World-Stabilized Annotations and Virtual Scene Navigation for Remote Collaboration

Steffen Gauglitz **Benjamin Nuernberger** **Matthew Turk** **Tobias Höllerer**
Department of Computer Sciene, University of California, Santa Barbara
Santa Barbara, CA 93106, USA
{sgauglitz, bnuernberger, mturk, holl}@cs.ucsb.edu

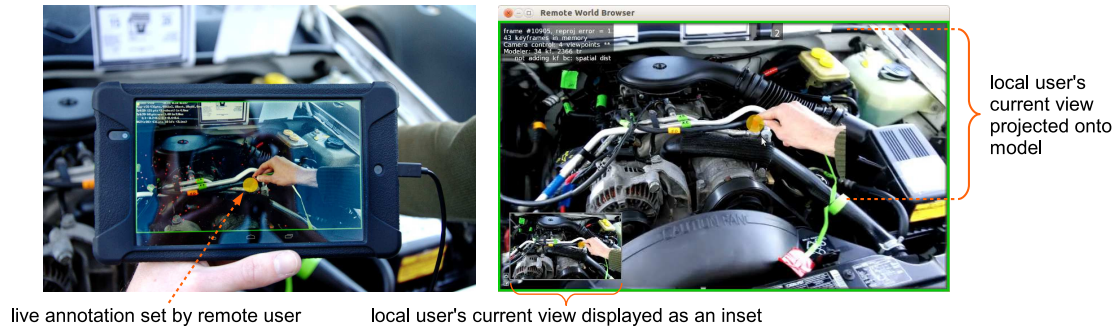

Figure 1. Live AR-based remote collaboration with our prototype. Left: the local user in front of a car engine, identifying a particular element which the remote user has marked with the yellow dot. Right: the remote user's view onto the scene, which (at this moment) shows more context than the local user's current view. The latter is shown as an inset on the bottom left as well as being projected onto the model. The remote user can browse this environment independently of the local user's camera and can set annotations, which are immediately visible to the local user in AR.

ABSTRACT

We present a system that supports an augmented shared visual space for live mobile remote collaboration on physical tasks. The remote user can explore the scene independently of the local user's current camera position and can communicate via spatial annotations that are immediately visible to the local user in augmented reality. Our system operates on off-the-shelf hardware and uses real-time visual tracking and modeling, thus not requiring any preparation or instrumentation of the environment. It creates a synergy between video conferencing and remote scene exploration under a unique coherent interface. To evaluate the collaboration with our system, we conducted an extensive outdoor user study with 60 participants comparing our system with two baseline interfaces. Our results indicate an overwhelming user preference (80%) for our system, a high level of usability, as well as performance benefits compared with one of the two baselines.

Author Keywords

CSCW; video-mediated communication; augmented reality; telepresence

UIST 2014, October 5–8, 2014, Honolulu, HI, USA.

ACM 978-1-4503-3069-5/14/10 ...$15.00.
http://dx.doi.org/10.1145/2642918.2647372

ACM Classification Keywords

H.5.1 [Information interfaces and presentation] Multimedia information systems: Artificial, augmented, and virtual realities; H.5.3 [Information interfaces and presentation] Group and organization interfaces: Computer-supported cooperative work

INTRODUCTION

In recent years, the use of video conferencing has become ubiquitous. However, with current technology, users are limited to passively watching disjoint video feeds which provide no means for interaction with the remote physical environment. As effective collaboration often involves sharing, exploring, referencing, or even manipulating the physical environment, tools for remote collaboration should provide support for these interactions. Researchers have explored various means to do so; however, two of the most common limitations in existing work are that the remote user's view onto the scene is constrained to the typically small field of view of the local user's camera, and that any support for spatially referencing the scene is contingent upon a stationary camera.

In this work, we leverage computer vision and the paradigm of augmented reality (AR) to facilitate more immersive interaction with the remote environment in general, and to address the aforementioned limitations in particular. We describe a fully functional system for mobile remote collaboration, running on off-the-shelf hardware, in which the remote user can (a) control a virtual camera and thus explore the live scene independently of the local user's current camera position, and (b) communicate via spatial annotations that are immediately

visible to the local user in AR (cf. Figure 1). Our system does not require any preparation or instrumentation of the environment. Instead, the physical scene is tracked and modeled incrementally in real time and in 3D, using monocular vision-based simultaneous localization and mapping (SLAM) and subsequent surface modeling. The emerging model then supports anchoring of annotations, virtual navigation, and synthesis of novel views.

We further present an extensive outdoor user study with 60 participants (30 teams) comparing our system with two baselines. Both user ratings and task performance are discussed in detail. To our knowledge, our study is among the first user studies overall to rely purely on visual SLAM technology as an enabling technology (rather than as the subject of interest) in an outdoor environment.

RELATED WORK

Research on remote collaboration and telepresence is multifaceted; for example, several systems focus on creating fully immersive, three-dimensional telepresence experiences with increasingly lower instrumentation barriers (e.g., [28, 37]).

Support for spatial references to the remote scene in video-mediated collaboration or telepresence has been an active research topic. Notable early works include "VideoDraw" [41] and the "DoubleDigitalDesk" [44]. Modalities that have been investigated include remote pointers [2, 5, 11, 21], drawing onto a live video stream [7, 11, 14, 21, 22, 34], and transferring videos of hand gestures [17, 22, 33]. These annotations are then displayed to the collaborator on a separate screen in a third-person perspective [11, 18], on a head-worn display [2, 17, 33], or via projectors [14].

However, in order to support spatially referencing physical objects, all of these systems either assume a stationary camera (at least during the relevant interaction), since otherwise the virtual annotations lose their referents [2, 7, 11, 14, 17, 21–23], or require extensive equipment and prepared environments to track and thus maintain the locations of the annotations [5, 33]. Further, the remote user's view onto the physical scene is either restricted to a stationary camera [11, 22] or tightly coupled to the local user's head or body movement [2, 5, 17, 21, 23], thus forcing the remote user to constantly re-orient and ask the local user to hold still (or, in the case of [2], enforcing this by freezing both users' views) when referencing an object.

One alternative is to use a robot which can be controlled by the remote user [14, 24]; however, this requires specialized hardware and limits the range of operation.

In our work, we leverage computer vision-based tracking and modeling and the paradigm of collaborative AR [3] to support world-stabilized annotations and virtual camera movements. Early implementations of both concepts for the purpose of collaboration are reported by Mayol et al. [29] and Lee and Höllerer [26].

Our system follows the conceptual framework for an AR-based collaboration system proposed in Gauglitz et al. [12], and our user study follows a similar pattern. However, while the prototype described therein works in planar environments only and offers only a camera "freeze" feature without any further navigation or view synthesis, our system operates in environments of arbitrary geometric complexity (which has implications for almost all components of the system), and provides a feature-rich virtual navigation that allows the remote user to explore the environment independently of the local user's current camera position. (Conceptually, this has been described as a possibility in Gauglitz et al. [12].) Further, our system consists of two stand-alone off-the-shelf hardware entities that communicate via wireless network.

