion Detection

Based on our pilot study, one of the criteria for selecting a checkpoint is whether a path becomes invisible (i.e., occluded or out of the image bounds) or not. To detect if part of the path is invisible, we project the path onto the previous checkpoint image. Once a point on the path is found out of the frame bounds, we directly select that position as the next checkpoint.

Invisibility is caused not only by out-of-frame bounds but also by occlusion. For such cases, we need to determine the occlusion of the path (Figure 5 (e)). One way to solve this particular occlusion detection problem is to employ a dense depth map. However, it is not easy to obtain a dense depth map with ordinary devices. RGB-D cameras are still not popular on mobile devices. Depth cameras involving infrared sensors do not work well outdoors due to the interference by the sunlight. Stereo cameras are still not standard in low-end smartphones. Shape-from-motion [HK18, VKB*18] on mobile has a distance limitation. Full 3D reconstruction from videos is usually too complicated and heavy. Users always prefer a simple and lightweight algorithm for getting results quicker at cheaper costs. For our navigation problem, the lightweight algorithm would allow users to quickly generate annotated checkpoint images on their mobile devices for sharing and/or printing right after they capture a navigation video. This is also important for refining the results through an iterative process by trying different paths and recapturing the corresponding videos. This motivates us to explore a simple and lightweight alternative solution.

We observe that a path gets occluded if and only if there is an object on its left (right) hand side when it turns left (right), as illustrated in Figure 6 (a). Our key idea is thus to determine left-hand-side (LHS) objects and right-hand-side (RHS) objects concerning the path, as illustrated in Figure 6.

It is easy to understand that objects on the left (right) hand side of the path will leave the image from any edges on its left (right) parts when they move behind the guider (Figure 7). Based on this observation, a feasible solution for determining LHS and RHS objects is to detect each pixel's *exiting* motion. Thus, we first compute the optical flow of adjacent frames in the video and then estimate the 2D pixel trajectory along the frames based on the computed flow. In our implementation, we use the dense optical flow (Figure 5 (b)) with dense inverse search (DISOptFlow [KTDVG16] in OpenCV) to track the movement of each pixel in the consecutive frames.

We then mark a pixel which leaves from the left (right) edge of the image, as "left" ("right"), corresponding to the red (blue) color region in Figure 5 (c). Otherwise, we mark the pixel as "undefined" (green regions in Figure 5 (c)). Applying this idea to all the image pixels leads to a map, which we call a *left-right* map.

We apply a spatial median filter on the left-right map to reduce the noise caused by optical flow. Furthermore, if parts of an object are on the left (right) hand side of the path, the other undefined parts of the same object are likely on the left (right) hand side of the path. Thus, we propagate the pixel value in the left-right map upward and backward to the pixels in the undefined regions if their colors are similar to the one with the left-right value. Other advanced object detection methods [LOW*20] may improve propagation accuracy, but more investigations are needed.

With the left-right map, we can determine the path occlusion. We

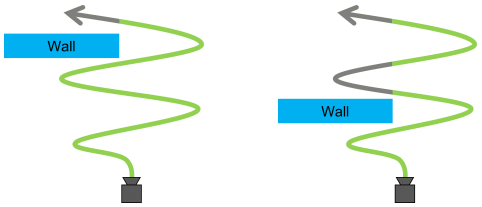


Figure 8: *Illustration of an ambiguous case when there are multiple turns. The object of one side can be before or after any turns. To address this, we make use of the timestamp of exiting.*

first project the 3D moving path onto the left-right map (Figure 5 (d)). If the path moves leftward in the 2D image and enters the "left" region in the left-right map or vice versa, we determine it as occluded. See an example of a path with the detected occlusion in Figure 5 (e).

