Multi-View Gesture Annotations in Image-based 3D Reconstructed Scenes

Benjamin Nuernberger[1*] Kuo-Chin Lien[1†] Lennon Grinta[1‡] Chris Sweeney[2§] Matthew Turk[1*] Tobias Höllerer[1*]

[1]Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106, USA

[2]Department of Computer Science
University of Washington, Seattle
Seattle, WA 98195, USA

(a) *First drawing.* **(b)** *Second viewpoint.* **(c)** *Second drawing.* **(d)** *Final 3D arrow.* **(e)** *Final 3D arrow.* **(f)** *Final 3D arrow.*

Figure 1: *Our system enables users to annotate wide-area reconstructed environments using interactive disambiguation of 2D drawing gestures. After drawing the initial annotation (a), the user is shown a second viewpoint with a semi-transparent helper ray indicating the set of 3D positions along which the annotation might be placed in 3D (b). The user draws a second annotation to disambiguate it (c), resulting in a final 3D annotation as shown from various viewpoints in (d-f). This figure shows the Piazza del Popolo in Rome, Italy (336 photos and 34,715 reconstructed 3D points) [Wilson and Snavely 2014]; original images courtesy of Flickr users rizziemelb, Kurt Eddy, and Graeme O. Churchard.*

Abstract

We present a novel 2D gesture annotation method for use in image-based 3D reconstructed scenes with applications in collaborative virtual and augmented reality. Image-based reconstructions allow users to virtually explore a remote environment using image-based rendering techniques. To collaborate with other users, either synchronously or asynchronously, simple 2D gesture annotations can be used to convey spatial information to another user. Unfortunately, prior methods are either unable to disambiguate such 2D annotations in 3D from novel viewpoints or require relatively dense reconstructions of the environment.

In this paper, we propose a simple multi-view annotation method that is useful in a variety of scenarios and applicable to both very sparse and dense 3D reconstructions. Specifically, we employ interactive disambiguation of the 2D gestures via a second annotation drawn from another viewpoint, triangulating two drawings to achieve a 3D result. Our method automatically chooses an appropriate second viewpoint and uses image-based rendering transitions to keep the user oriented while moving to the second viewpoint. User experiments in an asynchronous collaboration scenario demonstrate the usability of the method and its superiority over a baseline method. In addition, we showcase our method running on a variety of image-based reconstruction datasets and highlight its use in a synchronous local-remote user collaboration system.

Keywords: Annotations, interactive disambiguation, image-based reconstruction, 3D reconstruction, collaboration, augmented reality, virtual reality, virtual navigation, image-based rendering

Concepts: •**Human-centered computing** → *Collaborative interaction; Interaction techniques;* •**Computing methodologies** → *Mixed / augmented reality; Virtual reality;*

*e-mail:{bnuernberger,mturk,holl}@cs.ucsb.edu
†e-mail:kuochin@ece.ucsb.edu
‡e-mail:grinta@umail.ucsb.edu
§e-mail:csweeney@cs.washington.edu

VRST '16,, November 02 - 04, 2016, Garching bei München, Germany
ISBN: 978-1-4503-4491-3/16/11
DOI: http://dx.doi.org/10.1145/2993369.2993371

1 Introduction

Using multiple images of a physical environment to create a computer model of that scene is useful in many applications, including virtual navigation, remote collaboration, and augmented reality (AR). Typically, images of the scene are first acquired and then fused together to create a model of the scene using various computer vision techniques, such as structure from motion (SfM) [Snavely et al. 2006; Fuhrmann et al. 2015; Sweeney et al. 2015]. While most prior work has focused on creating such models, in this paper we are interested in how to annotate the models for a variety of applications, including synchronous and asynchronous remote collaboration.

We take the approach of using simple 2D gesture annotations, which have been successfully used in the past for collaboration in both virtual and augmented reality [Jung et al. 2002; Gauglitz et al. 2014b; Kasahara et al. 2012; Kasahara and Rekimoto 2014; Kim et al. 2014; Kim et al. 2015; Nuernberger et al. 2016; Fakourfar et al. 2016; Lien et al. 2016]. The inherent challenge with this approach is how to disambiguate a 2D annotation in a 3D world; from a single viewpoint, a back-projected 2D point corresponds to a 3D ray in the world and thus is inherently ambiguous in its 3D position. In synthetic virtual reality scenes, automatic disambiguation can be readily achieved since all the scene geometry and semantics are known *a priori* [Jung et al. 2002]. However, in image-based reconstructions (including ones used for augmented reality), scene geometry and semantics are typically not fully known. As a result, automated approaches to disambiguating 2D gesture annotations resort to trying to infer as much as possible from the drawings and

the reconstruction in order to place the annotation in 3D [Gauglitz et al. 2014a; Nuernberger et al. 2016; Lien et al. 2016]. Unfortunately, these automated disambiguation approaches fail when their inferences are incorrect or the scene geometry becomes too sparse. For example, if geometry corresponding to a particular part of an image does not exist, automatic disambiguation methods based on ray-casting will fail; an example where this is the case is shown in Figure 2.

(a) *RGB image.* **(b)** *Underlying semi-dense model.*

Figure 2: *In this semi-dense reconstruction, the trees are only partially modeled and thus ray-casting will not work for automatic disambiguation of 2D annotations. This model was created using MVE [Fuhrmann et al. 2015] and contains 993,506 points; it corresponds to the sparse Campus2 dataset in Table 1.*

Interactive disambiguation, on the other hand, allows users to actively contribute to the disambiguation process. While approaches to interactive disambiguation exist for simple point annotations in augmented reality [Polvi et al. 2016], to our knowledge no such method yet exists for 2D gesture annotations in image-based reconstructions (for the purpose of either virtual or augmented reality). In this paper, we present our method to fill in this gap.

Our method works as follows. A remote user is able to explore an existing 3D model of the scene created via structure from motion. Two modes of virtual navigation are supported — free-flight navigation and navigation constrained to the high resolution input photographs, using image-based rendering to transition between photos. The remote user can draw simple 2D arrow and circle annotations. To interactively disambiguate the annotations, the remote user is shown a second viewpoint to re-draw the annotation, which is chosen to effectively disambiguate the annotation using triangulation. We take special care to appropriately transition from the first drawing viewpoint to the second drawing viewpoint so that the user does not become disoriented.

1.1 Motivating Scenarios

Annotating image-based reconstructed models is important in many collaboration scenarios. This includes both asynchronous and synchronous collaboration in remote-remote configurations (*i.e.*, two remote users not physically present at the scene represented by the image-based model) and remote-local configurations (*i.e.*, only the local user is physically present at the scene represented by the image-based model).

