In Touch with the Remote World: Remote Collaboration with Augmented Reality Drawings and Virtual Navigation

Steffen Gauglitz* Benjamin Nuernberger† Matthew Turk‡ Tobias Höllerer§
Department of Computer Sciene, University of California, Santa Barbara
Santa Barbara, CA 93106, USA

Abstract

Augmented reality annotations and virtual scene navigation add new dimensions to remote collaboration. In this paper, we present a touchscreen interface for creating freehand drawings as world-stabilized annotations and for virtually navigating a scene reconstructed live in 3D, all in the context of live remote collaboration. Two main focuses of this work are (1) automatically inferring depth for 2D drawings in 3D space, for which we evaluate four possible alternatives, and (2) gesture-based virtual navigation designed specifically to incorporate constraints arising from partially modeled remote scenes. We evaluate these elements via qualitative user studies, which in addition provide insights regarding the design of individual visual feedback elements and the need to visualize the direction of drawings.

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

Keywords: CSCW; video-mediated communication; augmented reality; telepresence; touch; gesture recognition; depth interpretation

1 Introduction

Advances in computer vision and human-computer interaction paradigms such as augmented reality (AR) offer the chance to make remote collaboration significantly more immersive and thus to broaden its applicability. Specifically, integrating them allows remote users to explore remote scenes based on collected imagery [Uyttendaele et al. 2004; Snavely et al. 2006] as well as to communicate spatial information (e.g., referencing objects, locations, directions) via annotations anchored in the real world [Jo and Hwang 2013; Sodhi et al. 2013].

In [Gauglitz et al. 2014], we presented a fully operational, mobile system to implement this idea. The remote user's system received the live video and computer vision-based tracking information from a local user's smartphone or tablet and modeled the environment in 3D based on this data. It enabled the remote user to navigate the scene and to create annotations in it which were then sent back and visualized to the local user in AR. This earlier work focused on the overall system and enabling the navigation and communication within this framework. However, the remote user's interface was rather simple: most notably, it used a standard mouse for most interactions, supplemented by keyboard shortcuts for certain functions, and supported only single-point-based markers.

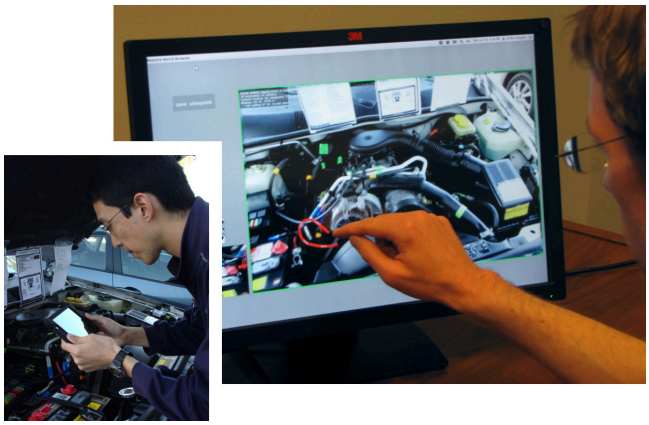


Figure 1: *A remote user points out an element in the environment (here: a car's engine bay) by drawing an outline around it. The annotation is world-stabilized and displayed in augmented reality to the local user (bottom left), who is holding a tablet.*

In this paper, we introduce a novel interface for the remote user in the context of this previous system that allows for more expressive, direct, and arguably more intuitive interaction with the remote world (cf. Figure 1). Several elements of our interface are fundamentally different from similar existing work; we evaluate these novel elements of this interface via qualitative user studies. Specifically, our contributions in this paper include:

- A novel touchscreen-based interface for live augmented reality based remote collaboration (Section 5);
- The integration of 2D drawings as world-stabilized annotations in 3D, including an evaluation of how to interpret (i.e., unproject) the two-dimensional drawings in three-dimensional space (Section 6);
- A multitouch gesture-based virtual navigation interface designed specifically to explore partially modeled remote scenes based on a skeleton of keyframes, including multitouch orbiting with "snap" and a hybrid approach to zoom, which combines changing the field of view and dollying (Section 7).

2 Related Work

Classic videoconferencing or telepresence systems lack the ability to interact with the remote physical environment. Researchers have explored various methods to support spatial references to the remote scene, including pointers [Bauer et al. 1999; Fussell et al. 2004; Kim et al. 2013], hand gestures [Kirk and Fraser 2006; Huang and Alem 2013; Oda et al. 2013], and drawings [Ou et al. 2003;

*e-mail: sgauglitz@cs.ucsb.edu
†e-mail: bnuernberger@cs.ucsb.edu
‡e-mail: mturk@cs.ucsb.edu
§e-mail: holl@cs.ucsb.edu

Fussell et al. 2004; Kirk and Fraser 2006; Chen et al. 2013; Kim et al. 2013]. Most notably in our context, Ou et al. [2003] investigated the use of drawings for collaboration including recognition of common shapes (for regularization, compression, and interpretation as commands).

However, in all of these works, the remote user's view onto the scene is constrained to the current view of the local user's camera, and the support for spatially referencing the scene is contingent upon a stationary camera.

More specifically, when drawings are used, they are created on a 2D surface as well as displayed in a 2D space (e.g., a live video), and it remains up to the user to mentally "unproject" them and interpret them in 3D. This is fundamentally different from creating annotations that are anchored and displayed in 3D space, that is, in AR.

Very recently, a few systems have been presented that support world-stabilized annotations [Gauglitz et al. 2012; Jo and Hwang 2013; Sodhi et al. 2013; Gauglitz et al. 2014]. Of those, two [Gauglitz et al. 2012; Gauglitz et al. 2014] support only single-point-based markers, and two [Gauglitz et al. 2012; Jo and Hwang 2013] can cope with 2D/panorama scenes only. The system by Sodhi et al. [2013] uses active depth sensors on both the local and the remote user's side and thus supports transmission of hand gestures in 3D (along with a different approach to navigating the remote scene); we will contrast their approach with ours in more detail in Section 5.1.

In other areas such as computer-aided design or interactive image-based modeling, the interpretation of 2D drawings in 3D is commonly used [Tolba et al. 1999; Igarashi et al. 2007; Zeleznik et al. 2007; Xin et al. 2008; van den Hengel et al. 2007]. However, the purpose (design/modeling vs. communication), intended recipient (computer vs. human collaborator) and, in most cases, scene (virtual model vs. physical scene) all differ fundamentally from our application, and the interpretation of 2D input is typically guided and constrained by task/domain knowledge. Thus, these techniques cannot immediately be applied here.