Sodhi et al. [39] present a system with a similar vision. They use a customized setup with a mobile device and two active depth sensors mounted onto a monopod which gives the ability to reconstruct the environment (on the local user's side) and to reconstruct and transfer hand gestures (from the remote user's side) in 3D. In contrast, our system needs only an off-the-shelf tablet, but uses simpler annotations. Another difference worth noting is the method of scene navigation by the remote user: Sodhi et al. equip the remote user with a mobile device as well and use the device's physical movement to navigate the remote (virtual) environment, while we use virtual navigation. We contrast the two approaches in more detail below. They conducted a usability study reporting results with respect to ease of use, etc., but no comparative or performance-based evaluation was reported.

The system by Adcock et al. [1] also reconstructs the environment via active depth sensors and allows the remote user to control the viewpoint (via a touch interface) and draw annotations, which are displayed using a statically mounted projector.

Further, Jo and Hwang [20] presented an interactive, fully mobile system which allows for world-stabilized drawings, albeit only for panoramas (i.e., rotation movements). Two particular interesting aspects of this work are the physical navigation of the panorama by the remote user and switching between front and back cameras (upon permission by the local user).

With respect to reconstructing a physical scene from image data and navigating it based on this reconstruction, our work bears similarity in particular with Snavely et al.'s "Photo tourism" [38] and similar works. However, while Photo tourism deals with a (potentially large) set of photos in a batch manner and offers offline navigation, we process a live video in real time and offer live viewing of the evolving model. Further, we extract a detailed 3D surface rather than using planar surface proxies for the rendering.

With respect to individual features, our system bears similarity with further works: Tatzgern et al. [42] very recently presented an AR system which allows the user to switch between a live video ("AR view") and synthesized views of a reconstructed environment ("VR view"). Lastly, Sukan et al. [40] presented a study comparing physical navigation with virtual navigation via saving and revisiting virtual snapshots. Here, we employ similar techniques for the interaction with a remote environment.

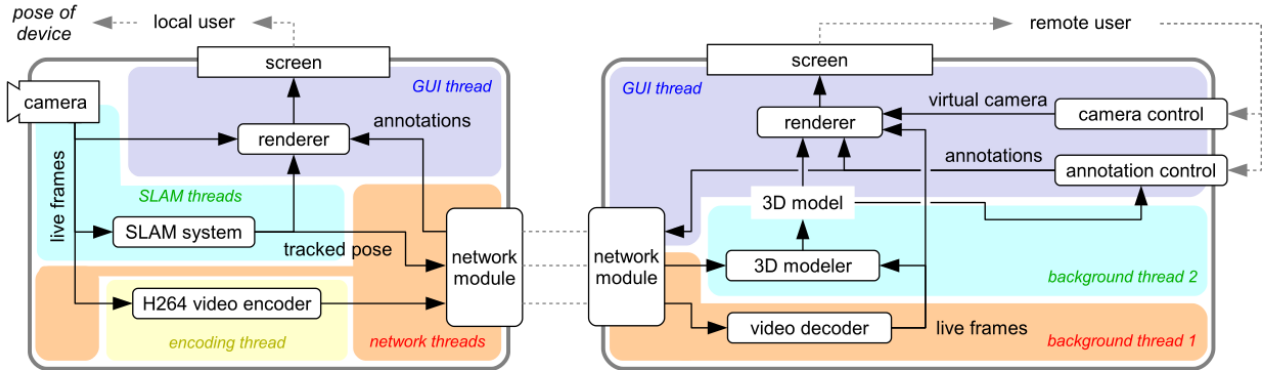

Figure 2. System architecture of both the local user's system (left; running on an Android-based lightweight tablet or smartphone) and the remote user's system (right; running on a commodity PC with Ubuntu). The main components are described in the text in detail.

SYSTEM OVERVIEW

Figure 2 shows an overview of the system architecture of both the local user's and the remote user's system. Since device hardware (camera and display), network communication, real-time processing, and background tasks are involved, both systems employ a host of components and threads.

LOCAL USER'S SYSTEM

The local user's interface, running on a lightweight tablet, is intentionally simple. From the user's perspective, it behaves exactly like a live video of the user's own view plus AR annotations, i.e., a classic magic lens. (We emphasize that the local user's view is not affected by remote user's camera control.) The only control the user exerts during its operation is by manipulating the position and orientation of the device. We note that the system could equally be used with other AR displays such as a head-worn or projective display.

Under the hood, the system runs a SLAM system and sends the tracked camera pose along with the encoded live video stream to the remote system. The local user's system receives information about annotations from the remote system and uses this information together with the live video to render the augmented view.

The system is implemented as an Android app, running on several state-of-the-art Android devices. For the user study, we used a Google Nexus 7 2013, a 7" tablet powered by a Qualcomm Snapdragon S4 Pro with 1500 MHz Krait quad core CPU. The core SLAM implementation, including access to the live camera frames and support for rendering for them, has been provided to us by Qualcomm. Communication with the SLAM system, handling of the raw image data, and rendering are implemented in C/C++, while higher-level app structure, user interface, and network communication are implemented in Java, with data exchange between the two layers via JNI. The live frames are encoded as a H.264 video stream. A minimal HTTP server streams the data (encoded video, tracked camera pose, and other meta-data) upon request from a remote connection, and manages remote requests for insertion/deletion of annotations (encoded as HTTP requests).

The system operates at 30 frames per second. We measured system latencies using a camera with 1000 fps which observed a change in the physical world (a falling object passing a certain height) as well as its image on the respective screen. The latency between physical effect and the local user's tablet display — including image formation on the sensor, retrieval, processing by the SLAM system, rendering of the image, and display on the screen — was measured as 205 ± 22.6 ms.

REMOTE USER'S SYSTEM

The remote user's interface, running on a commodity PC (without high-end GPUs or such), starts off as a live video stream, but is augmented by two controls, the *camera control* and *annotation control*.

The system consists of five main modules — network module, 3D modeler, camera control, annotation control, renderer — and the framework to hold them together. Due to this modular framework, different implementations for each module can be readily swapped in and out upon the start of the program via command line arguments. For example, for comparing our prototype against two baseline interfaces in our user study, we simply replaced the respective modules with simpler ones. In a similar fashion, we also implemented modules that load virtual 3D models (instead of modeling from live video) and allow for other types of camera control.

The latency between physical effect and the remote user's display — including image formation on the local user's sensor, retrieval, processing by the SLAM system, video encoding, transmission via wireless network, video decoding, rendering, and display — was measured as 251 ± 22.2 ms; i.e., less than 50 ms between the local and the remote user's display.

Network module

The network module receives the data stream from the local user's device, sends the incoming video data on to the decoder, and finally notifies the main module when a new frame (decoded image data + meta-data) is available.