For example, imagine a construction manager remotely viewing photos of a construction site, wishing to asynchronously collaborate with others. He or she can draw annotations which are placed in 3D using a reconstructed model of the scene. Later, another remote worker or an on-site worker can then view those annotations from novel viewpoints using the image-based model or in 3D augmented reality.

As another example, imagine an emergency response situation where expert responders are *en route* to the scene and want to obtain specific visual information of the situation before arrival. A remote expert structural engineer may desire to have certain close-up photos of specific parts of a building to determine if it may collapse. Rather than verbally describing where to go, the expert can draw annotations that are placed in 3D using a reconstructed model of the scene. As a result, local responders see the annotations in 3D augmented reality and thus can be guided directly to capture the viewpoints desired by the remote expert.

This last example highlights situations where an image-based reconstruction is not yet dense enough for use with other annotation authoring methods; this could either be due to a limited number of input photos or limited time available for creating the reconstruction. This last example also illustrates cases where specific high resolution photographs of particular parts of a model need to be made available; thus, rather than using automatic approaches to improving the model (*e.g.*, "next-best-view" approaches [Mauro et al. 2014]), our method can be used for remote expert users to guide local users or drones to capture specific images of the scene.

1.2 Contributions

We demonstrate the applicability of our method in a user experiment using an asynchronous collaboration task between two remote users. Furthermore, we present a prototype synchronous collaboration system for a remote-local user collaboration scenario. Our method is applicable both to small area (*e.g.*, indoor) scenes and wide-area (*e.g.*, outdoor) scenes, and both sparse and dense reconstructions. As such, it is a general approach for disambiguating 2D annotations in image-based 3D models. Our main contributions in this paper are:

1. A novel method for authoring 2D arrow and circle drawing annotations with interactive disambiguation in image-based 3D reconstructions.
2. A user experiment demonstrating the usability and applicability of our method in an asynchronous collaboration scenario between two remote users. Our method was shown to be intuitive, easy to use, and useful overall; furthermore, it enables faster understanding of annotations in 3D compared to a baseline approach.
3. A prototype system demonstrating our method running in a real-time remote-local user collaboration scenario.

2 Related Work

The related work for our method mainly falls into two categories: (1) annotation authoring in image-based reconstructed scenes, and (2) remote collaboration in AR.

2.1 Annotation Authoring in Image-based Reconstructed Scenes

Image-based reconstructions can be used for both virtual and augmented reality annotation authoring. Wither *et al.* give a detailed overview and taxonomy of AR annotations, specifically noting the lack of interfaces for authoring (*i.e.*, creating) annotations in AR [Wither et al. 2009]. While many types of annotations exist, our work falls under the category of 2D annotations for 3D scenes [Pierce et al. 1997; Bourguignon et al. 2001; Jung et al. 2002; Nuernberger et al. 2016; Polvi et al. 2016; Lien et al. 2016].

As mentioned previously, the main challenge for 2D annotation authoring for 3D scenes is that 2D drawings are inherently ambiguous in 3D. Automated methods exist for disambiguating 2D gesture annotations in 3D [Gauglitz et al. 2014a; Nuernberger et al. 2016;

Lien et al. 2016]; however, while such methods work for densely reconstructed models (*e.g.*, small indoor scenes with diffuse surfaces), they can fail when their automatic inferences are incorrect and for non-densely reconstructed models (*e.g.*, outdoor, wide-area scenes; see Figure 2).

Interactive disambiguation is another approach for resolving the 2D-3D ambiguity. The work most related to ours is Polvi *et al.*'s method, SlidAR, that allows users to interactively disambiguate a 2D point annotation in 3D AR [Polvi et al. 2016]. SlidAR provides interactive disambiguation of the depth of a ray-casted 3D point annotation by using triangulation via a manually assumed second viewpoint. Our method takes this a step further, handling 2D arrow and circle annotations rather than simple point annotations. In addition, while SlidAR was designed to work in small scenes for AR, our method is agnostic of the scene size and can be used in both a fully virtual image-based reconstructed environment and for AR as well. Finally, while SlidAR requires the user to manually and physically choose a second viewpoint for annotation disambiguation in AR, we provide an automatic method to determine a best second viewpoint to use in virtual image-based reconstructions.

2.2 Remote Collaboration in AR

Enhancing remote collaboration with augmented reality is an active area of research [Sodhi et al. 2013; Gauglitz et al. 2014b; Kasahara and Rekimoto 2014; Amores et al. 2015; Tait and Billinghurst 2015; Nuernberger et al. 2016; Fakourfar et al. 2016; Lien et al. 2016]. Typically, a remote user is helping a local user who is physically present in some environment; in the case of synchronous collaboration, both users are connected via a network connection (*e.g.*, for streaming video or a 3D model of the local user's environment).

While one annotation approach is to enhance collaboration by directly relaying hand gestures from the remote user to the local user [Sodhi et al. 2013; Amores et al. 2015], our work lies in the category of using drawing annotations to enhance the collaboration [Gauglitz et al. 2014b; Kasahara and Rekimoto 2014; Kim et al. 2014; Kim et al. 2015; Nuernberger et al. 2016; Fakourfar et al. 2016; Lien et al. 2016]. It has been shown that circle and arrow gesture annotations are the most common gestures in a remote collaboration scenario [Nuernberger et al. 2016], and this motivates our method of handling these two gesture types. We note that drawing annotations may be considered more useful due to arm fatigue with hand gestures [Hincapié-Ramos et al. 2014], asynchronous collaboration scenarios, and situations where view independence is used [Gauglitz et al. 2014b; Tait and Billinghurst 2015; Nuernberger et al. 2016; Lien et al. 2016].

View independence refers to the ability for the remote user to take on a viewpoint independent of the local user's current viewpoint; this has recently been achieved via image-based reconstructions of the local user's environment [Gauglitz et al. 2014b; Kasahara and Rekimoto 2014; Tait and Billinghurst 2015]. Typically, some form of image-based rendering is used to render the scene for the remote user. Recently it was confirmed that having view independence in remote collaboration speeds up task and user performance in collaboration [Tait and Billinghurst 2015]. Our method lies in this same category of view independent systems for remote collaboration, not only in AR but also in virtual image-based reconstructions as well. Section 3 describes how a remote user can navigate in the reconstructed scene in our method.

Finally, our method is also applicable to remote collaboration between two remote users, in either asynchronous or synchronous collaboration scenarios.

3 Virtual Navigation Interface

We have implemented two forms of virtual navigation — free-flight and constrained-to-photos. Users can either freely fly throughout the environment similar to common unconstrained 3D navigation interfaces, or navigate throughout the environment while being constrained to stay at the input photos[1]; in the latter case, image-based rendering techniques are used for transitioning between photos. Note that the constrained navigation interface may have an advantage over free-flight in the sense that the images may give the user a better sense of the scene than the reconstructed model, especially if the model is not dense.