To our knowledge, freehand 2D drawings have not been used before as world-stabilized annotations unprojected into 3D space for live collaboration. While freehand drawings have been used to create annotations in AR (e.g., [Kasahara et al. 2012]), the inherent ambiguity due to different depth interpretations has not been addressed explicitly in this context.

With respect to virtual navigation, there is a large body of work concerning 3D navigation from 2D inputs [Hanson and Wernert 1997; Zeleznik and Forsberg 1999; Tan et al. 2001; Hachet et al. 2008; Christie and Olivier 2009; Jankowski and Hachet 2013], including works specifically designed for multitouch interaction (e.g., [Marchal et al. 2013]). The motivation in this paper is not to propose a novel multitouch interaction to compete with those works, but rather to describe an appropriate solution given the particular constrained situation — that is, a partially modeled scene with highest rendering fidelity from a set of known viewpoints.

Our work is further related to the virtual exploration by Snavely et al. [2008]; we will discuss this relationship further in Section 7.

3 System Overview

We first give a brief overview of the collaborative system as a whole, as illustrated in Figure 2. The local user uses a lightweight tablet or smartphone. It runs a vision-based simultaneous localization and mapping (SLAM) system and sends the tracked camera pose along with the encoded live video stream to the remote system. From there, the local system receives annotations and displays them, overlaid onto the live video, to the local user; i.e., it acts as a classic magic lens. This system was implemented as an Android app, running on several state-of-the-art Android devices.

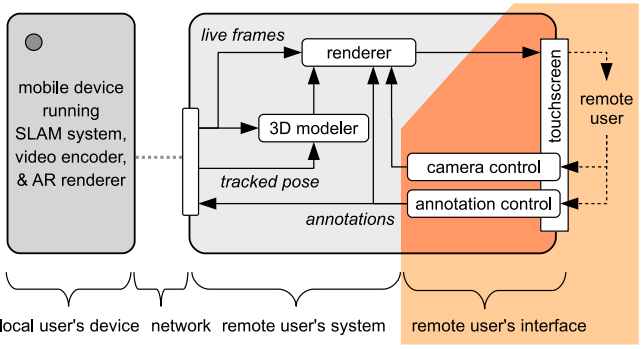


Figure 2: *Overview of the collaborative system including local user's system (left) and remote user's system (right). In this paper, we focus on the remote user's interface (highlighted area).*

The remote user's system, running on a commodity PC, receives the live video stream and the associated camera poses and models the environment in real time from this data. The remote user can set world-anchored annotations which are sent back and displayed to the local user. Further, thanks to the constructed 3D model, he/she can move away from the live view and choose to look at the scene from a different viewpoint via a set of virtual navigation controls. Transitions from one viewpoint to another (including the live view) are rendered seamlessly via image based rendering techniques using the 3D model, cached keyframes, and the live frame appropriately.

For an in-depth description of the whole system we refer the reader to [Gauglitz et al. 2014]. In the system described therein, the remote user used a standard PC interface, using the mouse and keyboard shortcuts for annotation control and virtual navigation. While the system as a whole was received very favorably (for example, 80% of the users preferred it over the two alternative interfaces, cf. [Gauglitz et al. 2014]), individual elements of the remote user's interface in particular were found to be suboptimal.

Thus, in this paper, we concentrate on the remote user's interaction with the system and present and evaluate a new interface for it (highlighted area in Figure 2).

4 User Evaluation & Feedback

The feedback from users who participated in our task performance-based user study reported in [Gauglitz et al. 2014] served as motivation and the initial source of feedback for this work. In that study, 30 pairs of participants used the prior system, as well as two alternative interfaces for comparison, to solve a particular collaborative task and subsequently rated the interfaces and provided comments. We will refer to these users as group 1.

We then incorporated novel elements and iterated the design as described in the following sections. The system was demonstrated during a three-hour open house event, where roughly 25 visitors (referred to as group 2) directly interacted with the system. Although these interactions were very brief and unstructured, we were able to observe how people intuitively used the system and which features appeared to work well.

Lastly, we asked eleven users to interact with the system following a structured protocol and provide feedback on particular design elements (group 3). They were compensated for their time commitment of about 50 minutes with US$10. Of these study participants, five were female; the age range was 18–22 years. Four stated that they were "somewhat familiar" with interactive 3D software (e.g., 3D modelers). All had used small-scale touchscreens (e.g., smartphones) on a daily basis, but only one had used large-scale touchscreens more than "occasionally to rarely." Five of the participants had participated in the earlier task performance-based study and were thus also asked to comment on the differences compared to the earlier design.

We will report on this user feedback in the respective design sections below.

5 The Touchscreen Interface

The remote user uses a touchscreen interface for all interactions with the system. In this section, we describe the main elements of the graphical user interface before focusing on two particular aspects — namely, the use of 2D drawings as world-stabilized annotations in 3D and gesture-based virtual navigation — in Sections 6 and 7, respectively.

We encourage the reader to watch the supplemental video, in which each feature of the interface is demonstrated.

5.1 Motivation for using a touchscreen vs. 3D input

One might argue that, for interaction in a three-dimensional space, one should use an interface which affords three-dimensional input and thus can, for example, create annotations in 3D. Sodhi et al. [2013] described a prototype system which uses three-dimensional input for the purpose of remote collaboration, by reconstructing the remote user's hand in 3D and transferring this reconstruction and a 3D "motion trail" into the local user's space.

However, even if sensors that support unthethered, unobtrusive 3D input (e.g., high resolution active depth sensors) become commonplace, additional issues remain. First, unless such sensors are matched with an immersive 3D display that can synthesize images in any physical space (such as a stereo head-worn display), the space in which the input is provided and the space in which objects are visualized remain separate, in much the same way as is the case with a standard computer mouse. This has the effect that *relative* spatial operations are very natural and intuitive (e.g., moving the mouse cursor downwards/indicating a direction in 3D, respectively), but *absolute* spatial operations (e.g., pointing to an object, which is arguably very important in our context) remain indirect and require the user to first locate the mouse cursor/representation of the hand, respectively, and position it with respect to the object of interest. This issue can be well observed in the illustrations and discussion by Sodhi et al. [2013].