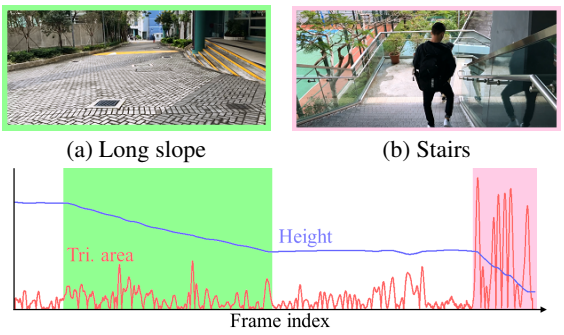
This method works well in most cases. However, when the path turns more than once in an image, ambiguous cases exist, as illustrated in Figure 8. It is because there is no front-back relationship data in the left-right map. Thus, the object can be either in front of or behind any of the turns in the image. To address this, in each pixel of the left-right map, we store the timestamp of exiting (i.e., frame index when the pixel leaves the image bounds in that frame). Objects with earlier timestamps are more likely in the front of the ones with later timestamps, since guiders are usually walking forward. The occlusion can then be determined with less ambiguity if the path enters the "left" or "right" pixels with an earlier timestamp than the current timestamp of the path for forwarding motion.

Discussions. Our left-right map can be used for path occlusion detection for the following reasons. First, a static object can occlude the path afterward only if it is in the image. If a static object is not in the camera frustum at any video frame, we can ignore it since it never occludes any path in our problem. If the static object appears in the video, there are two possibilities for its exit. This object can either remain in the image till the end of video, or exit from one of the image edges in a certain frame. For the former case, the object (e.g., a large building in the background) is probably too far from the camera, and there is no turn in the path. Thus, it never occludes the path. The latter case implies that there are turns in the path or the object is close to the camera. This object can occlude the path. As it leaves from the edges of the images, our left-right map works here.

Although our proposed method might not be as accurate as existing 3D reconstruction methods [CXG*16, FSG17, DNZ*17, SSHP17, HK18, VKB*18], it has its advantages in simplicity and efficiency. Most importantly, it generally works well for our primary task of automatic checkpoint selection. As mentioned above, if the path afterward becomes invisible due to object occlusion (e.g., the color-changing point of the path in Figure 5 (e)), we place a new checkpoint there.

4.3. Stair Detection

One of the criteria for placing checkpoints is the entry and exit of stairs. To accomplish this, we use a simple method to detect the



(c) Plot of height (blue) and its triangle area (red)

Figure 9: *There are a long slope (a) at the beginning (green region in (c)) and three stairs at the end of the video (pink region in (c)). We use the calculated triangle area curve to extract stairs.*

stairs in the video based on their 3D Y-Coordinate. It is not as simple as measuring the height difference between the 3D path points, since this cannot easily distinguish between stairs and long slopes with gradual changes. Our idea is to detect sudden changes in the height gradient. Figure 9 (c) shows a plot of height (blue line) of a walking video with a long slope at the beginning and three stairs at the end. We calculate the area of a triangle formed by each point of the height curve and its adjacent points (see the red line). The stairs can be extracted by finding the straight line segments in the curve of height between the high points of the triangle area curve. Figure 9 (b) gives an example of an automatically selected frame corresponding to the entry of a stair. Due to the typical structure of stairs, if the entry and exit of two consecutive stairs are close together, we mark them as one stair to reduce the need of image checkpoints.

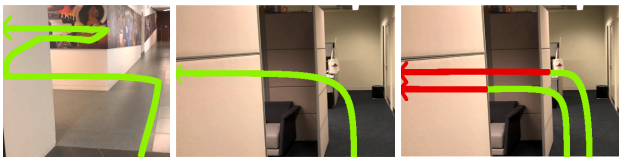
4.4. Other Rules

For the remaining rules, such as distance, we calculate the 3D norm-2 distance, and insert a checkpoint if the path distance exceeds a given threshold (50m for indoor and 100m for outdoor, selected manually in the current implementation). For the clearness rule, we refine the checkpoint position until the checkpoint image is not blurry. We detect the degree of blurriness by calculating the variance on the Laplacian filtered frame.

4.5. Occlusion-aware Checkpoint Annotation

After selecting the checkpoints from an input walking video, the next step is to annotate the checkpoint images with directional navigation arrows to improve the readability. The simplest way is to project the smoothed 3D path into the checkpoint images and display it as an arrow. However, it can be not very clear for users if physical objects occlude the path in the image (Figure 10). First, parts of the project path, which are occluded by physical objects, are meaningless and confusing for navigation. Second, it is not easy to determine the correct projected path without considering path occlusion when multiple objects occlude the paths.