In the following, we assume there is an existing set of cameras $\mathcal{C}$, each with an associated RGB image $I \in \mathcal{I}$. A bold $\mathbf{c} \in \mathcal{C}$ represents the 3D position of the camera C.

3.1 Free-flight Navigation

Free-flight navigation uses a standard computer mouse. Holding left-click while dragging causes side-to-side translation, parallel to the image plane. Holding right-click while dragging causes rotation-only movement. Finally, the scroll wheel causes translation along the optical axis, either forward or backward motion. Users can furthermore click onto camera frusta, representing the set of reconstructed cameras $\mathcal{C}$, to assume the position of that camera; the associated image $I \in \mathcal{I}$ for that camera is then shown in the view. To avoid unnecessary visual clutter, users are able to toggle whether the camera frusta are visible at any given time.

3.2 Constrained-to-Photos Navigation

This navigation interface is inspired by Photo Tourism [Snavely et al. 2006] and Microsoft's Photosynth. Rather than giving the user full freedom of movement, constraining the user's travel can help with the usability of the interface. Such constrained travel is also very important when there is a sparsity of scene data, in which case image-based rendering can provide a better sense of the scene compared to a sparse point cloud. We have implemented two travel techniques: straight (forward/backward) movement and rotation.

Straight movement (forward/backward) is performed via the mouse scroll wheel. We first filter out any cameras whose viewing directions are not within 30° of the ray $\mathbf{r}$ determined by the (x, y) location where the mouse scroll was performed. Furthermore, we make sure that the cameras we consider are in front or behind the current camera C_1 and at least a $0.1m$ away. This leaves us with a set of cameras $\mathcal{C}'$ which we further analyze for deciding where to move to. Specifically, we choose the camera $\hat{C}$ with the lowest cost as follows:

$$\hat{C} = \underset{C \in \mathcal{C}'}{\arg\min}((0.5 \cdot (1 - Dir(C, \mathbf{r})) + 0.5) \cdot ||\mathbf{c_1} - \mathbf{c}||^2) \quad (1)$$

Where $Dir(C, \mathbf{r})$ gives the dot product between the viewing direction of C and $\mathbf{r}$. This function gives an increasing cost to further away cameras and penalizes cameras whose viewing directions are far away from the input ray $\mathbf{r}$.

Rotation movement is performed by right-clicking and dragging the mouse. We first filter out any cameras that have viewing directions greater than 45° away from the current viewing direction. We then

[1] This kind of interface is very popular today in many interfaces for virtually navigating captured scenes, including Google Street View, Matterport, Mapillary, etc.

search the remaining cameras $\mathcal{C}'$ and choose the one with lowest cost via:

$$\hat{C} = \underset{C \in \mathcal{C}'}{\arg\min}((1 - Dir(C_{live}, C)) \cdot ||\mathbf{c_1} - \mathbf{c}||^2) \quad (2)$$

Where C_{live} represents the current, live virtual camera viewpoint. The image $\hat{I}$ of the camera $\hat{C}$ with the lowest cost is then shown using a proxy plane blending the current camera's image I_1 with $\hat{I}$.

During all transitions between cameras, the two images used in the transition are projected onto a proxy plane and alpha blended together according to the distance between each camera [Snavely et al. 2006]. If the two cameras do not share common points, a simple pairwise blending of the images occurs instead. Interpolated projection matrices and camera positions use linear interpolation, while interpolated orientation uses spherical linear interpolation; we use a "smoothstep" function as input to the interpolation.

4 Annotation Authoring Interface

Our system supports two types of drawing gestures — arrows and circles[2]. We use the $1 recognizer [Wobbrock et al. 2007] to detect which type the user drew.

Annotations are authored in a two step process (hence the name "multi-view"). First, the user draws the annotation from an initial camera viewpoint C_1. Upon completion of the first drawing, the system automatically determines a good second camera viewpoint $\hat{C}$ from which the user can then perform disambiguation via triangulation by drawing a second time. We now first describe how the second viewpoint is determined, provide specifics for arrow and circle gestures, describe how the image-based rendering transition to the second viewpoint is achieved, and finally present some implementation timing results.

4.1 Determining the Second Viewpoint

Since our method is agnostic to the density of the image-based reconstruction, we prefer showing a high resolution photo $I \in \mathcal{I}$ for the second viewpoint. Thus, we must choose a viewpoint $\hat{C}$ from the set of input cameras $\mathcal{C}$. Having a view too close (*i.e.*, without enough parallax) to the original viewpoint C_1 or too far away from it will not be useful for interactive disambiguation via triangulation, and thus we must be very careful in how we choose the second viewpoint.

The second viewpoint is determined as follows (see Figure 3). We determine a 3D line segment l to go from the current camera's position $\mathbf{c_1}$ through either the 2D arrow head or the average point of the circle gesture to a 3D point $\mathbf{b}$. This point $\mathbf{b}$ is either the intersection of the ray with the scene geometry, if it exists, or the 3D point along the ray at the distance of the average scene depth in C_1 in a window around the 2D arrow head point or average point of the circle gesture (we use a $100px$ window in our implementation). We then filter out cameras by making sure that at least part of l is seen in that camera. Finally, we search through all the remaining cameras $\mathcal{C}'$ for the camera $\hat{C}$ that has the best vantage point to disambiguate the annotation; specifically, we use a cost function in the following formulation:

$$\hat{C} = \underset{C \in \mathcal{C}'}{\arg\min}(-\alpha \cdot \beta \cdot InView(l, C) \cdot FillingView(l, C) + \gamma \cdot AbsDir(l, C)) \quad (3)$$

where $InView(l, C) = \frac{||\mathbf{b}' - \mathbf{c_1}'||}{||\mathbf{b} - \mathbf{c_1}||}$ calculates the percentage of l that is seen in C, where $\mathbf{b}'$ and $\mathbf{c_1'}$ are the 3D points on l that are closest to $\mathbf{b}$ and $\mathbf{c_1}$, respectively, and still inside the viewing frustum of C; this term favors views that can see more of the line l. $FillingView(l, C) = \frac{||b' - c_1'||}{diag(C)}$ calculates how much l fills up the view, where b' and c_1' are the projections of $\mathbf{b}'$ and $\mathbf{c_1'}$ into C; $diag(C)$ is the length of the image diagonal. The last term $AbsDir(l, C) = |Dir(l, C)|$ calculates the absolute value of the dot product between the line direction l and the optical axis of C; this term penalizes views that have directions more similar to l. Finally, α and γ are weight terms (set to 1 and 0.3 in our implementation) and β is a term that penalizes not seeing $\mathbf{b}$ (set to 1 when seeing $\mathbf{b}$ and 0.5 otherwise).

