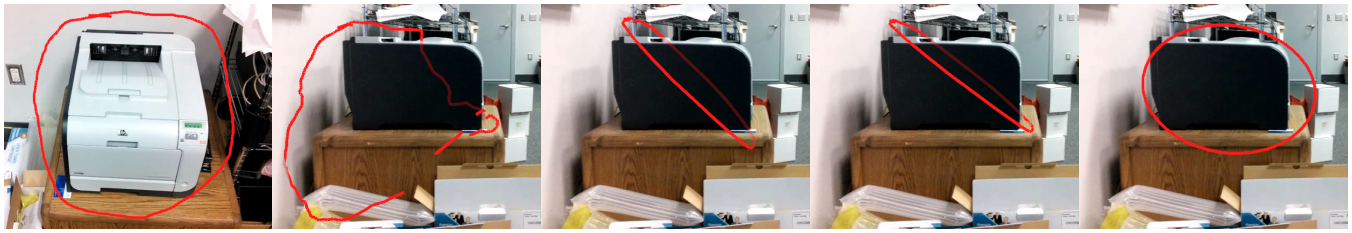
Interpreting 2D Gesture Annotations in 3D Augmented Reality

Benjamin Nuernberger* Kuo-Chin Lien† Tobias Höllerer‡ Matthew Turk§

University of California, Santa Barbara



(a) Original drawing (b) Spray-paint (c) Median depth plane (d) Dominant plane (e) Gesture enhanced

Figure 1: Alternative 3D interpretations (b-e) of the original 2D drawing (a) from different viewpoints. Previous methods (b-d) may not adequately convey the user's intention of referring to the printer compared to our gesture enhanced method (e).

Abstract

A 2D gesture annotation provides a simple way to annotate the physical world in augmented reality for a range of applications such as remote collaboration. When rendered from novel viewpoints, these annotations have previously only worked with statically positioned cameras or planar scenes. However, if the camera moves and is observing an arbitrary environment, 2D gesture annotations can easily lose their meaning when shown from novel viewpoints due to perspective effects. In this paper, we present a new approach towards solving this problem by using a *gesture enhanced* annotation interpretation. By first classifying which type of gesture the user drew, we show that it is possible to render the 2D annotations in 3D in a way that conforms more to the original intention of the user than with traditional methods.

We first determined a generic vocabulary of important 2D gestures for an augmented reality enhanced remote collaboration scenario by running an Amazon Mechanical Turk study with 88 participants. Next, we designed a novel real-time method to automatically handle the two most common 2D gesture annotations—arrows and circles—and give a detailed analysis of the ambiguities that must be handled in each case. Arrow gestures are interpreted by identifying their anchor points and using scene surface normals for better perspective rendering. For circle gestures, we designed a novel energy function to help infer the object of interest using both 2D image cues and 3D geometric cues. Results indicate that our method outperforms previous approaches by better conveying the meaning of the original drawing from different viewpoints.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles

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

IEEE Symposium on 3D User Interfaces 2016
19–23 March, Greenville, SC, USA
978-1-5090-0842-1/16/$31.00 ©2016 IEEE

1 Introduction

Annotating the physical world in augmented reality (AR) is a useful way to convey information. While there are many types of annotations for augmented reality [29], 2D drawing annotations provide one of the simplest ways to convey information, especially in applications where collaboration plays an important role. However, rendering 2D drawing annotations in 3D is not straightforward. When making simple gestures via 2D drawings, previous approaches typically render them using a spray-paint metaphor (Figure 1b) or render them onto 3D planes (cf. Figures 1c and 1d). This only works in highly constrained scenarios, such as when the camera does not move or with planar scenes. However, in unconstrained scenarios these techniques suffer from perspective effects and therefore do not optimally convey the intended information. So far, the problem of interpreting and rendering such 2D gesture annotations in arbitary 3D environments, such that they still convey the intended information when seen from different viewpoints, has been unsolved.

In this paper, we take a fresh approach toward solving this problem and focus on 2D gesture annotations often used in collaboration scenarios [3, 15]. We argue that not all 2D gesture annotations should be handled the same way, and therefore we use a gesture classifier to first determine what the user has drawn. Based on this classification, our method then takes appropriate steps to enable a meaningful interpretation and rendering of the annotations in 3D augmented reality.

To limit the scope of the problem, we use augmented reality enhanced remote collaboration [4, 6, 25] as a motivating application. In a typical scenario, a domain expert is assisting a novice user with a task that involves the physical environment. The domain expert is not physically present with the novice user, and therefore the domain expert is referred to as the "remote user." The novice user, also known as the "local user," has a camera that sends a live video feed to the remote user. The remote user helps the local user to complete the task by using verbal interaction and sending appropriate annotations back to the local user; such annotations are then displayed to the local user via an augmented reality display.

While other gestures, such as in-air hand gestures [15, 25], are arguably more natural or intuitive for remote collaboration, there still remain several motivations for using 2D drawing annotations for gesturing. First, humans typically learn to draw from a very young age, and therefore interacting with this portion of the user interface requires little to no training time. Second, drawings have

been used successfully in many instructional manuals and thus are very appropriate for remote collaboration [17]. Third, when the remote user has a drastically different viewpoint than that of the local user (cf. [27]), 2D annotations may be better for deictic gesturing than other gesturing methods. A significant problem with the remote user gesturing with his/her hands for deictic gestures is that the remote user must have the same viewpoint as that of the local user, otherwise the hand gestures may not be interpretable. Gauglitz et al. [5] also list several benefits of using 2D input instead of 3D input. Furthermore, Kim et al. [14] recently compared pointing annotations to drawing annotations and found that drawing annotations require less input from the remote user and less cognitive load on the local user than pointing annotations. Finally, more natural user interfaces, such as in-air gesturing, are often less ergonomical, leading to fatigue [8].

Most previous research on using 2D drawing annotations for remote collaboration employed only statically positioned cameras and used screen-stabilized annotations [3, 15]. The reason for this was obvious—once the camera moved, the drawing on the video feed was no longer positioned correctly with respect to the physical world. In order to be able to correctly interpret the screen-stabilized annotations, the local user had two choices: either maintain a viewpoint close to that of the remote user, or look at another screen or screenshot that showed the remote user's drawing viewpoint [15].