Second, even with such an immersive 3D display, haptic feedback is typically missing [Xin et al. 2008].

In contrast, touchscreens are not only ubiquitous today, but they afford direct interaction without the need for an intermediate representation (i.e., a mouse cursor), and provide haptic feedback during the touch. In our context, however, they have the downside of providing 2D input only. Discussing the implications of this limitation and describing and evaluating appropriate solutions in the context of live remote collaboration is one of the main contributions of this paper.

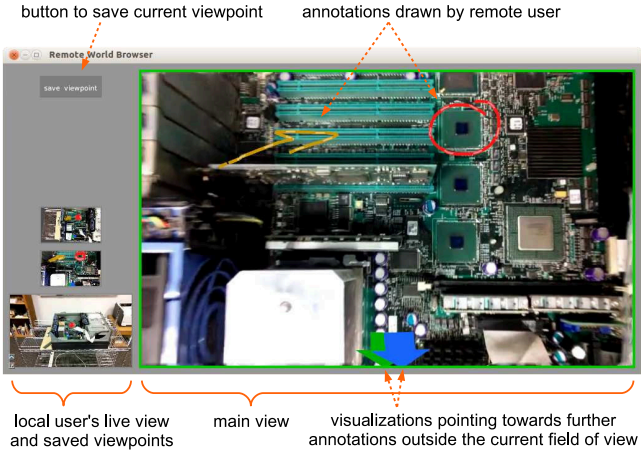


Figure 3: *Screenshot of the remote user's touchscreen interface*

5.2 Interface elements

Figure 3 presents the elements of the graphical user interface. The main part of the screen shows the main view, in which annotations can be created (Section 6) and gesture-based virtual navigation takes place (Section 7).

A side pane contains, from top to bottom, (1) a button to save the current viewpoint, (2) small live views of saved viewpoints, and (3) the local user's live view (whenever the main view is not identical to it). A tap onto any of the views in the side pane causes the main view to transition to that respective viewpoint.

A two-finger tap onto the main view while it is coupled to the local user's live view freezes the current viewpoint. (Note that only the *viewpoint* is frozen; the live image is still projected into the view [Gauglitz et al. 2014].)

When the user starts to draw an annotation while the main view is coupled to the (potentially moving) live view, the viewpoint is temporarily frozen in order to enable accurate drawing. In this case, the view automatically transitions back to the live view as soon as the finger is lifted. This feature was particularly well-received by group 2.

Thus, all interface functions are immediately accessible on the screen, allowing quick access and avoiding the need to memorize keyboard shortcuts (which was noted as a drawback in the previous system by group 1).

6 2D Drawings as Annotations in 3D Space

Using a single finger, the remote user can draw annotations into the scene. A few examples are shown in Figure 4(a). Since the camera is tracked with respect to the scene, these annotations automatically obtain a position in world coordinates. However, due to our use of a touchscreen, the depth of the drawing along the current viewpoint's optical axis is unspecified.

In principle, it is possible to ask the user to explicitly provide depth, for example by providing a second view onto the scene and having the user shift points along the unspecified dimension. This may be appropriate for professional tools such as CAD. However, in our case, we want to enable the user to quickly and effortlessly communicate spatial information, such as with hand gestures in face-to-face communication. Thus, we concentrate on ways to infer depth

automatically. In this section, we discuss and evaluate several alternatives to do so.

6.1 Depth interpretations

We start with the assumption that the annotation shall be in contact with the 3D surface in some way; i.e., annotations floating in mid-air are, for now, not supported.

Given an individual 2D input location $p = (x, y)$, a depth d can thus be obtained by un-projecting p onto the model of the scene. For a sequence of 2D inputs (a 2D drawing) $p_1, ..., p_n$, this results in several alternatives to interpret the depth of the drawing as a whole, of which we consider the following:

- **"Spray paint."** Each sample p_i gets assigned its own depth d_i independently; the annotation is thus created directly on the 3D surface as if spray painted onto it (Figure 4(b)).
- **Plane orthogonal to viewing direction.** The annotation is created on a plane orthogonal to the viewing direction. Its depth can be set to any statistic of $\{d_i\}$, for example, the minimum (to ensure that no part of the annotation lands behind surfaces) (Figure 4(c)) or the median (Figure 4(d)).
- **Dominant surface plane.** Using a robust estimation algorithm such as RANSAC or Least Median of Squares, one can estimate the dominant plane of the 3D points formed by $\{p_i\}$ and the associated $\{d_i\}$ and project the drawn shape onto this plane (Figure 4(e)).

All of these approaches appear to have merits; which one is most suitable depends on the context and purpose of the drawing. Spray paint appears to be the logical choice if the user intends to "draw" or "write onto" the scene, trace specific features, etc.[1]

For other types of annotations, planarity may be preferable. To refer to an object in its entirety, the user might draw an outline *around* the object. Drawings used as proxies for gestures — for example, drawing an arrow to indicate a direction, orientation, or rotation (cf. Figure 4 top two rows) — are likely more easily understood if projected onto a plane.

Another aspect for consideration, especially in the case of models reconstructed via computer vision, is the sensitivity to noise and artifacts in the model. A spray-painted annotation may get unintelligibly deformed and the minimum depth plane may be shifted. The median depth and dominant plane are more robust as single depth measurements carry less importance.

6.2 Evaluation & discussion

In order to test our hypotheses above, we asked the users from group 3 to communicate a piece of information to a hypothetical partner via a drawing. For example (for Figure 4(a) top to bottom): "In which direction do you have to turn the knob?"; "Where should the microwave be placed?"; "Where is the motor block?". We then showed the scene with the drawn annotation from a different viewpoint and asked the users which of the four depth interpretations (Figure 4(b-e)) was the most suitable interpretation for their particular drawing.