3D modeler

From the live video stream and associated camera poses, we construct a 3D surface model on the fly as follows. We select keyframes based on a set of heuristics (good tracking quality, low device movement, minimum time interval & translational distance between keyframes), then detect and describe features in the new frame using SIFT [27]. We then choose the four closest existing keyframes, match against their features

# of keyframes in model	1–10	11–25	>25
Keypoint detection	32 ± 2	32 ± 3	34 ± 2
Keypoint description	904 ± 124	868 ± 144	795 ± 167
Stereo matching	121 ± 59	166 ± 38	153 ± 45
Merging & filtering of vertices	$<1 \pm 1$	5 ± 1	9 ± 2
Updating tetrahedralization	$<1 \pm 1$	2 ± 2	3 ± 3
Calculating graph costs	10 ± 6	51 ± 14	128 ± 28
Solving graph cut	$<1 \pm <1$	$1 \pm <1$	1 ± 1
Extracting & smoothing surface	4 ± 8	9 ± 21	21 ± 9
Total time	1082 ± 140	1159 ± 175	1201 ± 208

Table 1. Average timings of the 3D modeler to process and integrate one new keyframe into the model, running on a single core of a 3 GHz Intel i7 CPU with 4 GB RAM. All times are given in milliseconds, with associated standard deviations. Incorporating all logs from the user study to be discussed below, the data in the three columns are based on 300, 429, and 481 keyframe integrations, respectively.

(one frame at a time) via an approximate nearest neighbor algorithm [30] and collect matches that satisfy the epipolar constraint (which is known due to the received camera poses) within some tolerance as tentative 3D points. If a feature has previously been matched to features from other frames, we check for mutual epipolar consistency of all observations and merge them into a single 3D point if possible; otherwise, the two 3D points remain as competing hypotheses.

Next, all tentative 3D points are sorted by the number of supporting observations and are accepted one by one unless one of their observations has been previously "claimed" by an already accepted 3D point (which, by construction, had more support). We require at least four observations for a point to be accepted, and we further remove a fraction of points with the largest 3D distances to their two nearest neighbors. The algorithm is thus robust to even large fractions of mismatches from the stereo matching stage.

To obtain a surface model from the 3D point cloud, we implemented the algorithm by Hoppe et al. [16]: First, a Delaunay tetrahedralization of the point cloud is created. Each tetrahedron is then labeled as "free" or "occupied," and the interface between free and occupied tetrahedra is extracted as the scene's surface. The labeling of the tetrahedra works as follows: A graph structure is created in which each tetrahedron is represented by a node, and nodes of neighboring tetrahedra are linked by edges. Each node is further linked to a "free" (sink) node and an "occupied" (source) node. The weights of all edges depend on the observations that formed each vertex; for example, an observation ray that cuts through a cell indicates that this cell is free, while a ray ending in front of a cell indicates that the cell is occupied. Finally, the labels for all tetrahedra are determined by solving a dynamic graph cut problem. For details on how the edge weights are computed we refer the reader to [16].

We refined Hoppe et al. [16]'s algorithm by taking the orientation of observation rays to cell interfaces into account, which reduces the number of "weak" links and thus the risk that the graph cut finds a minimum that does not correspond to a true surface.

As Hoppe et al. describe, both updating the graph costs and solving the graph cut can be implemented in a incremental manner, with cost almost independent of the overall model size. As these steps were found to take up a negligible amount of time in our application (cf. Table 1), for simplicity we did not even implement the incremental algorithm. Nonetheless, the entire processing of a new keyframe and updating of the 3D surface model is completed within 1-1.5 seconds (cf. Table 1), which is easily fast enough for our purposes, as it is smaller than the interval at which keyframes are added on average. Currently, the vast majority of the time is taken up by the keypoint description, which is independent of the model size. If necessary, the computation could be significantly sped up by using a more efficient descriptor implementation or algorithm and/or implementing the incremental graph update. Thus, the overall modeling algorithm is highly scalable to environments much larger than demonstrated here.

Camera control (virtual navigation)

The remote user's ability to navigate the remote world via a virtual camera, independent of the local user's current location, is one of the key contributions of our work.

As mentioned earlier, Sodhi et al. [39] provide a similar feature, but use physical device movement for navigation. While this is arguably very intuitive, using physical navigation has two disadvantages; one being generally true for physical navigation and the other one being specific to the application of live collaboration: First, the remote user needs to be able to physically move and track his/her movements in a space corresponding in size to the remote environment of interest[1], and "supernatural" movements or viewpoints (e.g., quickly covering large distances or adopting a bird's-eye view) are impossible. We refer to Bowman et al. [4] for a more detailed discussion of physical vs. virtual travel, and Sukan et al. [40] for a comparison on a particular task in the context of AR. Second, it does not allow coupling of the remote user's view to the local user's view (and thus have the local user control the viewpoint) without breaking the frame of reference in which the remote user navigates. Lanir et al. [25] presented a study on the issue of control of viewpoint with mixed results, suggesting that it may be dependent on the particular task.

We thus decided to use virtual navigation, and we deem it important that our navigation gives the remote user the option of coupling his/her view to that of the local user.

Within virtual navigation, we decided to use a keyframe-based navigation as explained in the following for several reasons: first, mapping unconstrained 3D navigation to 2D controls requires relatively complex interfaces [19]; second, our model is inherently constrained by what has been observed by the local user's camera, and a keyframe-based approach offers a natural way to constrain the navigation accordingly; third, a keyframe-based approach allows for image-based rendering with high levels of fidelity.

Therefore, the camera control module continually stores new keyframes with their associated camera poses from the live video stream if the tracking quality is good enough. (These

[1] In fully immersive virtual reality, "redirected walking" [36] can be used to "fit" a virtual environment into a smaller physical space under certain conditions.

keyframes are independent of the ones used by the modeler; they serve a different purpose and are maintained separately.) Keyframes that have become obsolete because they are close to a newer keyframe are automatically discarded.

The individual controls were designed to be as intuitive as possible and resemble controls familiar from other exploration tools such as Google Street View and Photo tourism [38]. Viewpoint transitions are implemented by smoothly interpolating between the camera poses and are rendered as visually seamlessly as possible (cf. the description below).

Freeze & back to live

The application starts with the remote view coupled to the local user's live view. With a single right-click, the remote user "freezes" his/her camera at the current pose, for example in order to precisely set annotations, or as a starting point for further navigation. Whenever the remote user's view is not coupled to the local user's view, the latter is displayed to the remote user as an inset (cf. Figure 3). A click onto this inset or pressing 0 immediately transitions back to the live view. (This feature, resembling the "Frame & Freeze" technique by Güven et al. [15], is the only camera control feature implemented in the prototype in [12].)