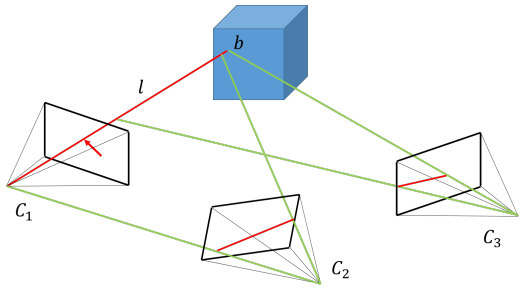


Figure 3: *Illustration of finding the best second viewpoint to disambiguate a 2D drawn arrow in viewpoint C_1. Here, the line l is defined by the ray cast from C_1 through the 2D arrow head and intersecting with the scene geometry at a 3D point b. C_2 and C_3 are candidate second viewpoints. In this illustration C_2 is favored over C_3 since more of the line segment l is seen in its view and the line segment l fills up more of its view.*

Once the second viewpoint is determined, we draw the line l as a semi-transparent helper line (*e.g.*, see Figures 1b and 1c) to help guide the user to disambiguate the annotation by drawing it a second time from the second viewpoint. Figure 4 shows an example bird's eye view of an actual image-based reconstruction after Equation 3 has been evaluated; in this case, $\hat{C}$ was the bright green camera frustum in the top-left of the figure.

4.2 Arrow Gesture Specifics

To disambiguate arrow gestures, we apply Equation 3 using the 2D arrow head point to calculate the line l; the arrow head point is calculated via analyzing changes in the drawing direction [Ou et al. 2003]. Once the second viewpoint is shown, the user draws a second arrow. Midpoint triangulation is used to determine the disambiguated 3D head and tail of the arrow. If the 3D ray cast through the 2D head of the second drawn arrow intersects with scene geometry, we use this intersection point instead; if l also intersected with scene geometry, we use the average of the two intersection points. An illustration of this process is shown in Figure 5.

Arrows are rendered in 3D with their heads oriented towards the user's view. Transitions between views are stabilized by adjusting pairwise proxy planes to pass through the arrow head.

[2]Arrows and circles were shown to be the most common gestures in a remote collaboration scenario [Nuernberger et al. 2016]; however, other 2D gestures could also be handled using an approach similar to what is described here.

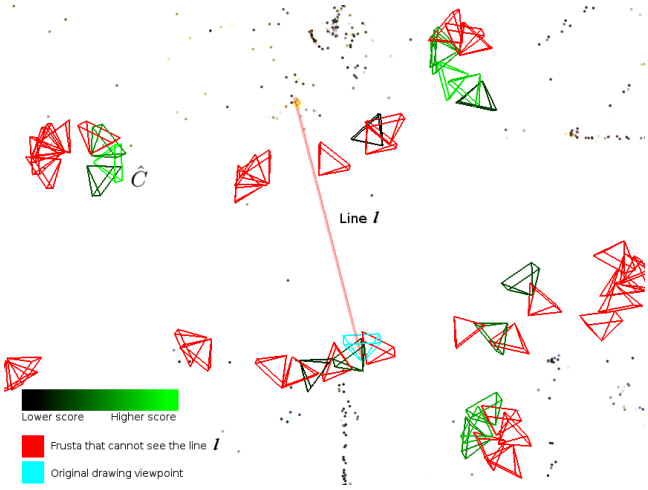


Figure 4: *Example of candidate second viewpoints from a bird's eye view of an outdoor courtyard scene, with sparse reconstructed 3D points shown alongside camera frusta. Colors of camera frusta: red, line l is not in view; brighter green, higher score in Equation 3; darker green, lower score in Equation 3; light blue, original drawing viewpoint C_1. The semi-transparent red line l can be seen in the middle of the figure.* ***[Best viewed in color]***

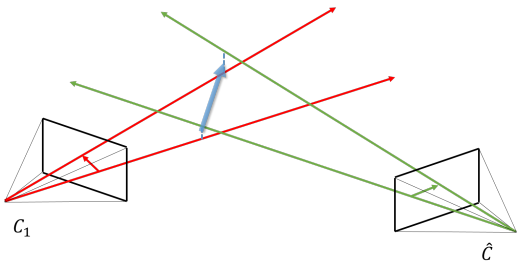


Figure 5: *Illustration of arrow disambiguation via triangulation of two user-drawn arrows. C_1 is the original drawing viewpoint and $\hat{C}$ is the second drawing viewpoint found via Equation 3. The blue arrow is the final 3D arrow.*

4.3 Circle Gesture Specifics

To disambiguate circle gestures, we apply Equation 3 using the average point of the circle gesture to calculate the line l. Once the second viewpoint is shown, the user draws a second circle. The center of the 3D circle is determined via midpoint triangulation using the average point of the second circle gesture to determine the triangulating ray.

To determine the 3D circle's radius, we find 3D rays corresponding to the top, left, center, right, and bottom points of the 2D circles. We then perform midpoint triangulation with the following pairs of rays: ($center_1$, $left_2$), ($center_1$, $right_2$), ($right_1$, $center_2$), ($left_1$, $center_2$), (top_1, top_2), ($bottom_1$, $bottom_2$). The radius is then the average distance between each of these triangulated points and the midpoint triangulation for ($center_1$, $center_2$).

Circles are rendered oriented towards the user's view. We stabilize transitions by adjusting pairwise proxy planes to pass through the circle's center point.

4.4 Transitioning to the Second Viewpoint

A naïve approach to moving the user to the second viewpoint determined by Equation 3 is to teleport there immediately. This approach, however, has long been known to increase spatial disorientation [Bowman et al. 1997]. Therefore, to help users stay oriented while transitioning to the second viewpoint, we implemented a modified version of the path planning algorithm described by Snavely *et al.* [Snavely et al. 2008].

The transition algorithm precomputes image-based rendering pairwise transition costs between all cameras and uses Dijkstra's algorithm [Dijkstra 1959] to determine an optimal path between two cameras. We precompute the pairwise transition costs and run Dijkstra's algorithm in real-time after Equation 3 has determined the second viewpoint. To help further maintain spatial orientation for the user, we force Dijkstra's algorithm to only consider cameras that can see the line l.