[1] Note that projective displays — used for remote collaboration for example by Gurevich et al. [2012] and appealing due to their direct, un-mediated overlay of annotations — can intrinsically only produce spray paint-like effects, unless not only the projector, but also the user's eyes are tracked and active stereo glasses (synced with the projector) are used in order to create the illusion of a different depth.

arrow head	none	attached	detached
example			
initially:	24.6%	23.1%	52.3%
with animation:	72.3%	15.4%	12.3%

Table 1: *Usage of different styles of arrow heads initially and after an animation visualizing the drawings' direction was introduced.*

We covered a range of different questions to prompt a variety of drawings. We also used scenes in which the original viewpoint was roughly perpendicular to the object's main surface (Figure 4 top) as well as slanted surfaces (Figure 4 middle and bottom row). Further, to be able to distinguish conceptual issues from sensitivity to noise, we used both virtual models (Figure 4 top and middle row) and models created by our system via SLAM and approximate surface modeling (Figure 4 bottom). We avoided situations in which all variants result in the same or nearly the same shape, that is, single planar surfaces. In total, we asked each user for 27 drawings, in four different environments.

We emphasize that we did not tell the users *how* they should communicate the information, and thus the drawn shapes varied appreciably in nature. For example, to refer to a particular object, some users traced the object's outline (similar to Figure 4 bottom), some circled it loosely, and others drew an arrow pointing towards it.

Arrow heads and indicating direction. Users used arrow-like drawings in several contexts, for example, to indicate a direction of rotation (e.g., Figure 4 top). However, the type of arrow head varied, as detailed in Table 1. Initially, roughly one quarter of all arrows were drawn without head; that is, the shape itself does not actually convey the direction. Users evidently assumed that the direction of the drawing would implicitly be communicated. As the drawings assume the role of gestures (and one would rarely add an arrow head with a hand gesture motioned in mid-air), this assumption is reasonable.

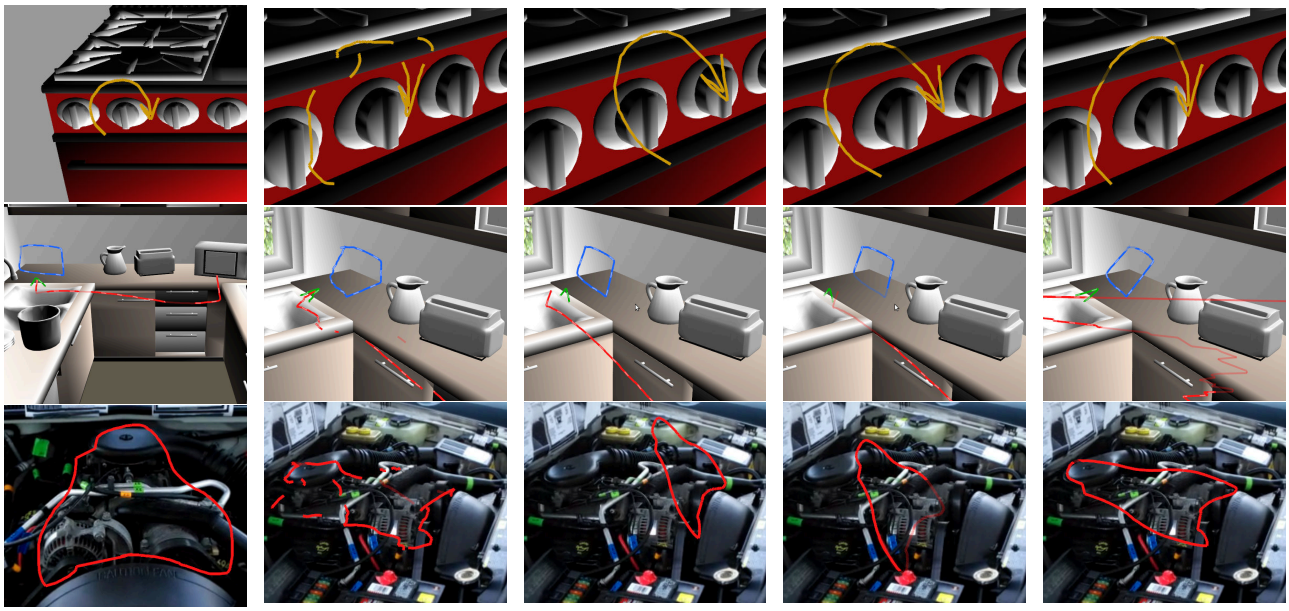
We had anticipated this variant due to prior observations (e.g., with users from group 2) and thus implemented an animated visualization to do so: here, the line contains lighter dashes which flow in the direction in which the line was drawn.[2] Thus far, we use a fixed speed of flow, but one could extend this visualization by additionally visualizing the speed of drawing via the speed of flow.

We introduced this animation after the first six drawings for each user; after that, almost three quarter of all arrows were drawn without head (cf. Table 1). The animation was rated as "very helpful" or "helpful" by 10 of 11 users; nobody perceived it as distracting (even though it was not necessary for some of the drawings) (cf. Figure 6).

User preference among depth interpretations. Figure 5 details the users' preference among the depth interpretations, broken down by various subsets. Overall, in every category, users tended to prefer the planar projections in general, and the dominant plane version in particular, which was the most frequently chosen variant in every category but one.

Limitations. One limitation of our current implementation is that drawings are treated as separate annotations as soon as the finger is

[2] The design was inspired by the visual appearance of animated Line Integral Convolution [Cabral and Leedom 1993] and can loosely be thought of as a coarse approximation for it for a single dimension and constant speed.



(a) original view (b) spray paint (c) minimum depth (d) median depth (e) dominant surface plane

Figure 4: *Depth interpretations of 2D drawings. In each row, (a) shows the viewpoint from which the drawing was created, and (b-e) show different depth interpretations from a second viewpoint (all interpretations have the same shape if seen from the original viewpoint). Annotation segments that fall behind object surfaces are displayed semi-transparently, which was deemed "very helpful" by 7 of 11 users (cf. Figure 6).*

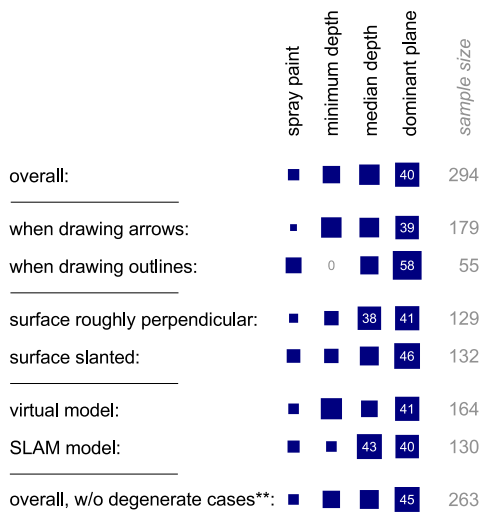


Figure 5: *Users' preference among the four different depth interpretations, broken down by various subsets. Per each row, the areas of the squares and the numbers indicate the preference for a particular variant in percent.*