Panning & zooming

By moving the mouse while its right button is pressed, the user can pan the view in a panorama-like fashion (rotate around the current camera position). To prevent the user from getting lost in unmapped areas, we constrain the panning to the angular extent of the modeled environment. To ensure that the system does not appear unresponsive to the user's input while enforcing this constraint, we allow a certain amount of "overshoot" beyond the allowed extent. In this range, further mouse movement away from the modeled environment causes an exponentially declining increase in rotation and visual feedback in the form of an increasingly intense blue gradient along the respective screen border (Figure 3). Once the mouse button is released, the panning quickly snaps back to the allowed range. Thus, the movement appears to be constrained by a (nonlinear) spring rather a hard wall.

The user can also zoom into and out of the view with the scroll wheel. Zooming is implemented as a change of the virtual camera's field of view (rather than dollying) to avoid having to deal with corrections for parallax or occlusions from objects behind the original camera position.

Click to change viewpoint

When the user right-clicks into the view, we compute the 3D hit point, and subsequently find the camera whose optical axis is closest to this point (which may be the current camera as well). We then transition to this camera and adapt yaw and pitch such that the new view centers around the clicked-upon point. This allows the user to quickly center on a nearby point as well as quickly travel to a far away point with a single click.

Saving & revisiting viewpoints

Further, the user can actively save a particular viewpoint to revisit it later (similar to the navigation investigated by Sukan et al. [40]). Pressing `Alt` plus any number key saves the current viewpoint; pressing the respective number key alone revisits

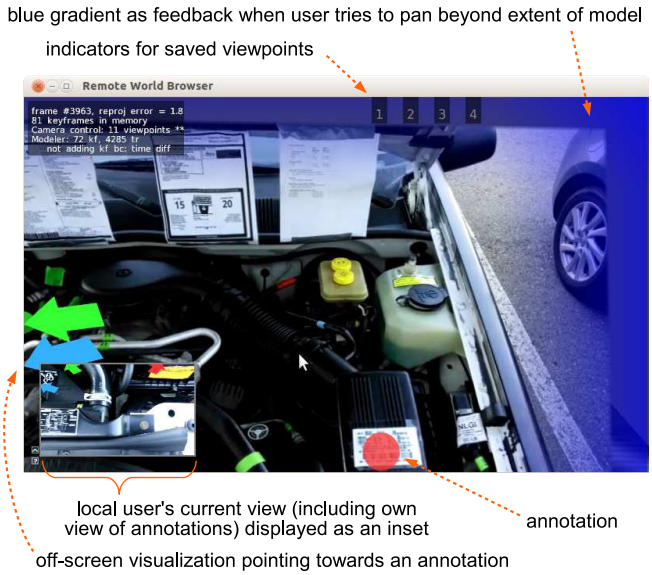

Figure 3. Screenshot of the remote user's interface.

this view later. Small numbers along the top of the screen indicate which numbers are currently in use (see Figure 3).

We postulate that this keyframe-based navigation is simple and intuitive and allows for flexible exploration of a scene. However, there are situations in which it is limited: it does not allow navigation to a never-visited viewpoint, or control over each degree of freedom individually, as may be desired by advanced users for fine-grained maneuvering [4, 19].

Annotation Control

In addition to being able to control the viewpoint, the remote user can set and remove virtual annotations. Annotations are saved in 3D world coordinates, are shared with the local user's mobile device via the network, and immediately appear in all views of the world correctly anchored to their 3D world position (cf. Figures 1 and 3).

For this prototype, we implemented only simple, animated spherical markers. If annotations are outside the user's current field of view, an arrow appears along the border of the screen pointing towards the annotation (see Figure 3; also cf. [12]). Annotations are "pulsing" with 1 Hz and 15% amplitude to increase their visual saliency. Together with the independent viewpoint control as described above, the remote user can thus effectively direct the local user to elements outside the local user's current field of view.

The remote user sets a marker by simply left-clicking into the view (irrespective if "live" or "decoupled"). The depth of the marker is derived from the 3D model, presuming that the user wants to mark things on physical surfaces rather than in mid-air. Pressing the space bar removes all annotations. More complex and/or automatic erasure management [11, 20] could be integrated as desired.

These annotations were sufficient for our task (cf. user study tasks below), but other tasks may require more complex annotations. As discussed earlier, other works have experi-

mented with 2D drawings [14, 22, 34] or transfer of hand gestures [17, 18, 22], which are undoubtedly more expressive but thus far have been used mostly with stationary cameras. More complex annotations like this could be integrated (now world-stabilized as well) as needed.

Renderer

Finally, the renderer renders the scene using the 3D model, the continually updated keyframes, the incoming live camera frame (including live camera pose), the virtual camera pose, and the annotations. In addition to a generally desirable high level of realism, a particular challenge rather unique to our application is the seamless transition to and from the live video (also cf. [42]). That is, as the virtual camera approaches the physical camera, the views should become identical, with no noticeable transition from "model view" to "live view." We achieve this by using image-based rendering as follows.

The 3D model is rendered as a polygonal model, upon which the images of the live frame and closest keyframes are projected using projective texture mapping. As the virtual camera moves, the different textures are faded in and out by adapting their opacity. In areas which are modeled accurately, transitions are thus completely seamless, with lighting effects naturally blending in. More sophisticated image-based rendering techniques (e.g., [6]) could be integrated as appropriate.

We note that the live view may extend beyond the area currently modeled in 3D. To ensure that these areas do not suddenly "disappear" immediately after transitioning away from the live view, we extend the model with a proxy plane (at the model's average distance to the camera) on which textures can be projected. To soften artifacts, we blur the parts of the projective textures that fall onto this part.

In effect, when the remote user's virtual camera pose is identical (coupled) to the local user's current physical camera pose, the rendered image is identical to the live video, and transitions to and away from it are seamless.

Due to the nature of our camera control, the camera is often at the location of a previously cached keyframe (albeit with possibly modified yaw, pitch, or field-of-view), which enables image-based rendering with high levels of fidelity. The 3D model is rarely visible as a polygonal model, thus modeling artifacts are rarely apparent. However, the more accurate the model, the better the transitions can be rendered, and the more precisely annotations can be anchored.

We encourage the reader to inspect the virtual navigation and the rendering in the supplemental video.

SYSTEM LIMITATIONS & DISCUSSION

While our prototype delivers a robust, real-time AR experience, there are some assumptions and system limitations.

Level of detail of model

Building upon a tetrahedralization of a sparse point cloud, the modeling approach taken here is very fast (as demonstrated above), but results in a model that does not exhibit the level of detail as achievable, for example, with dense volumetric fusion [9, 32, 35]. While the existence of the 3D model is quintessential for anchoring of annotations and rendering, a relatively coarse level of detail is, arguably, acceptable, as the keyframe-based navigation and rendering de-emphasize modeling artifacts.

However, techniques to create a more detailed model exist, and since our system is modular — none of the other components depend on this particular modeling algorithm — other algorithms could be plugged in as needed. This includes algorithms to densify the point cloud [10] while keeping the overall approach the same, using other, computationally more intensive, vision-based algorithms [35], or (where permissible by the application) active depth sensor-based modeling [32].