These problems can be solved by detecting occlusions. Thus, our proposed path occlusion detection method can also benefit the annotation of the selected checkpoints. We visualize the directional



(a) (b) (c)

Figure 10: *(a) Path segments occluded by physical objects are meaningless or even confusing for navigation. (b) It is not easy to determine which path is the correct one without considering the occlusion of the path. (c) In contrast the path with occlusion illustrates a turning point more clearly. Here we show two possible paths indicated by different occlusion.*

arrows for visible and occluded parts (Figure 5 (e)) in green and red, respectively. Such colored directional arrows can help followers identify the correct path in the images more easily, as illustrated in Figure 10 (c). Besides, we can also annotate the stair entry by text to increase the readability (Figure 13).

5. Results and Experiments

We implemented our system using C++ with OpenCV. All the results were generated on a Windows laptop with 4 Cores i7 1.9GHz CPU with 16GB RAM. We captured all the videos using an iPhone X camera with a resolution of 2436×1126. Due to the memory issue, we down-scaled them by 4x for further processing. Table 1 shows the timing statistics of the core steps of our method on different videos using our unoptimized code. The steps for checkpoint selection (Select) and stair extraction (Stair) are very fast (0.25s and 0.001s on average for 1700 frames). The optical flow and left-right map calculation steps are the most time consuming, and each takes half a minute on average. We believe that parallel programming (e.g., GPU or multi-threading) can greatly accelerate the speed. Moreover, our current implementation operates on every pixel in the video, while only the pixels near the projected path in the checkpoint images are needed. By focusing on the calculation of these core pixels, we can significantly accelerate our method.

5.1. Results

Figure 11 shows the example results of our path occlusion detection method, with the corresponding left-right maps. Our algorithm detects path occlusion very accurately, thanks to the left-right maps. Figure 1 shows one of the example results for checkpoint images automatically selected by our method. By checking the walking video, which can be found in the demo video, this small set of selected checkpoints faithfully capture the key information of the walking path. Figure 12(a & b) are the selected checkpoint images by two participants in our criteria study (Section 3). We can see that our method leads to the results similar to those by human subjects. We also compared our automatic checkpoint selection method with an alternative solution based on video summarization. Figure 12 (c) shows a set of selected checkpoints by a technique based on video summarization [YL95]. Such techniques often lead to duplicated frames or missing frames. Since video summarization cares more about video content than a compact representation of a walking path, their results are not good enough for navigation.

Length	#Frame	#CP	OptFlow	LR map	Stair	CP select
0:27	1534	5	25.385s	35.010s	0.001s	0.184s
0:38	1146	4	18.761s	24.486s	0.001s	0.199s
1:22	2476	7	44.211s	54.439s	0.001s	0.307s
0:59	1784	5	29.554s	41.220s	0.001s	0.274s
0:44	1317	5	22.662s	29.636s	0.001s	0.242s
1:13	2201	6	41.838s	47.801s	0.001s	0.283s

Table 1: *The timing statistics of core steps of our method for different videos. CP stands for "checkpoint" and LR stands for "Left-Right."*

5.2. Evaluation

Existing navigation works have already shown that a set of manually selected checkpoints from videos can help users (i.e., followers) to find their way more easily. Thus, in this paper, we focus on the evaluation of the guider side. Our goal here is to demonstrate that our system can select checkpoints similar to those by human subjects. We conducted two experiments to evaluate the effectiveness of our system, including preference and error.

First, we were interested in evaluating whether potential followers would prefer our automatically selected checkpoint images compared to those by four different simple automatic selection methods, including 1) video summarization [YL95]: we set the threshold to achieve the same number of checkpoints as ours for fair comparison; 2) Extracted with fixed interval: we place checkpoints for every 200 frames; 3) Extracted by a fixed range of curvature: we place a new checkpoint if its curvature satisfied $0.2 \leq \kappa \leq 4.0$ and its distance to its previous checkpoint exceeded by 100 frames; 4) Extracted by fixed turning angle: we traced the path direction and placed a new checkpoint if its direction changed for $70^o - 110^o$ and its distance to the previous checkpoint exceeded by 100 frames. We did not use the same number of checkpoints as ours for methods (2)-(5), as we believe these methods might have their optimal numbers of checkpoints.