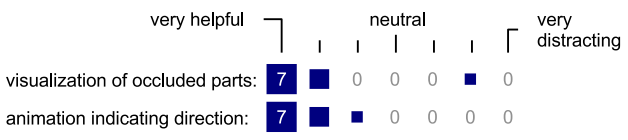


Figure 6: *Ratings of two visualization options (# of users).*

lifted, which means that they get moved onto separate planes by the three planar projection methods (cf. Figure 4(c-e) middle row). Individual segments that are created in very close succession — such as a detached head for an arrow, which was by far the most commonly used style of arrow before we introduced the animation indicating direction (cf. Table 1) — should ideally be treated as one entity.

Further, the "dominant plane" method is susceptible to extreme artifacts if the points lie on or close to a single line, such as seen in Figure 4(e) middle row. When removing the cases in which this occurred from consideration, the general preference for the dominant plane version increases further (last row in Figure 5). In future work, these degenerate configurations should be detected and one of the other variants (e.g., median depth) used instead.

Naturally, there are several cases in which none of these options will work satisfactorily, such as when trying to create an annotation in mid-air or behind physical surfaces. Of course, an interface may also offer to change the type of depth inference used, e.g., for advanced users. Even for 2D only, some interfaces offer a whole array of drawing tools [Gurevich et al. 2012]; however, it has been reported that users use freehand drawing most often [Gurevich et al. 2012]. With this consideration, the quest here is to find out what the default setting should be, such that gestures used in face-to-face communication can be emulated as efficiently as possible.

7 Gesture-based Virtual Navigation

As discussed in Section 5, the user can freeze the viewpoint (Figure 7(a)), save the current viewpoint, and go to the live view or any previously saved view (Figure 7(b)), all with a single tap onto the respective region in the interface. In addition, we implemented gesture-based navigation controls (Figure 7(c)–(e)) which we will discuss in this section. As a distinguishing element from drawing

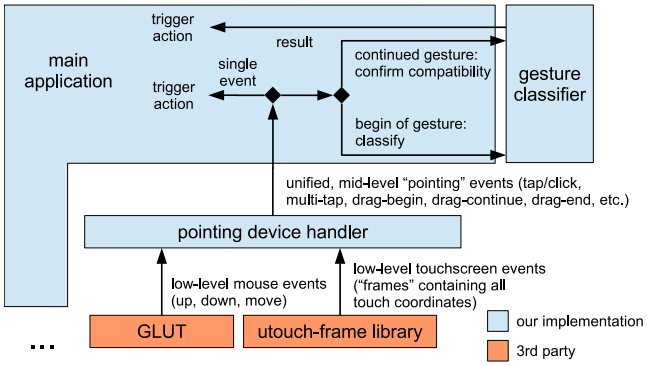


Figure 8: *Software stack for processing events from pointing devices (mouse & touchscreen).*

annotations (Section 6), all navigation-based controls on the main view are triggered by *two* fingers.

While we want to empower the remote user to explore the scene as freely as possible, only parts of the scene that have previously been observed by the local user's camera are known and can be rendered. From the incoming live video, we store keyframes based on a set of heuristics [Gauglitz et al. 2014] which are used as a kind of "skeleton" for the navigation.

Thus, our navigation shares an important characteristic with the work by Snavely et al. [2008]: from a more or less sparse set of camera poses we want to find paths/transitions that can be mapped to specific input controls. However, in contrast to their work, we do not attempt to mine longer paths and suggest them to the user, but rather to find suitable transitions given a user's specific input.

For all gestures, we designed the controls such that the 3D scene points underneath the touching fingers follow (i.e., stay underneath) those fingers throughout the gesture (i.e., "contact trajectories match the scene transformations caused by viewpoint modifications" [Marchal et al. 2013]) as far as possible.

The software stack that we implemented to process the touchscreen and mouse input is depicted in Figure 8. A pointing device handler receives low-level mouse and touchscreen events (from GLUT and the Ubuntu utouch-frame library[3], respectively) and generates events that are unified across the devices and of slightly higher level (e.g., recognize "click"/"tap" from down+*(no move)*+up events, distinguish from drag-begin, etc.). These events are received by the main application. Multi-stage events (i.e., gestures) are sent on to a gesture classifier which classifies them or, after successful classification, validates the compatibility of the continuing gesture. The gesture classifier operates on relatively simple geometric rules (e.g., for swipe, the touch trails have to be of roughly the same length and roughly parallel), which worked sufficiently for our purposes, as our qualitative evaluation (cf. Section 7.4) confirmed.

7.1 Panning

By moving two fingers in parallel, the user can pan, i.e., rotate the virtual camera around its optical center (Figure 7(c)).[4] We maintain the original up vector (typically: gravity, as reported by the local user's device) by keeping track of the original camera position,

[3] Open Input Framework Frame Library, https://launchpad.net/frame

[4] Pan is the only control among the three reported here which, apart from using a touchscreen swipe instead of a mouse drag, was present in the same form in our earlier system [Gauglitz et al. 2014].

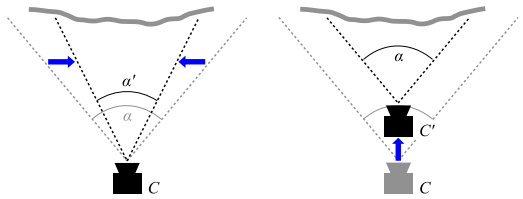


Figure 9: *"Zooming in" via decreasing the field of view (left) and dollying forward (right).*

accumulating increments for yaw and pitch separately and applying *one* final rotation for each yaw and pitch (rather than a growing sequence of rotations, which would skew the up vector).

As we will typically have imagery of part of the remote scene only, we constrain the panning to the angular extent of the known part of the scene. 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 swiping causes an exponentially declining increase in rotation and visual feedback in the form of an increasingly intense blue gradient along the respective screen border (see Figure 3 in [Gauglitz et al. 2014] and supplemental video). If the fingers are lifted off the screen at this moment, the panning snaps back to the allowed range.