Static scene

Our system generally assumes that the scene is static. While the SLAM system on the local user's side is quite robust to partial and/or gradual changes in the scene, adapting the modeling, navigation and rendering would require an active detection and invalidation of changed regions.

Stereo initialization

Like any monocular SLAM system, our system requires a stereo initialization (i.e., a distinct camera movement) before it tracks robustly. This is done quickly and thus not a problem if the user is aware of the requirement; however, we feel that it stands in the way for truly transparent and user-friendly operation. The research community is investigating ways to reduce this burden [13, 31].

Occlusion of annotations on local side

Currently, the 3D model is available only on the remote user's side. Thus, annotations can be correctly occluded by the physical scene by the remote user's renderer, but not by the local user's renderer. While other depth cues (most notably, parallax) still indicate the annotation's location, it would be desirable to enable occlusion. The remote system could send either the model geometry or alternatively some sort of local visibility information per annotation back to the local device. Designing an elegant, bandwidth-efficient solution for either approach is an interesting aspect for future work.

One-way annotations

Our system currently limits creation of annotations to the remote user. The rationale is that the local user can spatially refer to the environment with his/her hands (as seen in Figure 1 and the supplemental video) — it is the remote user who needs additional means (cf. related work using one-way annotations [11, 22, 25]). However, if deemed appropriate, it would be straightforward to also allow the local user to create annotations, for example by tapping onto his/her screen.

USER STUDY: DESIGN & METHOD

To evaluate our system, we conducted a remote expert–local worker user study comparing our system with two baselines: a video+audio only interface and an interface with static annotations (also called "markers" throughout the study). Both the local and the remote users were study participants.

We first conducted several pilot study trials with a total of 20 users (10 teams), during which we refined study parameters, overall procedure, and training procedure.

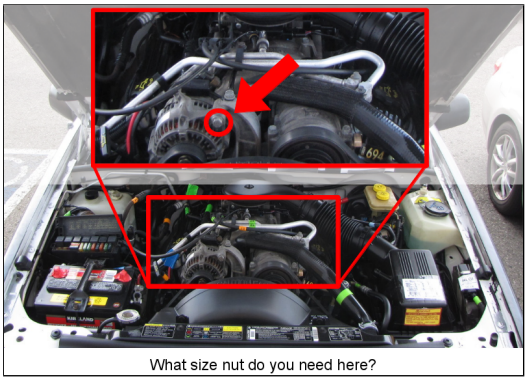

Figure 4. One example out of 80 individual tasks. These instructions were provided to the remote user, who then needed to communicate the information to the local user.

Task & Physical setup

We chose a "car repair" task for the study. The local user stood in front of a car, hood open, and received help from the remote user in "identifying the problem" (cf. [7] for a similar scenario). The study took place outdoors, and while we used a relatively isolated location with no direct sunlight, several environment factors were beyond our control, including weather, light conditions, passers-by, and noise from a nearby street and a nearby airport. The study's external conditions were thus close to a real scenario. Network infrastructure and the remote user's PC were mounted onto a cart and positioned adjacent to the car such that the participants could communicate verbally, but not see each other.

To make sure that the individual tasks were roughly equivalent, quick enough to perform, independent of the individual user's dexterity, and not dangerous in any way, we used proxy tasks that would require similar communication between the users but little or no physical labor, such as locating individual elements and finding pieces of information. For example, instead of unscrewing a bolt, we asked the users to identify its size by testing it with a set of provided nuts, which requires the same communication between the users in order to identify the correct element but little physical labor. For each individual task, the remote user was given simulated expert knowledge in the form of a specific question (e.g., size of a particular screw or cable, rating of a particular fuse, serial number of a part) and a diagram indicating where the answer could be located (see Figure 4). The remote user then had to communicate this information to the local user, who had to locate the requested information and write it down.

Conditions

In all conditions, the two users were able to talk to each other without restrictions.

- **Interface A: video only.** The live video from the local user's camera is streamed to the remote user; the remote user does not have any means of interacting with the video or providing visual/spatial feedback. This is similar to using today's standard video conferencing tools.
- **Interface B: video + static markers.** In addition to the features in condition A, the remote user can click into the video to create a marker visible to both users. However, the marker's position is stored in *screen* coordinates and thus moves with the camera rather than "sticking" to the world object. This condition is effectively similar to related works that assume a stationary camera [11].
- **Interface C: our prototype as presented above.**

As in Gauglitz et al. [12], conditions B and C both allowed up to five concurrent markers in different colors, with further clicks re-setting the oldest marker.

Experimental Design

We used a within-subjects design with one independent variable (interface type) and one dependent variable (task completion time). We also recorded the number of errors and obtained several user ratings via questionnaires. The order of the interfaces was completely balanced, with each of the six possible orderings traversed by five of the 30 teams. For each team, three lists with 15 tasks were created at random, with the remaining tasks reserved for training.

Our hypotheses about the study's outcome were as follows:

H1 Users will complete the task faster with interfaces B and C than with interface A.

H2 Users will complete the task faster with C than with B.

H3 Users will prefer interface C over both A and B.

Participants

60 users (18–30 years (mean 20.8), 29 female, 31 male) participated in the main study, working together in 30 teams (8 female/female, 7 female/male, 6 male/female, 9 male/male (local/remote user)). Each user received a compensation of USD 10; all teams additionally competed for a bonus of USD 10 / 20 / 40 per person for the top 5 / second fastest / fastest error-free task performances for all three conditions combined.

In two teams not included in the numbers above, one user was color blind. It remains unclear whether this affected the task performance. However, we used color extensively (markers, labels in the car, etc.), and at least one of the users was unable to disambiguate a few of the elements. We thus decided not to include the data from these two trials in the analysis.

Procedure & Training

Each participant completed a color blind test and a pre-study questionnaire with background information. The general setup and task were then explained in detail.

For each session, the study administrator explained the respective interface features in detail, then the users conducted several training tasks with the respective interface. Each user completed a minimum of five training tasks for each new feature and was asked explicitly if they were comfortable using the interface before proceeding to the timed session. After each session, the users filled out an intermediate questionnaire rating this interface. Lastly, each user filled out a post-study questionnaire and received their compensation.

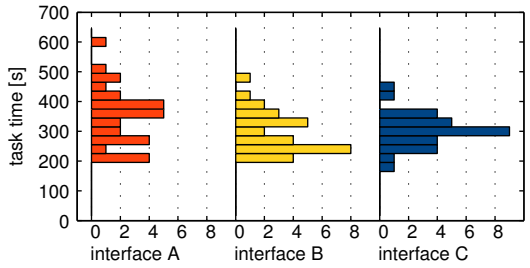

Figure 5. Histogram of task times per interface.

During the pilot study trials, it quickly became clear that not all of the camera control features that our prototype (interface C) featured were necessary for this particular environment and task, and that the limited amount of time prohibited explaining and training the user on each of them. We thus concentrated the training on a subset of features that appeared to be most useful in this context, namely, the freezing of the camera, and the saving/revisiting of viewpoints. However, the other features (panning, zooming, change viewpoint via click) were still available and were occasionally discovered and used by participants.