To overcome this limitation, recent research has used computer vision tracking in AR to avoid statically positioned cameras, thus enabling 2D drawing annotations to be world-stabilized [5, 13]. However, when the local user or remote user changes the viewpoint (e.g., to increase collaboration effectiveness [27]), such 2D drawing annotations may no longer adequately convey the intended information when using existing methods (Figure 1).

With the increasing trend of using AR for collaboration, enabling 2D drawing annotations to be rendered in 3D such that they convey their original intention from any viewpoint is an important challenge. Our main two contributions in this paper are:

1. An Amazon Mechanical Turk study consisting of 88 participants, showing that arrows and circles are the two most important 2D drawing annotations to handle.

2. A novel approach to handle these two most important 2D drawing annotations, interpreting and disambiguating appropriately for each. We term this approach, a "gesture enhanced" interpretation or rendering method for 2D drawing annotations in 3D AR. Our results indicate the method better conveys the original meaning of the drawing annotation from different viewpoints compared to previous methods.

2 Related Work

Our work broadly falls under the category of image plane interaction techniques for 3D environments [21]. The research most closely related to our work is in the area of 2D drawing annotations for 3D environments.

2.1 2D Drawing Annotations in 3D Environments

Previous approaches using 2D drawing annotations have largely focused on either a graffiti/spray-paint approach (cf. Figure 1b) [5, 7, 12, 28] or a planar approach (cf. Figure 1c, 1d) [1, 5, 12, 13, 22, 28].

Space Pen by Jung et al. [12] allows users to collaborate in a virtual environment by drawing in 3D. It supports drawing onto surfaces in a graffiti or spray-paint like manner or onto a user-created 3D plane. Space Pen also detects specific gestures, causing relevant actions to be performed, such as creating a door from a user drawn rectangle on a wall. Our work can be seen as a reinvigoration of this approach for augmented reality based remote collaboration.

Boom Chameleon by Tsang et al. [28] also provides both a spray-paint and planar approach. It allow users to save the 2D drawing annotations on screenshots that are displayed as 3D planes. They also describe a problem they call "ink drift," which occurs when the user is moving the viewpoint and the drawing simultaneously drifts as a result; they avoid this problem by using screenshots. For remote collaboration, Gauglitz et al. [5] temporarily freeze the viewpoint while the remote user is drawing and then transition back to the same viewpoint of the local user.

Other work mainly focused on placing 2D annotations onto 3D planes. Poupyrev et al. [22] determine the position and orientation of such a 3D plane via physically moving a computer tablet. Several works place 2D drawing annotations on a 3D plane parallel to the image plane at which the drawing was created [1, 5, 13]. Bourguignon et al. [1] place the 3D plane at the depth that contains the world origin; Kasahara et al. [13] use a fixed offset from the camera; and Gauglitz et al. [5] used a statistic of the different depths of the 2D annotation (cf. Figure 1c). Bourguignon et al. [1] further use the beginning or end of the stroke in relation to existing 3D geometry to determine where to place the annotation in 3D. Gauglitz et al. [5] also investigated fitting a 3D plane to the points, the so-called "dominant plane" approach (cf. Figure 1d). In our approach, we render 2D arrow and circle annotations in screen-space instead, using 3D points only as an "anchor" for the screen-space rendering.

2.2 Other Related Work

Although their goals are different, 2D sketch-based modeling interfaces [11, 20] and 2D gesture annotation interfaces both require inferring 3D information from a user's 2D strokes. Some approaches use only user drawn sketches to create 3D models from scratch [11], while others use multi-touch interaction to deform already existing 3D models [20]. Olsen et al. [18] provide a survey of sketch-based modeling interfaces.

Another approach is to use specific annotation tools to fit to various scene properties [23]. Methods like this could be molded into being used in a 2D drawing gesture application.

3 User Study

In order to design a method to correctly render different 2D annotations in augmented reality for remote collaboration, we designed a user study to: (1) help us determine a vocabulary of 2D gesture annotations that our system should handle; and (2) see if people draw different types of gestures based on task and viewpoint. Based on our observations of previous AR-based remote collaboration systems (*e.g.*, [5]), we hypothesized:

1. The three most frequently used annotations for referencing objects or locations will be circles, outlines, and arrows.
2. Users will draw more arrows for action tasks compared to referencing tasks.
3. Users will draw more arrows when objects or locations are at oblique angles.

We use the term *circle* loosely to include ellipses and any kind of closed loop drawing that roughly resembles a circle. We denote an *outline* as a drawing that appears to follow the outline (*i.e.*, border) of the object.

We used the overall problem of changing printer cartridges in an HP Color LaserJet CP2025, since this was a simple task that involves the physical environment, with both referencing and action tasks. There was a total of 7 tasks (4 referencing tasks and 3 action tasks, as occurred natually in changing printer cartridges) with saved images (screenshots) from different viewpoints in each case. The tasks were worded as simple questions so that participants could quickly understand what was being asked of them to convey to a hypothetical collaborator:

1. Where is the printer? (20 images)
2. Where is the printer door? (20 images)
3. Open the printer door. (20 images)
4. Where are the printer cartridges? (21 images)
5. Pull out the printer cartridges. (21 images)
6. Where is the red cartridge? (24 images)
7. Pull out the red cartridge. (24 images)

In general, viewpoints spanned angles looking straight at the printer (0°), 45° to the left/right of the printer, 90° to the left/right of the printer, at varying distances, and varying vertical angles; finally, several "other" viewpoints were included such as close-ups, top views, and a behind view from which the printer door was not able to be seen. Due to the restriction of our system requiring static scenes for the 3D modeling (cf. Section 5), we recorded three different models (tasks 1-3, 4-5, and 6-7) and thus had differing numbers of images for those groups of tasks.

To accomplish the objective of this study, we conducted a supervised within-subjects pilot study, and a larger, an unsupervised between-subjects study with a larger number of participants using Amazon Mechanical Turk (AMT)[1]. In the pilot study, participants used a 15.6" Lenovo U530 touchscreen laptop (i7-4510 CPU, 8 GB RAM). In the AMT study, participants were free to use whatever input device they liked. We used the JavaScript libraries Three.js and jQuery, and PHP [2] to record user input.