We compiled videos of several transitions with different transition times and numbers of cameras per transition, and we showed these to a group of researchers well versed in visualization and virtual reality. Overall, most found that limiting the number of cameras per transition makes the transition fast enough yet still usable for maintaining orientation. Thus, after some final empirical testing, we settled on using one second as the transition time between each pair of images and a maximum path length of three cameras per transition. Please see the supplemental video[3] for examples of these transitions.

4.5 Implementation Results

We have tested the remote user interface using various datasets, including reconstructions with sparse sets of cameras and ones with dense sets of cameras from online community photos [Wilson and Snavely 2014; Sweeney et al. 2015]. On average, evaluating Equation 3 took 18ms (min. 2ms; max 44ms) and the path planning algorithm 27ms (min. 0.3ms; max 105ms). Total average time for circle gestures was 59ms for the system to determine the second viewpoint (min. 4ms; max 148ms) and 0.7 ms to perform disambiguation via triangulation (min. 0.3ms; max 3ms). Total average time for arrow gestures was 39ms for the system to determine the second viewpoint (min. 4ms; max 126ms) and 0.6 ms to perform disambiguation via triangulation (min. 0.04ms; max 6ms). We used an Intel Core i7-4790 3.6GHz CPU with 16GB of RAM and an NVIDIA GeForce GTX 980 Ti GPU while obtaining these timing results. Datasets ranged from 30 cameras to 730 cameras, from 8,003 3D points to 334,622 3D points. Figures 1 and 10 show examples using reconstructions with dense sets of cameras. Please see the supplemental video for more results.

5 Annotation Authoring User Experiments

In order to evaluate the annotation authoring method, we conducted a user experiment using an asynchronous collaboration scenario between two remote users (*i.e.*, both users are not physically present in the scene represented by the image-based model). We used this scenario since it enabled us to use four very different image-based reconstructions in our evaluation; nevertheless, we also demonstrate our method working in a synchronous remote-local collaboration scenario in Section 6. Statistics of the four different models used in the experiment are detailed in Table 1. Figure 11 shows an example of the Campus1 dataset.

The experiment was divided into two parts, Part A and Part B. The

[3] https://youtu.be/phPAk7JnBG8

Reconstruction	# Points	# Cameras
Campus1	8,003	67
Campus2	15,872	136
Castle-P30	24,968	30
NYC Library	70,167	340

Table 1: *Statistics of the reconstructions used in the annotation authoring user experiment. Campus1 and Campus2 are datasets we created at a university campus; Castle-P30 is from [Strecha et al. 2008] and NYC Library is from [Wilson and Snavely 2014]; all datasets used the Theia SfM Library for the reconstructions [Sweeney et al. 2015].*

task in Part A was to draw circle or arrow gesture annotations to convey information to a future collaborator, while the task in Part B was to use the annotations from Part A to answer simple questions corresponding to the information conveyed from the annotations in Part A, such as "Which tree?" Participants either completed Part A only, or completed both Part B and then Part A (in that order, since completing Part A first would bias the results in Part B). In Part A, for each dataset participants drew two arrows and two circle annotations, making a total of 16 drawing annotations overall.

Participants in Part B were paired with a single participant from Part A. To further confirm the effectiveness of using 3D annotations, we compared showing participants in Part B our method's final 3D annotations in the field of view or showing a baseline approach of a screenshot on the side of the first drawn annotation by the participant in Part A (*i.e.*, the drawing before disambiguation). Participants in Part B were shown 3D annotations for the first two datasets and screenshots on the side for the last two datasets (or vice versa to balance the conditions); as a result, two participants completed Part B for each participant who only completed Part A.

5.1 Procedure

Participants first completed a pre-study questionnaire (mostly for gathering information on demographics), completed the tasks, and finally completed a post-study questionnaire that addressed the usability of the interface; participants in Part B also completed a separate Part B questionnaire. For Part A, participants were trained on both the navigation and annotation interfaces (because this study was mainly focused on evaluating the annotation authoring method, we only let participants use the constrained-to-photos navigation; see Section 7 for discussion on this choice). In both Part A and B, a set of practice tasks were used to familiarize participants with the actual task; the order of the tasks were balanced using a 4x4 Latin Square design. The actual tasks' procedures were as follows.

In Part A, participants clicked on a "Start Task" button, after which they saw a statement indicating which object to circle or point to. The experimenter told them to draw the annotations as fast as possible accurately. Once they completed the annotation process, participants could either repeat the task by pressing a button, if not satisfied with the annotation result, or stop the task by pressing a different button. Participants were required to begin drawing from a specific viewpoint C_1, while in the disambiguation process they were allowed to virtually navigate if they so desired.

In Part B, participants clicked on a "Start Task" button, after which they were shown 3D annotations in the field of view or screenshots on the side. In either case, for each task, the main viewpoint for Part B showed a large difference in perspective from that of the Part A's first viewpoint C_1. A simple question was shown asking the participants to identify what the annotation was referring to. Once confident of their answer, they clicked on a "Stop Task" button and then let the experimenter know their answer.

5.2 Results

12 participants completed the study (ages 19-22, avg. 20.25; 5 male), 4 completing only Part A, while 8 completing both Parts A and B. 11 participants were barely or not familiar with interactive 3D software; 10 used a computer mouse at least several days a week. All participants were relatively familiar with the two campus datasets and not familiar with the other two.

In Part A, participants took on average 5.28 seconds to draw the second annotation during the disambiguation phase (median 4.38; stdev. 4.06). On average, they repeated each task 0.21 times and used 1.18 views; hence, in most cases participants did not need to repeat the task and did not need to virtually navigate to a better view (*i.e.*, they typically only used the second viewpoint chosen by Equation 3).

In Part B, the average time it took users to answer a question was 2.64 seconds with 3D annotations in the view (median 2.20; stdev. 1.70) and 7.92 seconds with annotations in screenshots on the side (median 7.17; stdev. 4.82). A factorial ANOVA (with factors annotation condition, task, and dataset) confirmed a statistically significant difference between the two annotation conditions (p-value $\ll 0.001$), using the Tukey HSD post-hoc test. There was only 1 incorrect answer (1.56%) using 3D annotations in the view, while 18 incorrect answers (28.13%) using annotations in screenshots on the side (11 of these were in the campus datasets with which all participants said they were relatively familiar).

Questionnaire results are shown in Figures 6 and 7. As can be seen from the figures, participants generally agreed that the various aspects of the system and method were intuitive, easy to use, and useful overall. Finally, some comments from the questionnaire include:

"For the screenshot on the side you have to mentally change your perceived field of view to really understand the area."

"The annotation interface was easy to use and was useful because it was straightforward and did not require any previous knowledge to utilize."

"The 2nd view is helpful in that it allows you to view the designated area or object from a different perspective which helps create a mental map of the object and surrounding."