First, we invited eight participants to examine the checkpoint images generated by different methods in random order. These participants were familiar with the environments in our testing videos. We provided a questionnaire to the participants for their preference. They had to provide a score (from 0 to 9; the higher, the better) for each set of results.

Figure 15 shows the results of this preliminary study. Extracted checkpoints using fixed interval showed a good preference (average preference = 6.95) while they were not concise enough (average conciseness = 5.85). Our results were more preferred (average preference = 7.23) and had higher quality (average quality = 7.03) than fixed intervals. Meanwhile, our method extracted only 5.4 image checkpoints on average, and were more concise (average conciseness = 7.0).

Curvature and turning angle are highly noise-sensitive and scale-dependent. Thus, they led to unsatisfied results (average preference = 3.45 and 4.65, respectively). Finally, video summarization [YL95] only considers video content, and our participants least prefer it (average preference = 2.85).

In the second experiment, we used the same interactive system as in Section 3 to ask the participants to refine our automatically

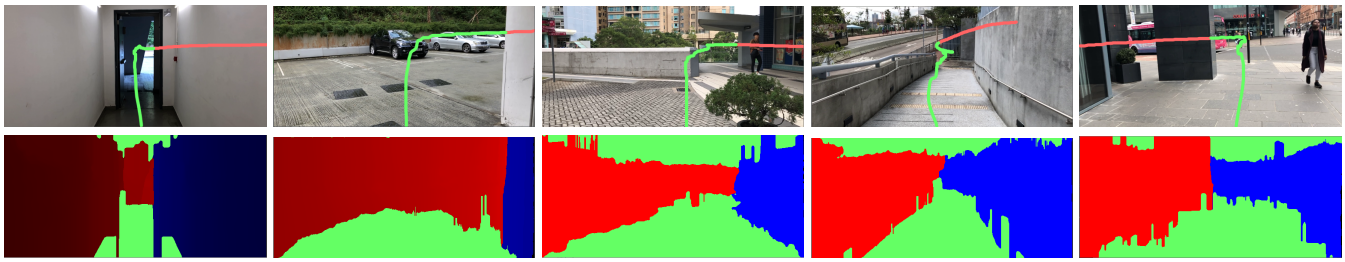
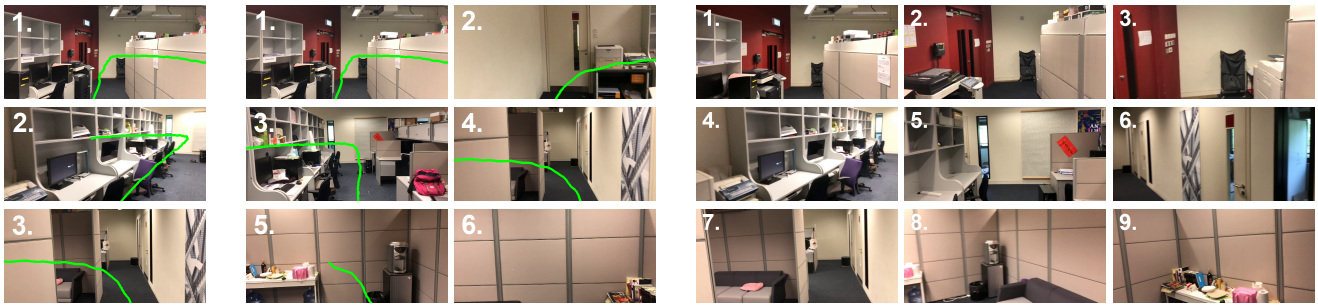


Figure 11: *Example results of our path occlusion detection algorithm for different input video frames and their corresponding left-right maps. The intensity of the red and blue colors is coded by the timestamp.*



(a) By P1 (b) By P5 (c) By video summarization [YL95]

Figure 12: *The checkpoints selected manually by two participants (a & b), and automatically by video summarization (c) [YL95]. Our corresponding result is shown in Figure 1. This video is used in the checkpoint selection criteria study.*