3.1 General Procedure

Participants were first given a pre-study questionnaire to gather demographic data. Next, they were told to pretend that they were in a video chat with their friend and that they were to draw onto the screen to help their friend solve a task. To help participants get acquainted with the study setup, a brief training task was performed. After this, they were shown a video of how to change the printer cartridges, in order to familiarize them with the study's actual task. Next, participants began the actual task. In the within-subjects pilot study, tasks were shown in the order of three models, whereas in the between-subjects main study, participants were limited to perform one task. Images and task for each model were randomized to avoid potential bias effects. Participants were given the liberty to draw in whatever way they wished to complete the task. After finishing the task(s), participants filled out a post-study questionnaire.

3.2 Supervised Pilot Study

Before running the actual study, we conducted a small supervised pilot study of 8 participants (5 female, 3 male; ages 18 to 22). The purpose of the supervised pilot study was to get an initial idea of the types of 2D gestures users draw for remote collaboration and to fine tune the procedure for the larger study. Participants were paid $10 USD for participating in the study which lasted no more than one hour. Since this was a supervised study, to help make sure we understood the participants' drawing intentions, we asked them to describe aloud what their drawing was meant to convey.

There was a total of 1,202 drawings after removing out outliers. Due to a small number of users, we only briefly report percentages of gestures drawn: 59.5% of drawings included arrows, 43.1% included circles, 11.9% included precise outlines. Based on our observations and results, we fine-tuned the study setup for the AMT study.

[1] http://mturk.com

[2] http://threejs.org; http://jquery.com; http://php.net

Gesture	Action (%)	Reference (%)
Arrow	519 (73.8)	467 (40.8)
Other	184 (26.2)	677 (59.2)

Table 1: Numbers and percentages of arrow and non-arrow gestures drawn for different types of tasks.

Gesture	0° (%)	45° (%)	90° (%)	Other (%)
Arrow	184 (50.1)	309 (50.2)	317 (58.2)	176 (55)
Other	183 (49.9)	306 (49.8)	228 (41.8)	144 (45)

Table 2: Numbers and percentages of arrow and non-arrow gestures drawn for different viewpoints.

3.3 Unsupervised Amazon Mechanical Turk Study

In order to efficiently obtain a larger experimental sample, we conducted a larger user study via Amazon Mechanical Turk, an online crowdsourcing marketplace. Participants were paid $0.20 USD and were limited to perform one task, taking on average 17 minutes and 33 seconds to complete the task. Since this study was unsupervised, participants were asked to type a description of each of their drawings. Each participant was required to be from the United States of America and was allowed to use any type of input device.

88 users participated in the study (37 male, 51 female), ages from 19 to 66 (average 33.8 years old). 80 stated English as their native language, 4 with 5-10 years of experience, 2 with 2-5 years, and 1 with less than 2 years[3]. 38 had corrected vision (*e.g.*, nearsightedness) and 49 had no vision impairments. 23% were very familiar with 3D software, 34% somewhat familiar, 23% barely familiar, and 20% not familiar. 88% used a computer mouse at least several times a week, 77% a trackpad, and 89% a touchscreen. 25% stated being familiar with the concept of AR, 43% had heard of AR, and 32% did not know what AR was.

There was a total of 1,847 drawings after removing outliers (*e.g.*, participants stated images did not load, did not draw, etc.). The experimenter manually went through each image and indicated whether or not a specific type of 2D drawing gesture was shown. If the drawing was obscure, we followed the terminology used by the participant in his or her description of the drawing. The results are aggregated in Tables 1 & 2 and Figures 2 through 4. Note that the columns do not sum up to 100% in the Figures 2 through 4 because, *e.g.*, users could draw both arrows and circles in the same drawing.

Figures 2 through 4 appear to confirm hypothesis 1: in 89% of the referencing tasks' drawings, users either drew arrows, circles, outlines, or a combination of those. To statistically test for hypothesis 1, a multinomial distribution of the occurrences of the nine categories of gestures was assumed. The top three gestures according to the fitted model were arrows (35.5%), circles (36.7%), and outlines (9.7%), with the χ^2 goodness of fit test being strongly not significant, showing that it fits the data well (the null hypothesis). We also applied a Bayesian approach to model the frequencies of the nine types of gestures as being generated from a multinomial distribution. Considering the top two and top three gestures, with 10^6 draws from the Dirichlet distribution (conjugate prior of the multinomial), arrows + circles occurred 10^6 times (100%) for the top two gestures, and arrows + circles + outlines occurred 999,848 times (99.98%) for the top three gestures; arrows + circles + words was the only other top three combination, occurring 152 times (0.02%). Based on these analyses, hypothesis 1 is confirmed.

To test hypotheses 2 & 3, logistic regression was performed. Task was categorized into 2 levels—action and referencing; view-

[3] Due to the AMT survey questionnaire format that was utilized, some users were able to skip some questions and thus in several cases there were only 87 user responses instead of 88 total.

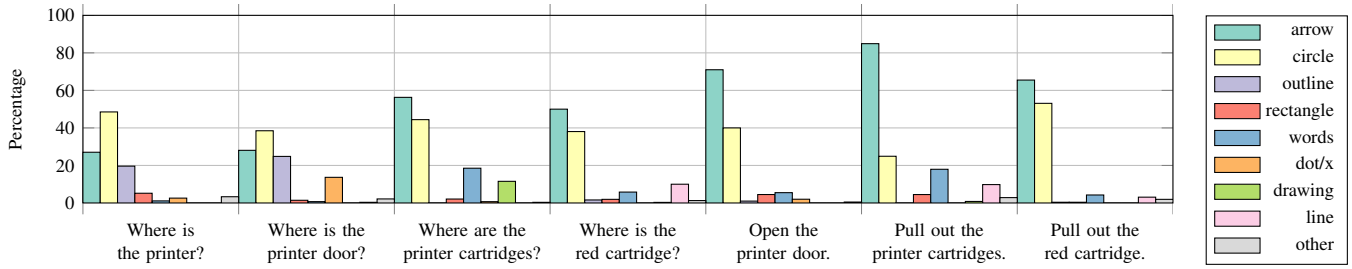


Figure 2: Percentages of types of gestures drawn subdivided by task (the first four are referencing tasks and the last three are action tasks). Each user in the AMT study only did one of the above tasks. Overall there were 1,847 drawings across 88 users.