7.2 Transitional zoom

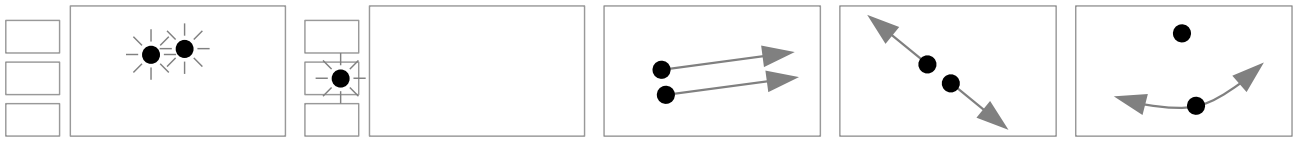
There are two different ways of implementing the notion of "zoom" in 3D: either via modifying the field of view (FOV) (Figure 9 left) or via dollying (i.e., moving the camera forward/backward; Figure 9 right). For objects at a given depth, both changes are equivalent; differences arise due to varying depths (i.e., parallax). Which motion is desired by the user may depend on the particular scene. Given the large overlap in effect (cf. the optical flow diagrams by Marchal et al. [2013]), we wanted to avoid providing two separate controls.

In our context, with an incomplete and/or imperfect model of the environment, changing the FOV has the advantage that it is trivial to render, while rendering a view correctly after dollying the camera might be impossible due to occlusions. However, changing the FOV disallows exploration of the scene beyond a fixed viewpoint, and its usefulness is limited to a certain range by the resolution of the image.

Here, we thus propose a novel hybrid approach: changing the FOV to allow for smooth, artifact-free, fine-grained control combined with transitioning to a suitable keyframe, if available, in front or behind the current location for continued navigation. We note that the availability of a keyframe automatically implies that it may be reasonable to move there (e.g., walk in this direction), while free dollying has to be constrained intelligently to not let the user fly through the physical surface when his/her intent may have been to get a close-up view. We implemented this hybrid solution as follows.

Given a two-finger "pinch" gesture (Figure 7(d)) with start points $s_{1,2}$ and end points $e_{1,2}$, we first calculate the change in FOV that corresponds to the change in distance $|s_1 - s_2|$ to $|e_1 - e_2|$. We then adapt yaw and pitch such that the scene point under $(s_1 + s_2)/2$ gets mapped to $(e_1 + e_2)/2$. (We disallow roll on purpose as it is rarely used or needed, cf. Marchal et al. [2013].)

To determine if transitioning to a different keyframe is in order, we determine the 3D world points $\{c_i^{3D}\}_{i=1..4}$ which, with the mod-



(a) two-finger tap on main view ⇒ freeze/go to point of interest
(b) tap on thumbnail view ⇒ go to respective viewpoint
(c) two-finger swipe ⇒ pan camera
(d) two-finger pinch ⇒ zoom
(e) orbit ⇒ orbit around point

Figure 7: *Navigation features.*

ified projection matrix, get mapped into the corners of the view $\{c_i\}$. The ideal camera pose R is the one that projects $\{c_i^{3D}\}$ to $\{c_i\}$ with its *native* projection matrix P (i.e., unmodified FOV). Therefore, we project $\{c_i\}$ using the camera pose of each of the stored keyframes, and select the camera k^* for which the points land closest to the image corners $\{c_i\}$; i.e.,

$$k^* = \arg\min_k \sum_{i=1}^{4} \left| P \cdot R_k \cdot c_i^{3D} - c_i \right|^2 \quad (1)$$

To ensure that the transition is consistent with the notion of dollying, we only consider cameras whose optical axis is roughly parallel to the current camera's optical axis.

If k^* is not identical to the current camera, we thus transition to R_{k^*} and adapt FOV, yaw and pitch again as described above. A hysteresis threshold can be used to avoid flip-flopping between two cameras with nearly identical score according to Equation (1).

In effect, the user can "zoom" through the scene as far as covered by available imagery via a single, fluid control, and the system automatically choses camera positions and view parameters (FOV, yaw, pitch) based on the available data and the user's input.

7.3 Orbiting with snap-to-keyframe

As a third gesture, we added orbiting around a given world point p^{3D}. The corresponding gesture is to keep one finger (relatively) static at a point p and move the second finger in an arc around it (Figure 7(e)). The orbit center p^{3D} is determined by un-projecting p onto the scene and remains fixed during the movement; additionally, we maintain the gravity vector (as reported by the local user's device). The rotation around the gravity vector at p^{3D} is then specified by the movement of the second finger.

Again, we want to guide the user to keyframe positions if possible as the rendering from those positions naturally has the highest fidelity. Thus, once the user finishes the orbit, the camera "snaps" to the closest keyframe camera pose. (The idea of orbiting constrained to keyframes again resembles the approach by Snavely et al. [2008], where possible orbit paths are mined from the set of pictures and then suggested to the user for exploration of the scene, but differs in that we do not find paths beforehand, but find a suitable transition given the user's input.)

Selection of the "snap" target pose. Given the list of available camera positions, we first filter out all cameras for which p^{3D} is not within the field of view. Among the remaining poses $\{R_k\}$, we select the one closest to the camera pose R_{input} at which the user ended the orbit:

$$k^* = \arg\min_k \; d(R_k, R_{\text{input}}) \quad (2)$$

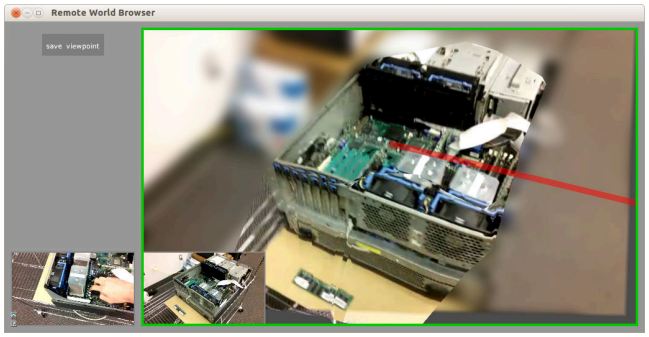


Figure 10: *Screenshot during an orbit operation, with the two visualizations of the "snap" position: 1. the inset image in the bottom left corner of the main view previews the target image; 2. the red line connects the orbit point and the target camera origin.*

where the distance $d(\cdot,\cdot)$ between two camera poses is defined as follows:

$$d(R_1, R_2) = \begin{cases} \frac{d_t(R_1,R_2)}{d_\angle(R_1,R_2)} & \text{if } d_\angle(R_1,R_2) > 0 \\ \infty & \text{otherwise} \end{cases} \quad (3)$$

where $d_t(\cdot,\cdot)$ is the translational distance between the camera origins, and $d_\angle(\cdot,\cdot)$ is the dot product of the optical axes.