To ensure a consistently high quality of the required stereo initialization, the study administrator conducted the initialization step (until the modeler had started to extract 3D surface) before handing the device to the local user.

USER STUDY: RESULTS & DISCUSSION

Task performance

Overall, 98.5% of the tasks were answered correctly (21 errors at a total of $30\times3\times15$ tasks). Analyzing the number of errors, Mauchly's test indicated that the assumption of sphericity against interface had not been violated ($W(2)=0.95$, $p=0.50$), and no significant effect of interface on the number of errors was found using a one-way repeated measures ANOVA ($F(2,58)=0.47$, $p=0.63$, and $\eta^2_{\text{partial}}=0.016$). Additionally, we note that 5 of the 21 errors were made on two particular subtasks in which the expert knowledge diagram may have been confusing. We conclude that all users worked meticulously and thus that the comparison of the task times is meaningful.

Analyzing the task times, Mauchly's test indicated that the assumption of sphericity had not been violated ($W(2)=0.99$, $p=0.93$). With a one-way repeated measures ANOVA, we found a significant effect of interface on task completion time with $F(2,58)=6.94$, $p=0.0020$, and $\eta^2_{\text{partial}}=0.19$. Post hoc comparisons using Tukey's HSD test indicated that users were significantly faster with both interfaces B ($M=313.6$, $SD=69.6$) and C ($M=317.5$, $SD=57.6$) than with interface A ($M=364.7$, $SD=96.7$), thus supporting hypothesis H1. No significant difference was found between B and C; hence, hypothesis H2 was not supported.

Questionnaires

In the intermediate questionnaires filled out immediately after each session (i.e., before the next interface was introduced), the users were asked to rate their level of agreement on a 7-point scale to the statements, "This interface helped me to solve the task," "This interface made me feel confident that I was doing the task correctly," and "I had difficulties using this interface." The responses are aggregated in Figure 6.

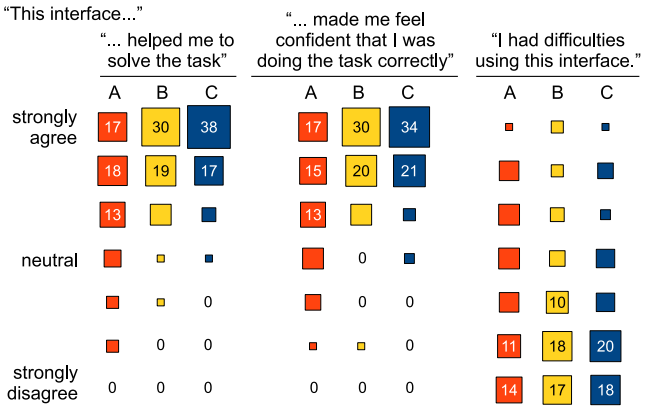

Figure 6. Results from intermediate questionnaires: interface ratings.

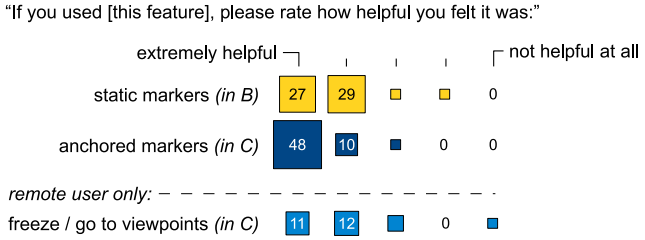

Figure 7. Results from intermediate questionnaires: individual features.

For these ratings, we decided to use a non-parametric test for the analysis to avoid the assumptions of interval data, normality and spherecity.

According to Friedman's test, the ratings in response to the first two statements differ significantly among the interfaces ($\chi^2(2)=34.5$, $p < 10^{-7}$, and $\chi^2(2)=37.8$, $p < 10^{-7}$, respectively). Pairwise comparisons with Bonferroni's correction applied indicated that both B and C were rated better than A. For the third question ("I had difficulties..."), Friedman's test also indicated a significant difference ($\chi^2(2)=8.1$, $p=0.017$), but pairwise comparisons with Bonferroni's correction applied only revealed a possible borderline significant difference between A and C at $p = 0.019$ (compared to the corrected threshold of 0.017).

Users were further asked to rate the helpfulness of individual features on a 5-point scale from "extremely helpful" to "not helpful at all" (Figure 7). Wilcoxon's paired signed rank test ($V=25$, $p < 10^{-4}$) showed that the anchored markers (in interface C) were perceived as more helpful than the static markers (in interface B), with 80% of the users perceiving the former as "extremely helpful." The camera control features were perceived as "extremely" or "very" helpful by 77% of the remote users.

In the post-study questionnaire, users were asked to rank the three interfaces ("Which interface did you like best /

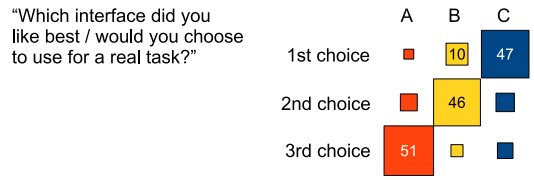

Figure 8. Results from post-study questionnaire: interface preference.

would you choose to use for a real task?"). There is a significant difference in ranking according to Friedman's test ($\chi^2(2) = 71.42$, $p < 10^{-15}$). Pairwise comparisons (with Bonferroni's correction applied) indicated that all pairwise differences are significant. 80% of the users selected interface C as their first choice (cf. Figure 8), supporting H3. The preference for C is 83% among tablet users and 77% among PC users, but this difference was not significant according to Wilcoxon's paired signed rank test ($V = 29$, $p = 0.45$).

Open-ended questions on the questionnaires revealed several interesting details, most notably an inadvertent side effect of the world-stabilization of markers: Since the markers in interface C had a constant world size, they ended up being quite large when the user got very close to the object, and were thus too large for some tasks, as described by this user: *"Overall, the markers and viewpoints were extremely helpful. However, [...] for the tasks where text needed to be located, the marker was much bigger than the words."* Having constant screen size, the markers in B did not have this side effect.

Several users expressed their preference for interface C, but commented that too many keys had to be memorized: *"[C] was more useful [...] but it was sometimes difficult to remember which buttons would navigate to which screen shots, what to press to unfreeze the pane, and so on. However, I imagine it would get much easier with additional practice"*; *"[C] was by far the most helpful user interface, but it was a bit difficult to use due to the [number of] buttons."*

Discussion

To summarize the results, an overwhelming majority of the participants (in both roles) preferred our system over both baselines (H3 supported); users performed significantly faster with it than with a video-only baseline (H1 supported); but no significant difference in task performance was found in comparison with a static marker condition (H2 not supported).