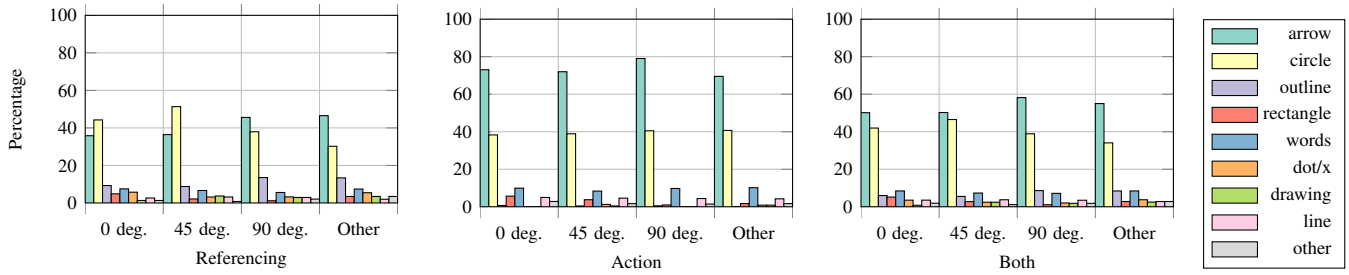


Figure 3: Percentages of types of gestures drawn subdivided by viewpoint and by type of task (referencing or action task, or both). Figure 1a is an example of a 0° viewpoint, Figure 6 is an example of a 45° viewpoint, and Figure 1e is an example of a 90° viewpoint.

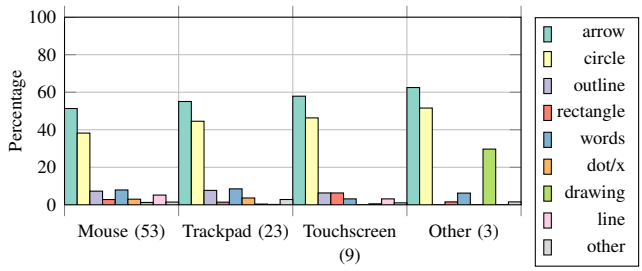


Figure 4: Percentages of types of gestures drawn subdivided by input device. The numbers of users for each device are reported in parenthesis.

Figure 6: This arrow, drawn by an Amazon Mechanical Turk user, could either be: (1) referring to the wire in the background, (2) indicating a "pulling" gesture (actual intention), or (3) indicating a relationship between the printer and the wire in the background.

points into 4 levels—0°, ±45°, ±90° (cf. Figures 1a, 6, and 1e, respectively), and other; and input device into 4 levels—mouse, trackpad, touchscreen, and other (*i.e.*, user-reported "Wacom drawing tablet" or "tackball," cf. Figure 4). The dependent variable was the binary outcome of the whether or not the user drew an arrow. Tables 1 and 2 report the actual numbers of arrow and non-arrow gestures drawn.

The odds ratio for drawing an arrow while performing an action task compared to while performing a referencing task was statistically significant at 4.671 (p-value < 0.001; coeff. 1.541; std. error 0.112), thus supporting hypothesis 2. Pairwise comparisons, with Bonferroni correction applied, indicated a statistically significant difference in the odds ratio of drawing an arrow at 90° compared to both 45° and 0° viewpoints, with odds ratios respectively of 1.464 (p-value 0.002; coeff. 0.382; std. error 0.126) and 1.456 (p-value 0.009; coeff. 0.376; std. error 0.144), supporting hypothesis 3. However, we note that the effect size for hypothesis 3 is relatively small and future studies should investigate this further.

4 Anchoring 2D Gesture Annotations in 3D

Based on the user study results, we chose to focus on handling the top two gesture annotations—arrows and circles. In the following, "anchor" refers to determining a way to world-stabilize a 2D annotation so that from any viewpoint, the rendered annotation still conveys the intended information from the original drawing. The input to the method is a sequence of (x, y) coordinates defining the user's gesture, and the output is a gesture classification and rendering of that gesture from any viewpoint. Our system utilizes an already existing 2D gesture classifier (see Section 5).

4.1 Arrows

There are at least three main challenges for anchoring 2D arrow gestures in 3D space.

First, there is an ambiguity associated with interpreting arrows drawn on pictures without contextual information. For example, there are at least three logical ways to interpret the arrow in Figure 6: (1) the arrow is pointing at a wire plugged into the electrical

(a) Original drawing (b) Regularized arrow (c) Arrow on 3D plane, anchored at head (d) Gesture enhanced

Figure 5: Illustration of arrow regularization, a naive rendering onto a 3D plane, and our gesture enhanced rendering based on surface normals.

Figure 7: This arrow, drawn by an Amazon Mechanical Turk user, is meant to refer to the printer. However, anchoring the arrow in 3D based on the location of the arrow head will not work since the 3D point underneath the arrow head tip is on the door, not the printer.

outlet; (2) the arrow is indicating a "pull" gesture with respect to the printer; or (3) the arrow is indicating a relationship between the blue tab on the printer and the electrical outlet in the background. In the first case, we desire to anchor the arrow at what its head is referring to; in the second case, we desire to anchor the arrow at what its tail is referring to; and in the third case, we desire to use both the arrow's head and tail to anchor it in 3D space.

Based on the drawings from the AMT study, we noticed that 90.5% of arrows for the referencing tasks were anchored on their heads. For action tasks, 72.9% of arrows were anchored on their tails; even though all actions are related to "pulling," some participants used arrows to still refer to the object. To handle this ambiguity, we make the assumption that most times users will want the arrow to be anchored at what its head is pointing to. To anchor an arrow on its tail, we give users two options—manually specifying to anchor on the tail or drawing another annotation (*e.g.*, a circle) near the desired anchor point. If the screen-space distance from the arrow's tail to the annotation is closer than the screen-space distance from the arrow's head to the annotation, anchoring is instead based on the arrow tail. We currently do not handle the third case of anchoring both on head and tail, and leave this as future work.