"I imagine that such a system would be used for virtual reality navigation as well as helping someone form a mental image/map of a place before they go there."

6 Synchronous Local-Remote Collaboration System

To demonstrate our annotation authoring method in a synchronous collaboration scenario, we built a prototype system for local-remote user collaboration.

6.1 System Overview

Our system consists of two major components, a remote user component and a local user component (see Figure 8). The remote user utilizes an existing reconstructed model of a physical space in which the local user currently is situated. The Theia SfM Library [Sweeney et al. 2015] is used to obtain sparse point cloud reconstructions, which are optionally fed into MVE [Fuhrmann et al. 2015] to obtain dense polygon mesh reconstructions. The remote

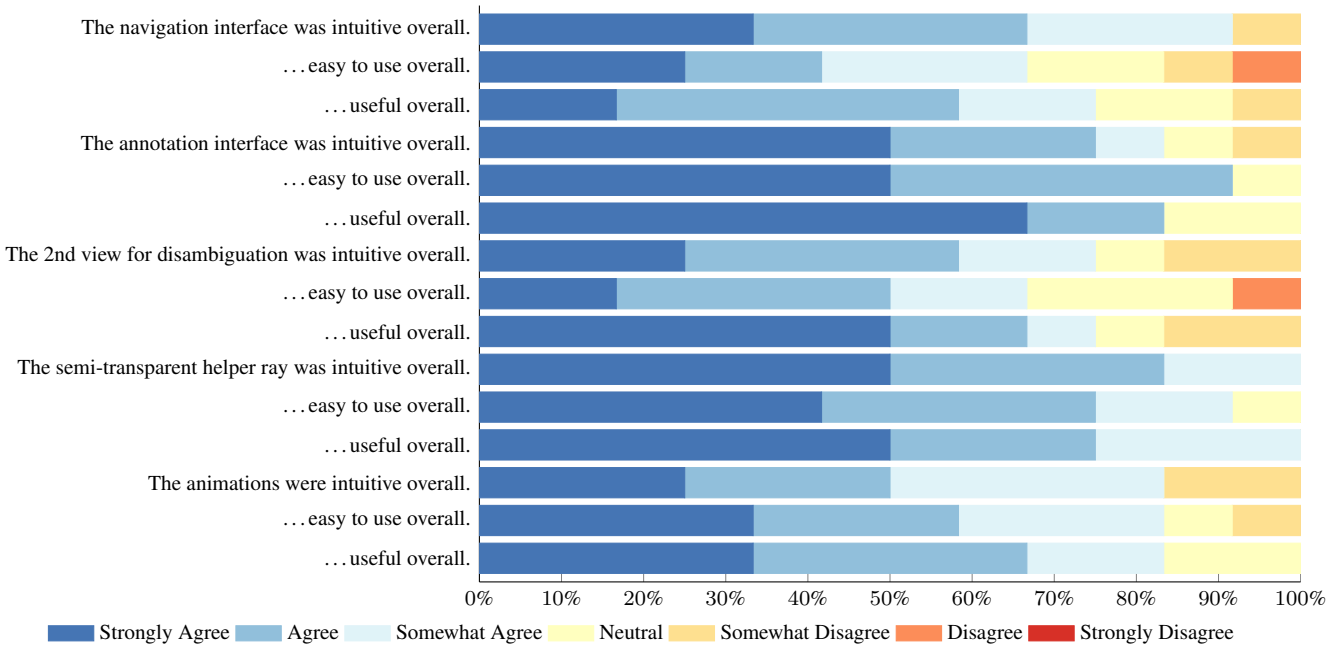


Figure 6: *Questionnaire results for Part A (12 participants).* ***[Best viewed in color]***

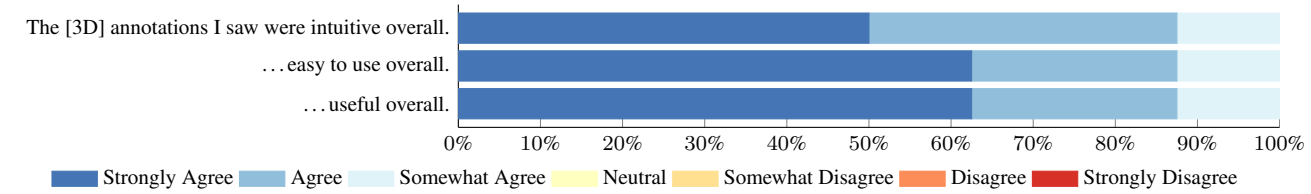


Figure 7: *Questionnaire results for Part B (8 participants).* ***[Best viewed in color]***

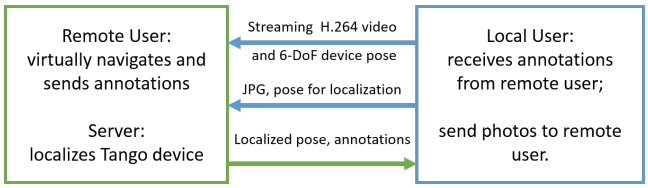


Figure 8: *Overview diagram for our prototype synchronous local-remote collaboration system.*

user is able to virtually navigate through and annotate the reconstructed model, using the interface described in Sections 3 and 4. Such annotations are then sent to the local user through a network connection.

The local user has a handheld augmented reality device; in our implementation, we chose to use the Google Project Tango tablet, featuring built-in visual tracking capabilities. The Tango tablet is linked to the remote user's system via a network connection (TCP in our implementation). H.264 encoded video and the 6 degree-of-freedom (DoF) device pose (position and orientation) is streamed to the remote user. The remote user then has the ability to follow the local user's device pose or to virtually navigate independently of the local user's view. The local user can also send back photos to the remote user which are then displayed in their correct 6-DoF pose in the remote user's view. Audio is streamed using a separate off-the-shelf solution.

To localize the Tango tablet to the model, the local user presses a "Localize" button on the tablet which sends a JPEG compressed video frame and the current 6-DoF pose over the network to the remote user's system. SIFT [Lowe 2004] and FLANN [Muja and Lowe 2009] are used to find 2D-3D correspondences to the sparse reconstruction; this is followed by P3P absolute pose estimation inside a RANSAC loop to localize the video frame [Kneip et al. 2011]. The reconstruction is manually scaled so that one unit equals one meter, to coincide with Tango's coordinate system scale. Note that our system allows the local user to continue to move throughout the physical environment while the localization process is being performed remotely. Localization accuracy is visually verified by showing the sparse point cloud in AR on the Tango device.