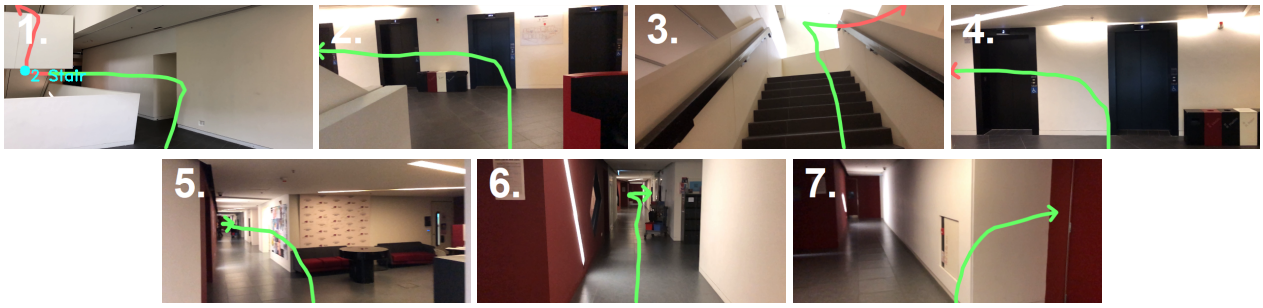


Figure 13: *A set of automatically selected checkpoints from a video other than those used in criteria study.*

Figure 14: *A set of automatically selected checkpoints from another video other than those used in criteria study.*

selected checkpoints interactively. We showed five videos to the eight participants (i.e., 40 cases in total) and used our automatically selected checkpoints as initial results. The participants had to add/remove/move the checkpoints freely. We measured the number of changes by the participants and calculated the offset in our results. As shown in Figure 16, 14 cases were directly accepted by the participants without adding or removing checkpoints. 10 cases were required to add one checkpoint. Among all the cases, six checkpoints were moved (yellow bar in Figure 16) by the participants with 83 frame offset on average. Our automatically selected checkpoints are very close to the manual selected ones.

6. Discussions

Manually choosing a compact set of critical checkpoints is not an easy task. Most of the participants said sometimes they were

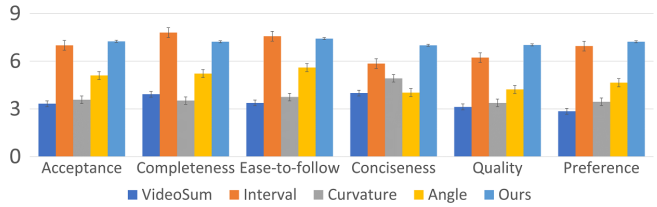


Figure 15: *Our evaluation results for comparing different methods. Error bars represent the standard error of mean.*

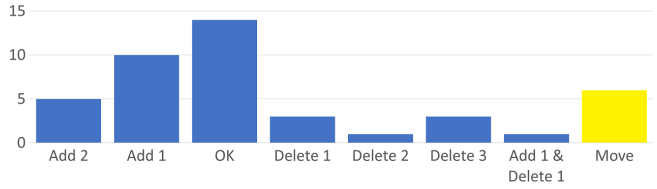


Figure 16: *The total number of checkpoint changes for all cases in our study. "Move" (Yellow bar) are additional operations among 40 cases in the blue bars.*

confused to place good checkpoints. P4 and P5 from the criteria study commented that they were too familiar with some routes to overlook several essential checkpoints easily. It is an interesting dilemma for the guider-follower as guiders are supposed to be very familiar with the path. At the same time, those who are too familiar with the path might not be sensitive enough during the manual selection of checkpoints. This observation further justifies the usefulness of our system.

Another interesting finding is that our timestamped left-right map looks visually similar to a depth map of the scene, as shown in Figure 17. It gives us an excellent direction to study whether exiting motion is a reliable depth cue or not, and whether it can help to perform 3D scene reconstruction in the future.

6.1. Limitations and Future Works

Our current method suffers from several limitations. Our method highly depends on the performance of a SLAM tracking implementation (ARKit in our case) and the optical flow implementation, which is sometimes unstable and scene dependent. We would investigate more information from the video and measured data (e.g., IMU in smartphones, backward motion) to improve the performance. We believe that more accurate reconstruction algorithms and more robust implementations will appear due to the advancement of computer vision and the popularity of AR in the near future.