Preview visualization "snap" target. During an on-going orbit process, we provide two visualizations that indicate where the camera would snap to if the orbit ended at the current location (see Figure 10): First, we display the would-be target keyframe as a preview in a small inset in the bottom left corner of the main view. Second, we display a semi-transparent red line from the orbit point p^{3D} to the would-be target camera origin. While less expressive than the preview image, this visualization has the advantage of being in-situ: the user does not have to take their eyes off the model that he/she is rotating. We opted for a minimal visualization (instead of, for example, visualizing a three-dimensional camera frustum) in order to convey the essential information but keep visual clutter to a minimum.

7.4 Qualitative evaluation & discussion

We asked the users from group 3 to try out the individual navigation controls and rate them. The results are aggregated in Figure 11.

After having been told only that two-finger gestures would control the camera (since one-finger gestures draw annotations), and asked to pan, all users intuitively guessed the gesture (i.e., swipe) correctly — unsurprisingly, perhaps, given their exposure to touch-screen interfaces — and were able to control the camera as suggested; the control received high ratings in terms of both intuitiveness and ease of use. Additionally, all users correctly identified

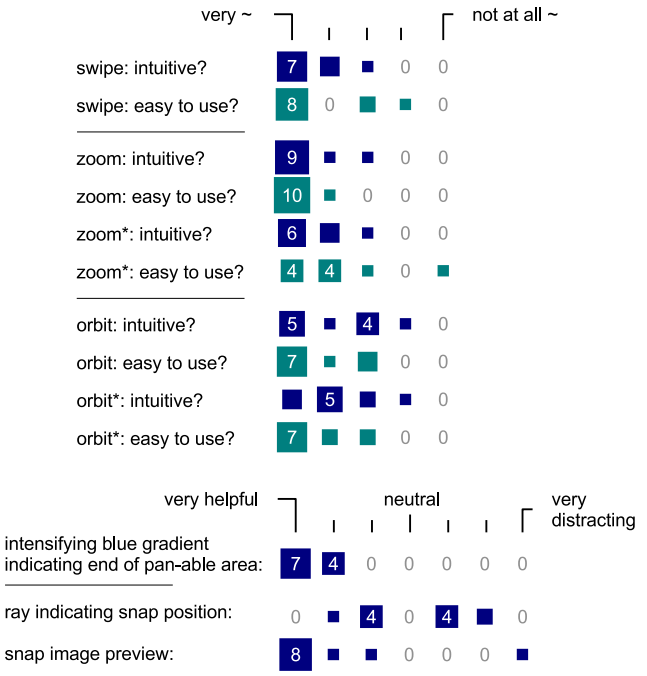


Figure 11: *Ratings of various navigation elements (areas of squares are proportional to # of users).*

what the intensifying blue gradient communicated; this visualization was rated as "very helpful" by 7 of 11 users, and as "helpful" by the others.

For zooming, again all users intuitively guessed the gesture (i.e., pinch) correctly and were able to control the camera as suggested. We let them try out zoom first with the transitional component disabled, i.e., only the FOV was changed. This control was given the highest rating on a 5-point scale for intuitiveness and ease of use by 9 and 10 out of 11 users, respectively. With transitional component enabled (labeled "zoom*" in Figure 11), the control still received high ratings, though clearly lower than without. Several users appreciated the fact that it transitioned to other frames and thus allowed to zoom further, however, the decreased smoothness and possibility of getting "stuck" were noted as downsides. The idea of using a hysteresis threshold to decrease artifacts was a result of this feedback (i.e., it was not implemented at the time of this evaluation).

For orbiting, we first introduced the control on a *virtual* model without any constraints/snap-to-keyframe. While the ratings for intuitiveness are lower than for the other methods, the ratings for ease of use are similar. With snap-to-keyframe, on a model reconstructed from images (labeled "orbit*" in Figure 11), the ratings are very similar, suggesting that the constraint was not irritating. The two visualizations received very different ratings, however: while the preview image was rated as "very helpful" by 8 of 11 users, the red line was perceived as "(somewhat or slightly) helpful" by only half the users, and as "(somewhat or slightly) distracting" by the other half. Conversations made clear that despite explanations, not all users understood and/or saw value in this visualization.

8 Conclusions

We presented a touchscreen interface via which a remote user can navigate a physical scene (which is modeled live using computer vision) and create world-stabilized annotations via freehand drawings, thus enabling more expressive, direct, and arguably more intuitive interaction with the scene than previous systems using mouse-based interfaces. Our contributions include the design and qualitative evaluation of gesture-based virtual navigation controls designed specifically for constrained navigation of partially modeled remote scenes, and the integration of freehand drawings including the analysis and evaluation of different alternatives to unproject them into the three-dimensional scene. While 2D drawings for communication have previously been explored, new challenges arise when integrating them into an immersive AR framework.

With respect to our gesture-based virtual navigation, we suggest that the consistently high ratings indicate that the controls are designed appropriately. By design, our controls are dependent on the viewpoint coverage provided by the local user; an area that deserves further investigation is how to ensure that the user cannot get "stuck" in the case of unfavorable viewpoint distributions.

With respect to the interpretation of 2D drawings in 3D, given our results, we suggest that a planar interpretation of the drawings is most likely the most useful default in the context of remote collaboration, in particular if their implementation takes into account two aspects that have been identified above (namely, segments created in close succession should be grouped, and degenerate cases for the case of dominant plane estimation should be identified and avoided).

Additionally, it has been demonstrated that it is important to visualize the direction in which the annotations are drawn.

In future work, one could combine the methods and choose, for example, the dominant plane if the fraction of supporting inliers is large and median depth plane otherwise; or attempt to recognize particular shapes [Ou et al. 2003] and use this information in interpreting their shape in 3D. We note that none of our interpretations assume semantic knowledge of any kind about the scene; if such knowledge and/or 3D segmentations are available, this may enable higher-level interpretations.