Differences in task performance

We note two particular artifacts of the study design that may have reduced the differences in task performance between the interfaces in general and counteracted potential benefits of interface C in particular:

First, as users started, the difficulty of verbally giving spatial and directional instructions — including confusion regarding "left" and "right" (relative to the car's driving direction or the user's perspective?) and "up" and "down" (in world or screen coordinates?), as well as misidentification of elements — was very apparent, which supports the need for spatial annotations. However, as an artifact of the training, the within-subjects design, and the small task environment, users quickly became experts in giving and understanding directions for this particular task, possibly more so than becoming experts in using the interfaces. Oftentimes, users came up with names for different parts of the engine, which did not have to be technically accurate in order to be effective (the most creative moniker may have been "rainbow keyboard" for the fuse box). As one user commented when asked about the usefulness of the camera control: *"[It] wasn't very useful since I already knew the layout of the engine from previous tasks."* For real applications, we might assume the opposite: users become familiar with the interfaces available to them and communicate with new partners in new environments. To account for this, different environments could be used for each training and interface session, which however poses a challenge in terms of practicality and ensuring fair conditions across all sessions.

Second, because of the increasingly familiar environment, and additionally motivated by the incentives, the tasks became very fast-paced. Thus, the remote user's mental load of understanding the next instructions (cf. Figure 4), aligning them with his/her mental model of the engine, and then with his/her view of the scene, was oftentimes slower than the actual task execution, which required little physical labor and could thus be completed very quickly. As the time-critical path shifted from the local to the remote user, a potential benefit of the virtual camera control — namely, that the remote user could browse and familiarize him/herself with the environment (e.g., for the next task) — became less relevant, while a potential downside (increased complexity) became more relevant. One user commented: *"Using the multiple views was a little more hectic on the 'expert's' side, but only because we were trying to complete the tasks as quickly as possible. In a situation in which this technology would be used, this feature would no doubt be very helpful for quality and efficiency of task completion."* We note that this is also an artifact of the study setup: in a real application, no simulated expert knowledge has to be understood, and the local user's task may require more physical labor or travel.

Other factors that may have played a role are: While the users' ratings suggest that they did not perceive interface C as more difficult to use, it may require more training for optimal use (cf. comments on the number of keys above). Lastly, with C, some users took extra time to carefully position the camera to save viewpoints for later use, but the amortization of this time investment was limited by the short task duration.

Type of task

In this work, in order to validate our system design and establish its usefulness, we chose a relatively simple task with clearly defined roles, as commonly used in this context (e.g., [2, 5, 11, 12, 34]). Looking forward, it would be important to study more complex collaborative tasks. As the features that the system provides are very general in nature, we are confident that they provide benefits in a variety of contexts.

CONCLUSIONS

In this paper, we described a system for mobile remote collaboration that uses real-time computer vision-based tracking and modeling of the environment to facilitate live, remote

controlled AR annotations and remote virtual scene navigation. We believe that these features add promising new dimensions to remote collaboration. We further presented an extensive comparative user study.

Our system was overwhelmingly preferred by users. With regard to task performance time, our study showed a significant benefit compared with only one of the two baseline interfaces. We discussed several artifacts which may have contributed to this outcome. Some of these are common to many studies of collaboration, such as small environments, proxy tasks, and simulated expert knowledge. Thus, we hope that our study also informs the design of future studies in this area. Altogether, our results demonstrate the maturity and usability of our system, as well as producing insights into which aspects of the interface can be further improved. Limitations of the system which future work might target have also been discussed above.

While the system is complex under the hood, we feel that one important advantage of our approach is that it seamlessly extends the ubiquitous videoconferencing paradigm; it adds to it, but it does not take anything away. If desired, the user can fall back to simply watching the live video feed.

In the bigger picture, our work bridges and creates a synergy between video conferencing (Skype, Apple FaceTime, Google Hangouts, etc.) and remote world exploration (Microsoft Photosynth, Quicktime VR [8], Google StreetView, [43], etc.) in a unique coherent interface via live collaborative AR. However, our prototype has only started to tap into the significant potential of this synergy. Areas to explore include the integration of more complex annotations such as gestures or gaze [29] in a world-stabilized manner and the integration of cloud-based data (e.g., maps/larger models of the environments). The live navigation of remote worlds — both within virtual navigation as well as in comparison with physical navigation [20, 39] — also warrants further investigation.

This synergy could further be explored beyond the two-user remote expert–local worker scenario, with respect to both roles and numbers of users: We note that the presented paradigm is equally applicable to scenarios with shared expertise or a local expert (e.g., in an educational context), and naturally extends to more than one local user and/or more than one remote user. Here, the environment model is to integrate video streams and annotations from all users.

ACKNOWLEDGMENTS

We thank Qualcomm, and in particular Serafin Diaz, Daniel Wagner and Alessandro Mulloni, for making their SLAM implementation available to us for this project. We thank Susan Fussell for advice concerning the design of the user study, and Yuqi Chen for her help with the statistical analysis. Further, we thank Jhon Faghih-Nassiri and Krithika Chandrasekar for looking into multimedia streaming, Lukas Gruber and Domagoj Baričević for OpenGL tricks, John O'Donovan letting us borrow the car for the study, Michael Liebling and Kevin Chan for letting us use their high-speed camera, and Mock Suwannatat and Julia Gauglitz for their help recording the video.

This work was supported by NSF grants IIS-1219261 and IIS-0747520, ONR grant N00014-14-1-0133, and a Chancellor's fellowship for S.G. from UCSB.

REFERENCES

1. Adcock, M., Anderson, S., and Thomas, B. RemoteFusion: Real time depth camera fusion for remote collaboration on physical tasks. In *Proc. ACM SIGGRAPH VRCAI* (2013), 235–242.
2. Bauer, M., Kortuem, G., and Segall, Z. "Where are you pointing at?" a study of remote collaboration in a wearable videoconference system. In *Proc. IEEE ISWC* (1999), 151 –158.
3. Billinghurst, M., and Kato, H. Collaborative augmented reality. *Commun. ACM 45* (July 2002), 64–70.
4. Bowman, D. A., Kruijff, E., LaViola, J., and Poupyrev, I. *3D User Interfaces: Theory and Practice*. Addison-Wesley, Boston, 2005.
5. Chastine, J., Nagel, K., Zhu, Y., and Hudachek-Buswell, M. Studies on the effectiveness of virtual pointers in collaborative augmented reality. In *Proc. IEEE 3DUI* (March 2008), 117 –124.
6. Chaurasia, G., Duchene, S., Sorkine-Hornung, O., and Drettakis, G. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph. 32*, 3 (2013), 30:1–30:12.
7. Chen, S., Chen, M., Kunz, A., Yantaç, A. E., Bergmark, M., Sundin, A., and Fjeld, M. SEMarbeta: mobile sketch-gesture-video remote support for car drivers. In *Proc. Augmented Human Int'l Conf.* (2013), 69–76.
8. Chen, S. E. QuickTime VR: an image-based approach to virtual environment navigation. In *Proc. ACM SIGGRAPH* (1995), 29–38.
9. Curless, B., and Levoy, M. A volumetric method for building complex models from range images. In *Proc. ACM SIGGRPAPH* (1996), 303–312.
10. Furukawa, Y., and Ponce, J. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 8 (2010), 1362–1376.
11. Fussell, S. R., Setlock, L. D., Yang, J., Ou, J., Mauer, E., and Kramer, A. D. I. Gestures over video streams to support remote collaboration on physical tasks. *Hum.-Comput. Interact. 19* (2004), 273–309.
12. Gauglitz, S., Lee, C., Turk, M., and Höllerer, T. Integrating the physical environment into mobile remote collaboration. In *Proc. ACM MobileHCI* (2012).
13. Gauglitz, S., Sweeney, C., Ventura, J., Turk, M., and Höllerer, T. Model estimation and selection towards unconstrained real-time tracking and mapping. *IEEE Trans. Vis. Comput. Graphics 20*, 6 (2014), 825–838.
14. Gurevich, P., Lanir, J., Cohen, B., and Stone, R. TeleAdvisor: A versatile augmented reality tool for remote assistance. In *Proc. ACM CHI* (2012), 619–622.