Our current implementation did not handle dynamic objects such as pedestrians, and such moving objects may affect our results. In the future, we would use object recognition to segment out pedestrians or other dynamic objects. We did not handle "hole" cases if a path is invisible and reappear shortly in the same image. These cases often happen when the path moves behind thin objects (e.g., human, streetlights, and columns) in the scene. A filtering of short-term occlusion might solve the problem.

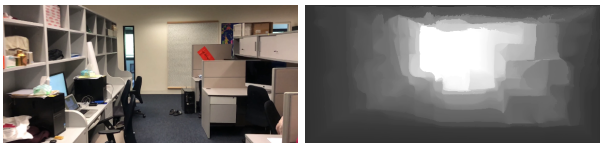


Figure 17: *Our timestamped left-right map (Right) looks similar to depth map.*

Some of the criteria suggested during manual checkpoint selection are not easy to be implemented. For example, further studies are needed to examine the possibility of robust recognition of common reference objects such as post boxes, eye-catching objects, and landmarks in videos. Also, multi-way paths or stairs (i.e., with multiple exits and/or entries) might not be properly handled by our simple solution. One possible solution is to employ visual recognition techniques to identify the exits and entries.

Finally, guider-follower methods often only support one-to-one navigation, meaning that there is only one starting point, one destination, and one path. However, in real scenarios, followers may often come from different starting points (e.g., different entries), go to different destinations (e.g., specific rooms), and with different paths based on their personal conditions (multiple-to-multiple navigation). In the future, we plan to investigate such navigation. It is not easy since it is not efficient for the guider to walk every possible path from every place. One possible idea is to combine multiple walking videos from multiple guiders if there are shared routes, and to generate a set of checkpoint images on the fly based on personal preferences or real-time road conditions.

Acknowledgements

This research was supported by grants from City University of Hong Kong (Project No. 7005590) and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU, and funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161.

References

[BGVGT*17] BRENA R. F., GARCÍA-VÁZQUEZ J. P., GALVÁN-TEJADA C. E., MUÑOZ-RODRIGUEZ D., VARGAS-ROSALES C., FANGMEYER J.: Evolution of indoor positioning technologies: A survey. *Journal of Sensors 2017* (2017). 2

[BS06] BEEHAREE A. K., STEED A.: A natural wayfinding exploiting photos in pedestrian navigation systems. In *Proceedings of the conference on Human-computer interaction with mobile devices & services* (2006), ACM, pp. 81–88. 1, 2

[CB05] CHITTARO L., BURIGAT S.: Augmenting audio messages with visual directions in mobile guides: an evaluation of three approaches. In *Proceedings of the international conference on Human computer interaction with mobile devices & services* (2005), ACM, pp. 107–114. 3

[CSJ15] CHU W.-S., SONG Y., JAIMES A.: Video co-summarization: Video summarization by visual co-occurrence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), IEEE, pp. 3584–3592. 1, 3

[CW10] CHANG Y.-J., WANG T.-Y.: Comparing picture and video

prompting in autonomous indoor wayfinding for individuals with cognitive impairments. *Personal and Ubiquitous Computing 14*, 8 (2010), 737–747. 2

[CXG*16] CHOY C. B., XU D., GWAK J., CHEN K., SAVARESE S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2016), Springer, pp. 628–644. 3, 6

[CZDZ18] CAI S., ZUO W., DAVIS L. S., ZHANG L.: Weakly-supervised video summarization using variational encoder-decoder and web prior. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), Springer, pp. 184–200. 1, 3

[DKD98] DEMENTHON D., KOBLA V., DOERMANN D.: Video summarization by curve simplification. In *Proceedings of the ACM International Conference on Multimedia* (1998), ACM, p. 211–218. 1, 3

[DM18] DATT M., MUKHOPADHYAY J.: Content based video summarization: Finding interesting temporal sequences of frames. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (2018), IEEE, pp. 1268–1272. 1, 3

[DNZ*17] DAI A., NIESSNER M., ZOLLHÖFER M., IZADI S., THEOBALT C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. In *ACM Transactions on Graphics (ToG)* (2017), vol. 36, ACM, p. 76a. 3, 6