6.2 Results

The remote user's system runs with an average frame rate of 58Hz when not connected to the local user and 14.2Hz when connected to the local user, on an Intel Core i7-4790 3.6GHz CPU with 16GB of RAM and an NVIDIA Quadro K5000. The local user's system, a Google Project Tango tablet, runs with an average frame rate of 30Hz. In our experiments, it takes an average of 3.8 seconds to perform localization with an average round trip time of 4.4 seconds between pressing the "Localize" button and receiving the localized pose from the remote user's system; it takes on average 88.2ms to compress the frame and send it along with the pose on the Tango

tablet side. All experiments were performed using a shared campus WiFi network. Figure 9 shows an example of the collaboration. Please see the supplemental video for more results.

7 Discussion

While automatic methods for disambiguating 2D gesture annotations in 3D work nicely for densely reconstructed scenes, they may not always work for semi-dense scenes and will usually fail in very sparse scenes (for an example semi-dense scene where this is the case, see Figure 2). Our method is agnostic to the density of the reconstructions. On the one hand, as sensors and image-based reconstruction algorithms improve, the availability of dense reconstructions will only increase. On the other hand, sparse reconstructions will always be quicker to produce (especially for reconstructing wide-area, outdoor unstructured sets of photos) and more efficient in terms of computer memory storage; sparse models can also be produced more easily with sparse sets of photos compared to dense models. Finally, while many outdoor scenes are able to be reconstructed densely to a certain extent, many fine scene details are still extremely difficult for computer vision algorithms to reconstruct at all (*e.g.*, small objects, non-diffuse surfaces, etc.; see Figure 2). For these reasons we chose a rather simple baseline approach to compare against in Section 5. One downside for using sparse reconstructions is that any method will not be able to easily determine if the line l is occluded from a second viewpoint; for this special case, our method allows users to simply virtually navigate to a better viewpoint.

In this paper, we focused specifically on evaluating the multi-view gesture annotation authoring method. For this we chose to use an asynchronous collaboration task, allowing us to evaluate our method across four different image-based reconstructions. Previous work has focused more on experimenting with synchronous collaboration scenarios. In this work we were mainly concerned with the usability of the annotation authoring method and thus both synchronous and asynchronous scenarios are applicable. We note that both scenarios include view independent situations which demonstrate the importance of correctly placing 2D annotations in 3D [Gauglitz et al. 2014b; Kasahara and Rekimoto 2014; Tait and Billinghurst 2015; Nuernberger et al. 2016]. One downside of interactive disambiguation methods in synchronous collaboration scenarios is that they generally take longer than automatic approaches; future work should explore first employing an automatic disambiguation method followed by interactive correction by the user as needed.

If the viewpoint chosen by Equation 3 is not satisfactory, the user has the option to virtually navigate to a better view to perform the disambiguation. A possible failure case with our method is that no good second viewpoint $\hat{C}$ may exist for disambiguating 2D gesture annotations. In this case, free-flight navigation along with semi-dense models will help in the disambiguation. Other types of 3D annotation authoring methods may also be explored in this case [Wither et al. 2009].

Future work should explore adjusting Equation 3 to incorporate the path planning algorithm presented in Section 4.4. For example, a candidate viewpoint C_n may have a lower image-based rendering transition cost compared to another candidate viewpoint C_m. By incorporating this into Equation 3, the transition to the second viewpoint should maintain higher spatial orientation as a result of having better image-based rendering transitions.

Finally, it should be re-emphasized that since our method is agnostic to the density of reconstructions, it is extremely useful in outdoor image-based reconstructions, which tend to be built using structure from motion. The impact of this includes both virtual and augmented reality applications. While much previous AR work focuses on indoor and small scene reconstructions (*e.g.*, [Klein and Murray 2007; Newcombe et al. 2011]), less work has explored outdoor, wide-area AR scenes under a variety of scene conditions. Our method is especially useful in this latter case.

8 Conclusion

We presented a novel annotation authoring method for use in image-based 3D reconstructed scenes. A remote user is able to virtually navigate through the scene with free-flight or constrained-to-photos navigation and can also annotate the scene using 2D arrow and circle gestures. To disambiguate the drawing gestures in 3D, we show users a second viewpoint from which they can then draw the gesture a second time to achieve disambiguation via triangulation. We introduced an automatic way to determine such a second viewpoint and transition the user there in such a way to keep the user spatially oriented. A user experiment with an asynchronous remote-remote user collaboration task demonstrated the usability of our method and its superiority over a baseline method. Finally, a prototype synchronous local-remote user collaboration system was demonstrated as well as the annotation authoring method being used in many wide-area image-based 3D reconstructions.

Acknowledgements

This work was supported by NSF grants IIS-1219261 and IIS-1423676, and ONR grant N00014-14-1-0133. We thank Domagoj Baričević for useful discussions regarding different aspects of the work.

References

AMORES, J., BENAVIDES, X., AND MAES, P. 2015. Showme: A remote collaboration system that supports immersive gestural communication. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI EA '15, 1343–1348.

BOURGUIGNON, D., CANI, M.-P., AND DRETTAKIS, G. 2001. Drawing for Illustration and Annotation in 3D. In *Computer Graphics Forum*, vol. 20, Wiley Online Library, 114–123.

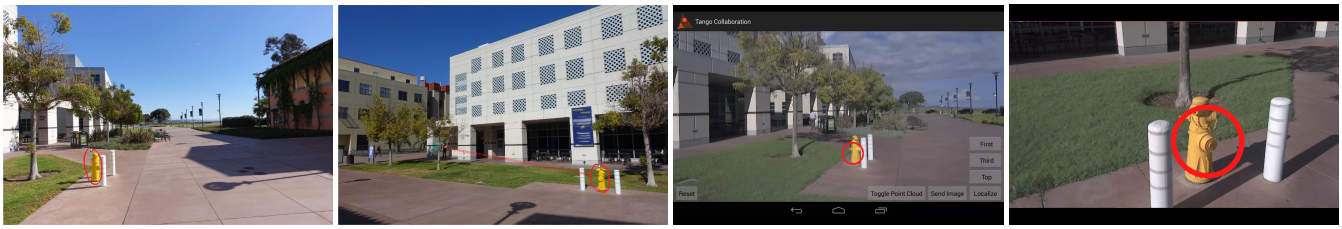
BOWMAN, D. A., KOLLER, D., AND HODGES, L. F. 1997. Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. In *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)*, IEEE Computer Society, Washington, DC, USA, VRAIS '97, 45–52.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 1 (Dec), 269–271.

FAKOURFAR, O., TA, K., TANG, R., BATEMAN, S., AND TANG, A. 2016. Stabilized annotations for mobile remote assistance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '16, 1548–1560.