To determine the location of the arrow's head, we search for large changes in the direction of the 2D drawing [19]. Currently our implementation handles one-stroke arrows, drawn either starting from the tail or the head. For arrows drawn from the tail, we detect the first two vertices by finding a change in direction greater than 90°, and the third vertex (the arrow head) by finding a change in direction greater than 45° to account for noisy arrow drawings[4].

Second, after determining which part of the arrow to use as an anchor point, there still may be an ambiguity for determining the exact 3D anchor point. This is illustrated in Figure 7. 100 of the 986 arrow drawings (10.14%) in the AMT study exhibited this characteristic. However, only 18 (1.83%) of these had arrows whose head was greater than 10 pixels away from the anchor point (12 of which were drawn from one user). Thus, the vast majority of users drew arrows directly onto the object. Based on this, we use the closest foreground object within a small search region near the arrow head as the 3D anchor point and the new arrow head[5].

Third, once we have identified where to anchor the arrow, we need to render it appropriately for a given viewpoint. We could naively render the arrow onto a 3D plane whose orientation is orthogonal to the drawing viewpoint's optical axis and whose depth is at the anchor point (motivated by previous approaches [1,5,13]), but this approach still suffers from perspective effects when seen from a drastically different viewpoint as shown in Figure 5c. Ideally, we would like to render the arrow closely to how the user originally drew it when the rendering camera is near the original viewpoint and also render the arrow appropriately at farther away viewpoints.

To achieve this, we utilize the unit surface normal vector $\mathbf{n}$ at the 3D anchor point for the arrow head[5] to determine the 3D direction $\mathbf{d}$ of the rendered tail as follows:

$$\mathbf{d} = \alpha \cdot \mathbf{n} + (1-\alpha) \cdot \frac{\mathbf{t}-\mathbf{h}}{\|\mathbf{t}-\mathbf{h}\|} \quad (1)$$

Where α linearly increases from 0 to 1 as the distance between the rendering viewpoint and original drawing's viewpoint increases, reaching 1 after a short distance threshold; $\mathbf{h}$ is the 3D head point back-projected from the 2D head point; and $\mathbf{t}$ is the 3D tail point back-projected from the 2D tail point. Using α allows us to render the arrow tail to be pointing closely to how the user originally drew it whenever the live viewpoint is close to the original drawing's viewpoint (cf. Figures 5a and 5b); this is important since we eventually render the arrow in screen-space in a regularized form (see below) and thus do not want to immediately render the arrow in a drastically different way from which the user drew the arrow.

The head $\mathbf{h}$ and tail $\mathbf{t}$ are projected from 3D into 2D as follows:

$$\mathbf{h}_{2D} = proj(\mathbf{h}) \quad (2)$$

$$\mathbf{t}_{2D} = proj(\mathbf{h} + \lambda \mathbf{d}) \quad (3)$$

Where $proj$ applies the rendering camera's model-view-projection matrix, projecting 3D points to 2D points; and $\lambda = \|\mathbf{h}-\mathbf{t}\|$ is the length of the tail.

To handle cases where the angle θ between the rendering viewpoint's principal axis and the surface normal $\mathbf{n}$ is close to 180° (*i.e.*, parallel), we further adjust the rendered tail after projection to 2D as follows, to represent the direction into the view:

$$\mathbf{t}_{2D} = \beta \cdot (\mathbf{h}_{2D} + \mathbf{t}_{offset}) + (1-\beta) \cdot \mathbf{t}_{2D} \quad (4)$$

[4]For an arrow drawn from its head, we simply reverse the sequence of (x,y) coordinates and run the same procedure.

[5]Anchoring on the arrow tail takes an analogous approach.

(a) Original drawing (b) Segmented 3D points (c) Gesture enhanced (d) Segmented 3D points (e) Gesture enhanced

Figure 8: Illustration of our 2D-3D co-segmentation and gesture enhanced circle rendering from different viewpoints.

Where $\beta \in [0,1]$ is 0 when $\theta < \phi$ (we found that 150° works sufficiently) and between 0 and 1 otherwise, inversely proportional to $\cos\theta$; and $\mathbf{t}_{offset}$ moves the tail to be vertically below $\mathbf{h}_{2D}$ in screen-space (we used 150px below $\mathbf{h}_{2D}$ in our implementation).

Arrows are rendered in screen-space in a "regularized" form drawing a line between $\mathbf{h}_{2D}$ & $\mathbf{t}_{2D}$ and adding two lines for the head of the arrow. An illustration of the approach is shown in Figure 5.

4.2 Circles

Figure 1 illustrates how spray-paint and planar approaches do not adequately convey the original intention of 2D circle annotations when rendered from different viewpoints. Our new approach is illustrated in Figure 8.

Previous methods are not able to appropriately transfer circle annotations to different viewpoints. To interpret the intention of the drawing, they merely estimate the depth of the annotation stroke points but do not fully take advantage of the rich 2D image and 3D geometric information that can be obtained from the modeled scene. In this paper, we achieve the circle annotation transfer by (1) first extracting the 2D convex hull of the original drawing; (2) using this as a prior for extracting the object of interest, both in 2D image space and in 3D space, using both 2D and 3D cues (we refer to this step as 2D-3D co-segmentation); and (3) finally, generating a new annotation for each viewpoint based on this object extraction result. Specifically, we extract the object of interest by minimizing the following energy function:

$$E = E_{2D}(\mathbf{P}, T(\mathbf{Q})) + E_{3D}(T^{-1}(\mathbf{P}), \mathbf{Q}), \tag{5}$$

Here, the optimization goal is to label $\mathbf{P}$ and $\mathbf{Q}$ to be foreground or background, where $\mathbf{P}$ are the 2D points in the user-annotated frame I and $\mathbf{Q}$ are the 3D points in the model. T is a transformation that projects $\mathbf{Q}$ to the image plane, and T^{-1} projects $\mathbf{P}$ back to the 3D space. E_{2D} is an energy term to ensure good 2D segmentation quality by, *e.g.*, maximally separating the color difference between foreground and background. E_{3D} is a convexity-based term to encourage the segmentation result to be a convex hull in the 3D space where the transition from convex to concave parts is more likely the separation point between objects [26].