[DP17] DAVIDSON P., PICHÉ R.: A survey of selected indoor positioning methods for smartphones. *IEEE Communications Surveys & Tutorials 19*, 2 (2017), 1347–1370. 2

[FABF13] FALLAH N., APOSTOLOPOULOS I., BEKRIS K., FOLMER E.: Indoor human navigation systems: A survey. *Interacting with Computers 25*, 1 (2013), 21–33. 2

[FCDS17] FORSTER C., CARLONE L., DELLAERT F., SCARAMUZZA D.: On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics 33*, 1 (2017), 1–21. 4

[FSG17] FAN H., SU H., GUIBAS L. J.: A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (2017), IEEE, pp. 605–613. 3, 6

[HGL*09] HILE H., GRZESZCZUK R., LIU A., VEDANTHAM R., KOŠECKA J., BORRIELLO G.: Landmark-based pedestrian navigation with enhanced spatial reasoning. In *International Conference on Pervasive Computing* (2009), Springer, pp. 59–76. 2

[HK17] HAN S., KIM J.: Video scene change detection using convolution neural network. In *Proceedings of the International Conference on Information Technology* (2017), ACM, pp. 116–119. 1, 3

[HK18] HOLYNSKI A., KOPF J.: Fast depth densification for occlusion-aware augmented reality. In *ACM Transactions on Graphics (ToG)* (2018), ACM, p. 194. 3, 5, 6

[HVC*08] HILE H., VEDANTHAM R., CUELLAR G., LIU A., GELFAND N., GRZESZCZUK R., BORRIELLO G.: Landmark-based pedestrian navigation from collections of geotagged photos. In *Proceedings of the international conference on mobile and ubiquitous multimedia* (2008), ACM, pp. 145–152. 1, 2

[KO10] KAMEDA Y., OHTA Y.: Image retrieval of first-person vision for pedestrian navigation in urban area. In *Proceedings of the International Conference on Pattern Recognition (ICPR)* (2010), IEEE, pp. 364–367. 1, 2

[Kol04] KOLBE T. H.: Augmented videos and panoramas for pedestrian navigation. In *Proceedings of the Symposium on Location Based Services & TeleCartography* (2004), Springer. 1

[KTDVG16] KROEGER T., TIMOFTE R., DAI D., VAN GOOL L.: Fast optical flow using dense inverse search. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2016), Springer, pp. 471–488. 5

[KVGUH18] KANEHIRA A., VAN GOOL L., USHIKU Y., HARADA T.: Viewpoint-aware video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), IEEE, pp. 7435–7444. 1, 3

[LLB*15] LEUTENEGGER S., LYNEN S., BOSSE M., SIEGWART R., FURGALE P.: Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research 34*, 3 (2015), 314–334. 4

[LM13] LI M., MOURIKIS A. I.: High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research 32*, 6 (2013), 690–711. 2, 4

[LOW*20] LIU L., OUYANG W., WANG X., FIEGUTH P., CHEN J., LIU X., PIETIKÄINEN M.: Deep learning for generic object detection: A survey. *International journal of computer vision 128*, 2 (2020), 261–318. 5

[MHB11] MCGOOKIN D., HERTELEER I., BREWSTER S.: Transparency in mobile navigation. In *CHI Extended Abstracts on Human Factors in Computing Systems* (2011), ACM, pp. 1903–1908. 1

[MJC95] MENG J., JUAN Y., CHANG S.-F.: Scene change detection in an mpeg-compressed video sequence. In *Digital Video Compression: Algorithms and Technologies* (1995), vol. 2419, International Society for Optics and Photonics, pp. 14–26. 1, 3

[MS07] MILLONIG A., SCHECHTNER K.: Developing landmark-based pedestrian-navigation systems. *IEEE Transactions on Intelligent Transportation Systems 8*, 1 (2007), 43–49. 1

[MUS15] MARCHAND E., UCHIYAMA H., SPINDLER F.: Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics (TVCG) 22*, 12 (2015), 2633–2651. 2

[NLZ17] NERURKAR E., LYNEN S., ZHAO S.: System and method for concurrent odometry and mapping, Nov. 23 2017. US Patent App. 15/595,617. 4