FUHRMANN, S., LANGGUTH, F., MOEHRLE, N., WAECHTER, M., AND GOESELE, M. 2015. MVE—An image-based reconstruction environment. *Computers & Graphics 53, Part A* (Dec.), 44–53.

GAUGLITZ, S., NUERNBERGER, B., TURK, M., AND HÖLLERER, T. 2014. In Touch with the Remote World: Remote Collaboration with Augmented Reality Drawings and Virtual Navigation. In *Proceedings of the 20th ACM Symposium*



(a) *Remote user's first drawing.* **(b)** *Remote user's second drawing.* **(c)** *Local user's view.* **(d)** *Image sent by local user.*

Figure 9: *Screenshots of a remote user annotating the local user's physical environment. The remote user draws the annotation (a-b). The local user is guided by the annotation to capture a new image (c). The remote user sees the new viewpoint (d). Notice the local user's localized camera represented by a light-blue camera frustum and the semi-transparent helper ray in (b). This outdoor environment was reconstructed with 136 photos, creating 15,872 reconstructed 3D points.*

(a) *First drawing.* **(b)** *Second drawing.* **(c)** *Final 3D arrow.* **(d)** *Final 3D arrow.*

Figure 10: *Screenshots from annotating a reconstruction of the Pantheon (730 photos with 334,622 reconstructed 3D points). Note the red, semi-transparent line in (b). The final 3D arrow is shown in (c-d). Original images courtesy of Flickr users 63457916@N00, Michiel Jelijs, and Andy Rusch.*

on Virtual Reality Software and Technology, ACM Press, New York, New York, USA, VRST '14, 197–205.

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 annual ACM symposium on User interface software and technology*, ACM Press, New York, New York, USA, UIST '14, 449–459.

HINCAPIÉ-RAMOS, J. D., GUO, X., MOGHADASIAN, P., AND IRANI, P. 2014. Consumed Endurance: A Metric to Quantify Arm Fatigue of Mid-air Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '14, 1063–1072.

JUNG, T., GROSS, M. D., AND DO, E. Y.-L. 2002. Annotating and Sketching on 3D Web Models. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, ACM, New York, NY, USA, IUI '02, 95–102.

KASAHARA, S., AND REKIMOTO, J. 2014. Jackin: Integrating first-person view with out-of-body vision generation for human-human augmentation. In *Proceedings of the 5th Augmented Human International Conference*, ACM, New York, NY, USA, AH '14, 46:1–46:8.

KASAHARA, S., HEUN, V., LEE, A. S., AND ISHII, H. 2012. Second Surface: Multi-user Spatial Collaboration System Based on Augmented Reality. In *SIGGRAPH Asia 2012 Emerging Technologies*, ACM, New York, NY, USA, SA '12, 20:1–20:4.

KIM, S., LEE, G., SAKATA, N., AND BILLINGHURST, M. 2014. Improving co-presence with augmented visual communication cues for sharing experience through video conference. In *Proceedings of the 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 83–92.

KIM, S., LEE, G. A., HA, S., SAKATA, N., AND BILLINGHURST, M. 2015. Automatically freezing live video for annotation during remote collaboration. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI EA '15, 1669–1674.

KLEIN, G., AND MURRAY, D. 2007. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, Washington, DC, USA, ISMAR '07, 1–10.

KNEIP, L., SCARAMUZZA, D., AND SIEGWART, R. 2011. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, CVPR '11, 2969–2976.

LIEN, K.-C., NUERNBERGER, B., HÖLLERER, T., AND TURK, M. 2016. PPV: Pixel-Point-Volume Segmentation for Object Referencing in Collaborative Augmented Reality. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '16.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov.), 91–110.

MAURO, M., RIEMENSCHNEIDER, H., SIGNORONI, A., LEONARDI, R., AND VAN GOOL, L. 2014. A unified framework for content-aware view selection and planning through view importance. *British Machine Vision Conference*, 1–11.

MUJA, M., AND LOWE, D. G. 2009. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *Interna-*

(a) *First drawing.* **(b)** *Second viewpoint.* **(c)** *Second drawing.*

(d) *Final 3D circle.* **(e)** *Final 3D circle.* **(f)** *Final 3D circle.*

Figure 11: *Screenshot of annotating in the Campus1 dataset. The user draws an annotation (a) and automatically sees a second viewpoint for disambiguation (b) with a semi-transparent line l of the same color as the annotation. A second drawing (c) determines the final 3D circle as seen from different viewpoints (d-f).*

tional Conference on Computer Vision Theory and Applications (VISAPP '09), 331–340.

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 127–136.

NUERNBERGER, B., LIEN, K. C., HÖLLERER, T., AND TURK, M. 2016. Interpreting 2d gesture annotations in 3d augmented reality. In *Proceedings of the 2016 IEEE Symposium on 3D User Interfaces (3DUI)*, 149–158.

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*, ACM, New York, NY, USA, ICMI '03, 242–249.

PIERCE, J. S., FORSBERG, A. S., CONWAY, M. J., HONG, S., ZELEZNIK, R. C., AND MINE, M. R. 1997. Image plane interaction techniques in 3D immersive environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, ACM Press, New York, New York, USA, I3D '97, 39–43.

POLVI, J., TAKETOMI, T., YAMAMOTO, G., DEY, A., SANDOR, C., AND KATO, H. 2016. SlidAR: A 3D positioning method for SLAM-based handheld augmented reality. *Computers & Graphics 55* (Apr.), 33–43.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics 25*, 3 (July), 835–846.

SNAVELY, N., GARG, R., SEITZ, S. M., AND SZELISKI, R. 2008. Finding paths through the world's photos. *ACM Transactions on Graphics 27*, 3 (Aug.), 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 SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '13, 179–188.

STRECHA, C., VON HANSEN, W., GOOL, L. V., FUA, P., AND THOENNESSEN, U. 2008. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

SWEENEY, C., HÖLLERER, T., AND TURK, M. 2015. Theia: A fast and scalable structure-from-motion library. *Proceedings of the 23rd ACM International Conference on Multimedia*, 693–696.

TAIT, M., AND BILLINGHURST, M. 2015. The effect of view independence in a collaborative ar system. *Computer Supported Cooperative Work 24*, 6 (Dec.), 563–589.

WILSON, K., AND SNAVELY, N. 2014. Robust global translations with 1dsfm. In *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, Springer International Publishing, Cham, Switzerland, 61–75.

WITHER, J., DIVERDI, S., AND HÖLLERER, T. 2009. Annotation in outdoor augmented reality. *Computers & Graphics 33*, 6 (Dec.), 679–689.

WOBBROCK, J. O., WILSON, A. D., AND LI, Y. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, NY, USA, UIST '07, 159–168.