We adopt a piecewise optimization strategy to efficiently solve Equation (5), *i.e.*, minimizing one term first and then the other, iterating between the two. Note that although we do not pre-train the color models, the convex hull obtained from fitting the user's input drawing helps make a good initialization such that minimizing the first term can resort to expectation-maximization style solvers. GrabCut [24] was used for solving this 2D term in our implementation. For solving the second term, it is computationally expensive to explicitly calculate the convexity of every potential foreground configuration. We instead use the method of Stein et al. [26] directly, which only evaluates convexity locally to determine if the foreground region can grow to its neighboring points.

For rendering the gesture enhanced circle, we project each point in $\mathbf{Q}$ into 2D screen-space (Figures 8b, 8d) and then fit an ellipse to these points using Fitzgibbon and Fisher's method [2] (Figures 8c, 8e). Fitting the ellipse to all the points enables a temporally smooth visualization, but it also causes the fitted ellipse to be smaller than the point of interest in many cases. To adjust for this, we scaled the ellipse by 125% which was a simple but effective solution.

5 Implementation & Evaluation

We followed the approach by Gauglitz et al. [6], using a monocular SLAM system on a Nexus 7 tablet to stream a video to a commodity desktop PC, which builds a sparse 3D model using the approach of Hoppe et al. [10]. To isolate the particular aspect of evaluating our method to interpret 2D gesture annotations in 3D, we used the final 3D model with its saved keyframe images; this allowed us to do a more controlled evaluation without introducing confounding factors due to a live, updating model and varying viewpoints. To assign a depth value to pixels corresponding to an unmodeled area, we used the average depth from that viewpoint. A variation of the $1 Recognizer [30] was used for gesture classification.

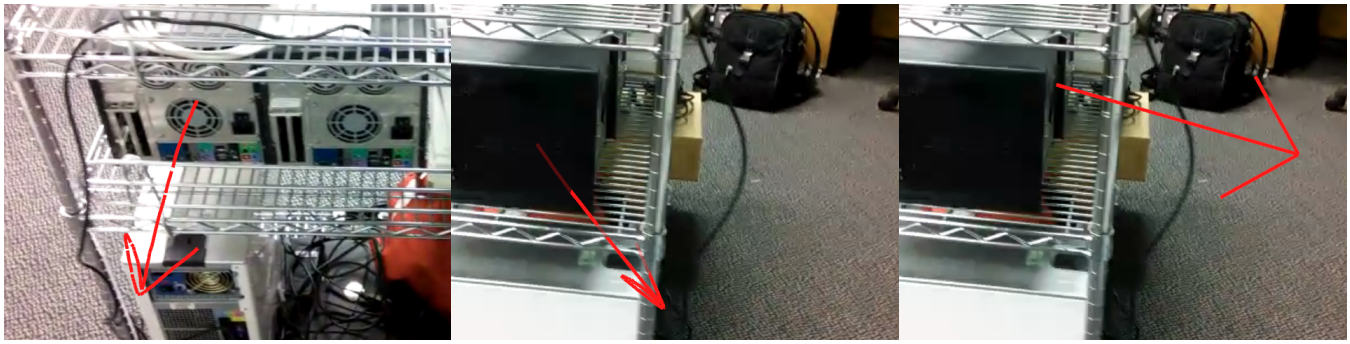
We experimented performing the circle segmentation with both the RGB color image and the depth map image, and we found that the depth image is a better approach when the underlying 3D model is dense. In cases where the user drew into a homogeneous part of the image causing GrabCut [24] to oversegment and return an empty region, we simply ignored the segmentation and used the convex hull of the user drawn circle.

To evaluate our novel arrow and circle annotation methods, we compared our methods against the median depth plane interpretation since we considered it the most competitive method introduced in previous work [5]. Temporal smoothness of the rendered gesture enhanced arrows and circles when transitioning between different viewpoints can be seen in the supplemental video.

5.1 Gesture Enhanced Arrows Evaluation

To evaluate our arrow anchoring method, we conducted another AMT study where we showed each participant 16 images of a user-drawn arrow, in which 8 indicate a particular object in the scene and 8 are associated with an action. For each image, we showed an additional 2 images side-by-side from a different viewpoint, showing the gesture enhanced and median depth plane annotation transfer interpretations (in random order). Half of these additional images had an approximately 45° change in viewpoint and half with an approximately 90° change. Examples of these images are shown in Figure 9. We asked the participants, "Which image (left or right) best conveys the same meaning as the drawing in the first picture?" With 320 votes generated by 20 participants, 243 (76%) votes chose the gesture enhanced arrows to better convey the meaning of the original drawing, whereas only 77 (24%) votes chose the median depth arrows. The results in Table 3 show that our proposed method is favored over the median depth method for both referencing and action tasks. Our method was chosen over the median depth method in all pairs of images except for the one shown in Figure 10 (tied

(a) Original drawing (b) Median depth plane (c) Gesture enhanced



(d) Original drawing (e) Median depth plane (f) Gesture enhanced

Figure 9: Arrow renderings from different viewpoints used in Section 5.1, comparing the median depth plane [5] and gesture enhanced interpretations for two action tasks.

	Referencing	Action	Total
Gesture enhanced	130 (81%)	113 (71%)	243 (76%)
Median depth	30 (19%)	47 (29%)	77 (24%)

Table 3: The arrow AMT study results confirm that our proposed gesture enhanced arrows better convey the user's annotations from different viewpoints for both referencing tasks and action tasks.

10 to 10), which had the smallest change in viewpoint among all 16 pairs of images.

5.2 Gesture Enhanced Circles Evaluation

To evaluate our 2D-3D co-segmentation method for anchoring circle annotations, we chose to perform a more objective evaluation procedure. We first manually marked the ground-truth objects in five drastically different viewpoints for three different 3D models we recorded. Based on the ground-truth labeling, we generated 10 ellipses for each view, with $\pm 10\%$ random variation on the axes and ± 5 pixels random variation on the centers, to simulate the user input circle annotations and then transferred these annotations to the remaining other 4 views. We evaluated these 10 ellipses $\times$ 5 source views $\times$ 4 transfer views $\times$ 3 models $= 600$ annotation transfer results by filling the circles we rendered and checking how well they overlap the ground-truth object in terms of the popular intersection-over-union (IoU) score used in image segmentation benchmarks. Table 4 shows a quantitative comparison between the 2D-3D co-segmentation method and the median depth method. For all three models, our gesture enhanced circles achieve a better IoU score than the median depth method.