[QLS18] QIN T., LI P., SHEN S.: Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics 34*, 4 (2018), 1004–1020. 2, 4

[RPZ*17] ROY Q., PERRAULT S. T., ZHAO S., DAVIS R. C., PATTENA VANIYAR A., VECHEV V., LEE Y., MISRA A.: Follow-my-lead: Intuitive indoor path creation and navigation using interactive videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2017), ACM, pp. 5703–5715. 1, 2, 3, 4

[Sha95] SHAHRARAY B.: Scene change detection and content-based sampling of video sequences. In *Digital Video Compression: Algorithms and Technologies* (1995), vol. 2419, International Society for Optics and Photonics, pp. 2–14. 1, 3

[SSHC15] SHU Y., SHIN K. G., HE T., CHEN J.: Last-mile navigation using smartphones. In *Proceedings of the International Conference on Mobile Computing and Networking* (2015), ACM, pp. 512–524. 1, 3

[SSHP17] SCHÖPS T., SATTLER T., HÄNE C., POLLEFEYS M.: Large-scale outdoor 3d reconstruction on a mobile device. *Computer Vision and Image Understanding 157* (2017), 151–166. 3, 6

[VKB*18] VALENTIN J., KOWDLE A., BARRON J. T., WADHWA N., DZITSIUK M., SCHOENBERG M., VERMA V., CSASZAR A., TURNER E., DRYANOVSKI I., ET AL.: Depth from motion for smartphone ar. In *ACM Transactions on Graphics (ToG)* (2018), ACM, p. 193. 3, 5, 6

[WARG13] WITHER J., AU C. E., RISCHPATER R., GRZESZCZUK R.: Moving beyond the map: automated landmark based pedestrian guidance using street level panoramas. In *Proceedings of the international conference on Human-computer interaction with mobile devices & services* (2013), ACM, pp. 203–212. 1

[WBRM14] WENIG D., BRENDING S., RUNGE N., MALAKA R.: Using split screens to combine maps and images for pedestrian navigation. *Journal of Location Based Services 8*, 3 (2014), 179–197. 1

[WGL*18] WANG Q., GUO B., LIU Y., HAN Q., XIN T., YU Z.: Crowdnavi: Last-mile outdoor navigation for pedestrians using mobile

crowdsensing. *Proceedings of the ACM on Human-Computer Interaction 2*, CSCW (2018), 179. 1, 2

[WSS*16] WENIG D., STEENBERGEN A., SCHÖNING J., HECHT B., MALAKA R.: Scrollinghome: bringing image-based indoor navigation to smartwatches. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services* (2016), ACM, pp. 400–406. 1, 3

[XYZ*15] XU H., YANG Z., ZHOU Z., SHANGGUAN L., YI K., LIU Y.: Enhancing wifi-based localization with visual clues. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2015), ACM, pp. 963–974. 2

[YL95] YEUNG M. M., LIU B.: Efficient matching and clustering of video shots. In *Proceedings., International Conference on Image Processing* (1995), vol. 1, IEEE, pp. 338–341. 2, 7, 8

[YWL12] YANG Z., WU C., LIU Y.: Locating in fingerprint space: Wireless indoor localization with little human intervention. In *Proceedings of the International Conference on Mobile Computing and Networking* (2012), ACM, pp. 269–280. 2

[YWY*16] YIN Z., WU C., YANG Z., LANE N., LIU Y.: ppNav: Peer-to-peer indoor navigation for smartphones. In *IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (2016), pp. 104–111. 1, 2

[YYR17] YAN Z., YE M., REN L.: Dense visual slam with probabilistic surfel map. *IEEE transactions on visualization and computer graphics (TVCG) 23*, 11 (2017), 2389–2398. 2

[ZCSG16] ZHANG K., CHAO W.-L., SHA F., GRAUMAN K.: Summary transfer: Exemplar-based subset selection for video summarization. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (2016), IEEE, pp. 1059–1067. 1, 3

[ZSL*17] ZHENG Y., SHEN G., LI L., ZHAO C., LI M., ZHAO F.: Travi-navi: Self-deployable indoor navigation system. *IEEE/ACM Transactions on Networking 25*, 5 (2017), 2655–2669. 2