Acknowledgements

We thank Qualcomm, in particular Serafin Diaz, Daniel Wagner and Alessandro Mulloni, for providing the SLAM implementation used in this work, as well as Mathieu Rodrigue and Julia Gauglitz for their help producing the video.

This work was supported by NSF grants IIS-1219261 and IIS-0747520, and ONR grant N00014-14-1-0133.

References

BAUER, M., KORTUEM, G., AND SEGALL, Z. 1999. "Where are you pointing at?" A study of remote collaboration in a wearable videoconference system. In *Proceedings of the 3rd International Symposium on Wearable Computers (ISWC)*, 151 –158.

CABRAL, B., AND LEEDOM, L. C. 1993. Imaging vector fields using line integral convolution. In *Proceedings of the 20th ACM SIGGRAPH Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 263–270.

CHEN, S., CHEN, M., KUNZ, A., YANTAÇ, A. E., BERGMARK, M., SUNDIN, A., AND FJELD, M. 2013. SEMarbeta: Mobile sketch-gesture-video remote support for car drivers. In *Proceedings of the 4th Augmented Human International Conference*, 69–76.

CHRISTIE, M., AND OLIVIER, P. 2009. Camera control in computer graphics: Models, techniques and applications. In *ACM SIGGRAPH ASIA 2009 Courses*, 3:1–3:197.

FUSSELL, S. R., SETLOCK, L. D., YANG, J., OU, J., MAUER, E., AND KRAMER, A. D. I. 2004. Gestures over video streams to support remote collaboration on physical tasks. *Hum.-Comput. Interact. 19* (September), 273–309.

GAUGLITZ, S., LEE, C., TURK, M., AND HÖLLERER, T. 2012. Integrating the physical environment into mobile remote collaboration. In *Proceedings of the ACM SIGCHI International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*.

GAUGLITZ, S., NUERNBERGER, B., TURK, M., AND HÖLLERER, T. 2014. World-stabilized annotations and virtual scene navigation for remote collaboration. In *Proceedings of the 27th ACM Symposium on User Interfaces Software and Technology (UIST)*.

GUREVICH, P., LANIR, J., COHEN, B., AND STONE, R. 2012. TeleAdvisor: A versatile augmented reality tool for remote assistance. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, 619–622.

HACHET, M., DECLEC, F., KNODEL, S., AND GUITTON, P. 2008. Navidget for easy 3D camera positioning from 2D inputs. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, 83–89.

HANSON, A. J., AND WERNERT, E. A. 1997. Constrained 3D navigation with 2D controllers. In *Proceedings of the 8th Conference on Visualization*.

HUANG, W., AND ALEM, L. 2013. HandsInAir: A wearable system for remote collaboration on physical tasks. In *Proceedings of the Conference on Computer Supported Cooperative Work Companion (CSCW)*, 153–156.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 2007. Teddy: A sketching interface for 3D freeform design. In *ACM SIGGRAPH 2007 Courses*.

JANKOWSKI, J., AND HACHET, M. 2013. A survey of interaction techniques for interactive 3D environments. In *Proc. Eurographics–STAR*.

JO, H., AND HWANG, S. 2013. Chili: Viewpoint control and on-video drawing for mobile video calls. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, 1425–1430.

KASAHARA, S., HEUN, V., LEE, A. S., AND ISHII, H. 2012. Second surface: Multi-user spatial collaboration system based on augmented reality. In *ACM SIGGRAPH Asia 2012 Emerging Technologies*, 20:1–20:4.

KIM, S., LEE, G. A., SAKATA, N., VARTIAINEN, E., AND BILLINGHURST, M. 2013. Comparing pointing and drawing for remote collaboration. In *Proceedings of the IEEE International Symposium on Mixed and Augmeted Reality (ISMAR) Extended Abstracts*, 1–6.

KIRK, D., AND FRASER, D. S. 2006. Comparing remote gesture technologies for supporting collaborative physical tasks. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, 1191–1200.

MARCHAL, D., MOERMAN, C., CASIEZ, G., AND ROUSSEL, N. 2013. Designing intuitive multi-touch 3D navigation techniques. In *Human-Computer Interaction – INTERACT 2013*, P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler, Eds., vol. 8117 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 19–36.

ODA, O., SUKAN, M., FEINER, S., AND TVERSKY, B. 2013. Poster: 3D referencing for remote task assistance in augmented reality. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, 179–180.

OU, J., FUSSELL, S. R., CHEN, X., SETLOCK, L. D., AND YANG, J. 2003. Gestural communication over video stream: supporting multimodal interaction for remote collaborative physical tasks. In *Proceedings of the 5th International Conference on Multimodal interfaces (ICMI)*, 242–249.

SNAVELY, N., SEITZ, S., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3D. *ACM Trans. Graph. 25*, 3, 835–846.

SNAVELY, N., GARG, R., SEITZ, S. M., AND SZELISKI, R. 2008. Finding paths through the world's photos. In *Proceedings of the ACM SIGGRAPH Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 15:1–15:11.

SODHI, R. S., JONES, B. R., FORSYTH, D., BAILEY, B. P., AND MACIOCCI, G. 2013. BeThere: 3D mobile collaboration with spatial input. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 179–188.

TAN, D. S., ROBERTSON, G. G., AND CZERWINSKI, M. 2001. Exploring 3D navigation: combining speed-coupled flying with orbiting. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 418–425.

TOLBA, O., DORSEY, J., AND MCMILLAN, L. 1999. Sketching with projective 2D strokes. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 149–157.

UYTTENDAELE, M., CRIMINISI, A., KANG, S. B., WINDER, S., SZELISKI, R., AND HARTLEY, R. 2004. Image-based interactive exploration of real-world environments. *IEEE Comput. Graph. Appl. 24*, 3, 52–63.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. Videotrace: Rapid interactive scene modelling from video. *ACM Trans. Graph. 26*, 3.

XIN, M., SHARLIN, E., AND SOUSA, M. C. 2008. Napkin Sketch: Handheld mixed reality 3D sketching. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, 223–226.

ZELEZNIK, R., AND FORSBERG, A. 1999. Unicam – 2D gestural camera controls for 3D environments. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, 169–173.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 2007. SKETCH: An interface for sketching 3D scenes. In *ACM SIGGRAPH 2007 Courses*.