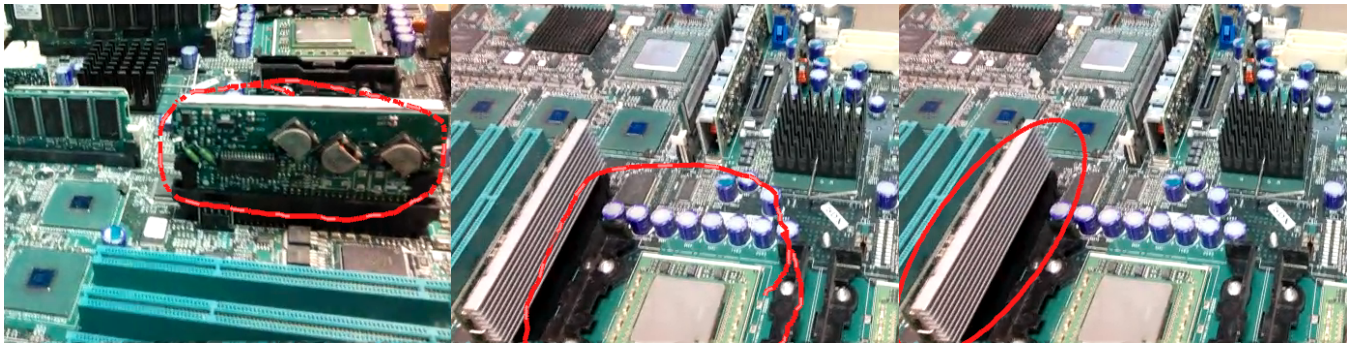
Additional results are shown in Figures 11 and 12 as well as in the supplemental video, showing that our method is able to handle varying types of scenes appropriately.

(a) Original drawing (b) Median depth plane (c) Gesture enhanced

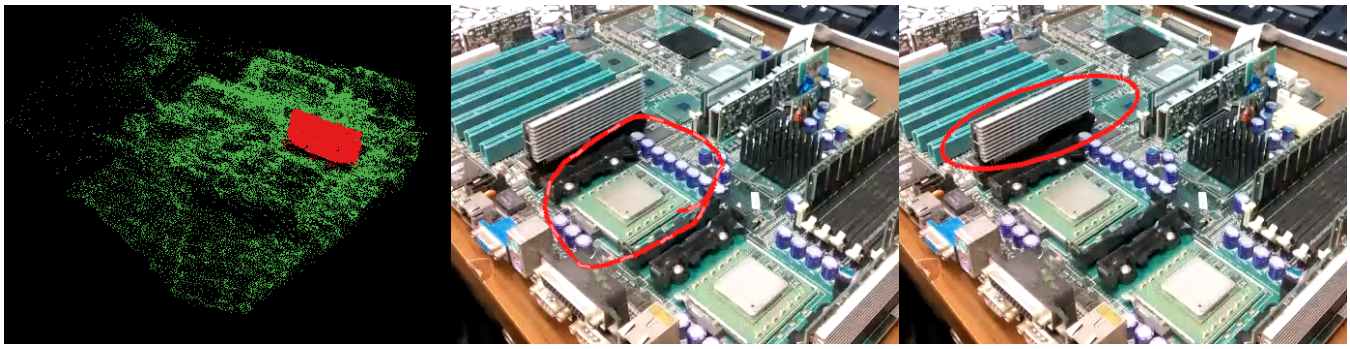
Figure 10: A side-by-side comparison task in the arrow evaluation AMT study – "Pull out this spark plug." This comparison had the smallest viewpoint change and also was the only one in which our gesture enhanced method was not chosen more than the median depth plane method (tied 10 to 10).

5.3 Timings

The proposed method works in real-time, with the following average timings from a desktop PC (Intel Core i7-950 CPU @ 3.07GHz, NVIDIA Quadro K5000 GPU). Gesture classification and anchoring arrow annotations both take less than 1 ms. Anchoring circle annotations takes 1.09 seconds (most of the time being used by GrabCut [24]). The classification and anchoring steps are only run once for each input gesture on the remote user's side, the results of which would then be sent over the network to the local user. Therefore, with our unoptimized implementation, there is currently about one second of lag for sending circle annotations to the local user. Rendering a gesture enhanced arrow takes less than 1 ms per frame, and an ellipse rendering takes about 5 ms per frame (due to fitting the ellipse to the projected 3D points at each viewpoint [2]).



(a) Original drawing (b) Dominant plane (c) Gesture enhanced



(d) View of 3D segmentation (e) Dominant plane (f) Gesture enhanced

Figure 11: Circle rendering from different viewpoints, comparing the dominant plane [5] and gesture enhanced interpretations. Note that in this example, the dominant plane interpretation drastically and incorrectly changes what the original drawing had intended to convey, whereas the gesture enhanced interpretation correctly maintains the original drawing's meaning.

	Printer	PC	Crunch
Gesture enhanced	49.50	40.85	52.62
Median depth	43.67	19.28	29.85

Table 4: A quantitative comparison of the proposed gesture enhanced circles and the median depth circles in terms of the average IoU score (in %). Our proposed method generates better ellipses to cover the selected object from different viewpoints.