15. Güven, S., Feiner, S., and Oda, O. Mobile augmented reality interaction techniques for authoring situated media on-site. In *Proc. IEEE and ACM ISMAR* (Washington, DC, USA, 2006), 235–236.
16. Hoppe, C., Klopschitz, M., Donoser, M., and Bischof, H. Incremental surface extraction from sparse structure-from-motion point clouds. In *Proc. BMVC* (Bristol, UK, 2013).
17. Huang, W., and Alem, L. HandsinAir: A wearable system for remote collaboration on physical tasks. In *Proc. ACM CSCW* (2013), 153–156.
18. Huang, W., Alem, L., and Tecchia, F. HandsIn3D: Augmenting the shared 3D visual space with unmediated hand gestures. In *SIGGRAPH Asia 2013 Emerging Technologies* (2013), 10:1–10:3.
19. Jankowski, J., and Hachet, M. A survey of interaction techniques for interactive 3D environments. In *Eurographics – STAR* (2013), 65–93.
20. Jo, H., and Hwang, S. Chili: Viewpoint control and on-video drawing for mobile video calls. In *ACM CHI Extended Abstracts* (2013), 1425–1430.
21. Kim, S., Lee, G. A., Sakata, N., Vartiainen, E., and Billinghurst, M. Comparing pointing and drawing for remote collaboration. In *Proc. IEEE ISMAR Extended Abstracts* (2013), 1–6.
22. Kirk, D., and Stanton Fraser, D. Comparing remote gesture technologies for supporting collaborative physical tasks. In *Proc. ACM CHI* (2006), 1191–1200.
23. Kurata, T., Sakata, N., Kourogi, M., Kuzuoka, H., and Billinghurst, M. Remote collaboration using a shoulder-worn active camera/laser. In *Proc. ISWC* (2004), 62–69.
24. Kuzuoka, H., Oyama, S., Yamazaki, K., Suzuki, K., and Mitsuishi, M. GestureMan: A mobile robot that embodies a remote instructor's actions. In *Proc. ACM CSCW* (2000), 155–162.
25. Lanir, J., Stone, R., Cohen, B., and Gurevich, P. Ownership and control of point of view in remote assistance. In *Proc. ACM CHI* (2013), 2243–2252.
26. Lee, T., and Höllerer, T. Viewpoint stabilization for live collaborative video augmentations. In *Proc. IEEE and ACM ISMAR* (Washington, DC, USA, 2006), 241–242.
27. Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int'l Journal of Computer Vision 60*, 2 (2004), 91–110.
28. Maimone, A., Yang, X., Dierk, N., State, A., Dou, M., and Fuchs, H. General-purpose telepresence with head-worn optical see-through displays and projector-based lighting. In *Proc. IEEE Virtual Reality 2013* (Orlando, USA, March 2013).
29. Mayol, W. W., Davison, A. J., Tordoff, B. J., Molton, N. D., and Murray, D. W. Interaction between hand and wearable camera in 2d and 3d environments. In *Proc. BMVC* (Sept. 2004).
30. Muja, M., and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. VISSAPP* (2009), 331–340.
31. Mulloni, A., Ramachandran, M., Reitmayr, G., Wagner, D., Grasset, R., and Diaz, S. User friendly SLAM initialization. In *Proc. IEEE ISMAR* (2013), 153–162.
32. Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (2011).
33. Oda, O., Sukan, M., Feiner, S., and Tversky, B. Poster: 3D referencing for remote task assistance in augmented reality. In *Proc. IEEE 3DUI* (March 2013), 179–180.
34. Ou, J., Fussell, S. R., Chen, X., Setlock, L. D., and Yang, J. Gestural communication over video stream: supporting multimodal interaction for remote collaborative physical tasks. In *Proc. ICMI* (2003), 242–249.
35. Pradeep, V., Rhemann, C., Izadi, S., Zach, C., Bleyer, M., and Bathiche, S. Monofusion: Real-time 3D reconstruction of small scenes with a single web camera. In *IEEE ISMAR* (2013), 83–88.
36. Razzaque, S. *Redirected Walking*. PhD thesis, University of North Carolina at Chapel Hill, 2005.
37. Sadagic, A., Towles, H., Holden, L., Daniilidis, K., and Zeleznik, B. Tele-immersion portal: Towards an ultimate systhesis. In *Proc. 4th Annual Intl. Workshop on Presence of Computer Graphics and Computer Vision Systems* (2001).
38. Snavely, N., Seitz, S., and Szeliski, R. Photo tourism: exploring photo collections in 3D. *ACM Trans. on Graphics 25*, 3 (2006), 835–846.
39. Sodhi, R. S., Jones, B. R., Forsyth, D., Bailey, B. P., and Maciocci, G. BeThere: 3D mobile collaboration with spatial input. In *Proc. ACM CHI* (2013), 179–188.
40. Sukan, M., Feiner, S., Tversky, B., and Energin, S. Quick viewpoint switching for manipulating virtual objects in hand-held augmented reality using stored snapshots. In *Proc. IEEE ISMAR* (2012).
41. Tang, J. C., and Minneman, S. L. VideoDraw: a video interface for collaborative drawing. *ACM Trans. Inf. Syst. 9* (1991), 170–184.
42. Tatzgern, M., Grasset, R., Kalkofen, D., and Schmalstieg, D. Transitional augmented reality navigation for live captured scenes. In *Proc. IEEE Virtual Reality* (Minnesota, USA, 2014).
43. Uyttendaele, M., Criminisi, A., Kang, S. B., Winder, S., Szeliski, R., and Hartley, R. Image-based interactive exploration of real-world environments. *IEEE Comput. Graph. Appl. 24*, 3 (May 2004), 52–63.
44. Wellner, P., and Freeman, S. The DoubleDigitalDesk: Shared editing of paper documents. Tech. Rep. EPC-93-108, EuroPARC, 1993.