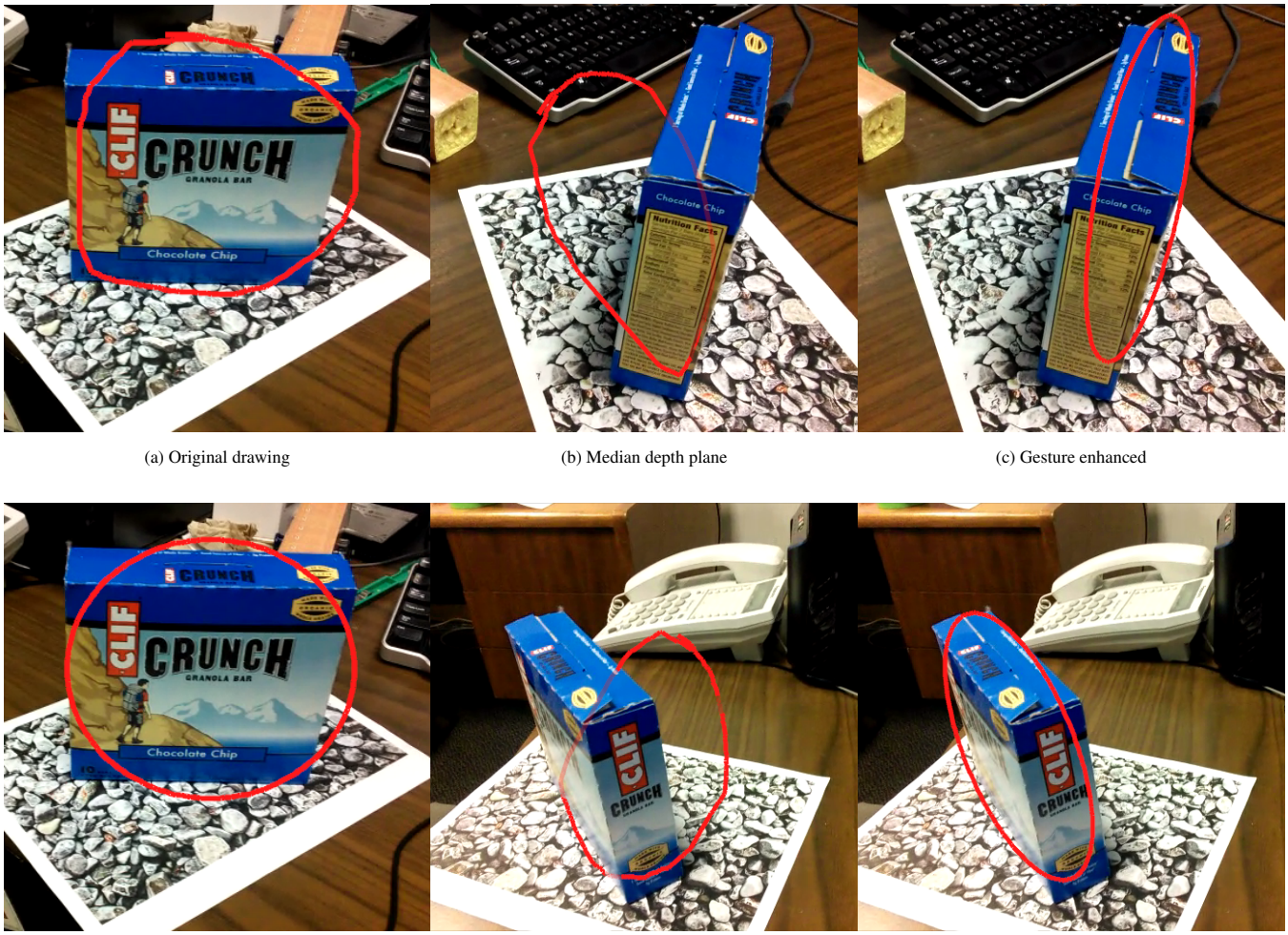
6 Discussion & Future Work

Our objective in this paper was to determine if we can appropriately interpret 2D gesture annotations for rendering from different 3D viewpoints in order to better convey the original drawing's intentions. This is an important problem, especially given recent research results that show how view independence in collaborative AR systems can lead to faster task completion times, increased user confidence, and decrease in the amount of verbal communication needed to complete the task [27]. Due to the inherent lack of information when going from 2D to 3D, we emphasize that it is impossible to infer with full assurance what the user was intending to convey when drawing a 2D annotation for a three-dimensional world. Our method is a first approach to solving this problem for 2D gesture annotations.

In this paper, our focus was not on performing a detailed usability evaluation for AR collaboration; instead, we wanted to isolate the particular aspect of appropriately rendering 2D gesture annotations in 3D for the goal of conveying important information (*e.g.*, for remote or asynchronous collaboration). While we only handled the top two gesture annotations (arrows and circles), future work should investigate more sophisticated drawing annotations, such as drawn text, distinguishing curved arrows from straight arrows, arrows anchored on both tail and head, simple lines and curves, etc. Gesture enhanced arrows are rendered orthogonally to the surface, and while this may work for referencing tasks, future work should examine how to render arrows appropriately for action tasks when the angle of the arrow is important to convey. Whether or not animation of the original drawing makes a difference to interpreting the drawing from other viewpoints or for collaboration purposes is another area of future work (cf. [5]). It would also be interesting to see whether or not users care that we "regularize" the arrows and circles (cf. Figures 5 and 8).

As with any method that relies on a 3D reconstruction of the world, our method can also perform suboptimally with poorly reconstructed scenes. Although arrow anchoring relies on surface normals being properly reconstructed, in our experience, we found that even many sparse reconstructions [10] work sufficiently for our algorithm. However, for more sophisticated interpretations of 2D gestures in 3D (e.g., an arrow indicating the action of rotating a cylindrical pipe), more accurate and dense reconstructions will most likely be necessary.

For circle anchoring, the main limitation is in assuming that the object of interest is convex, and the 3D reconstruction must be able to capture this convexity property. Although the convexity prior has its theoretical foundation in psychophysical literature (*e.g.*, [9]), its study in the field of AR-based remote collaboration is only beginning. For future work, more validation is required with different types of 3D reconstructions, and more 3D geometric priors could be integrated into our 2D-3D co-segmentation approach to hopefully better answer the fundamental question: What is an object in 3D, particularly for different types of 3D reconstructions (*i.e.*,



(a) Original drawing (b) Median depth plane (c) Gesture enhanced

(d) Gesture enhanced (e) Median depth plane (f) Gesture enhanced

Figure 12: Circle rendering from different viewpoints, comparing the median depth [5] and gesture enhanced interpretations.

sparse, semi-dense, or dense)?

Handling 2D circle annotations has also unveiled another flavor of the image segmentation problem: given a 2D image with a depth map and a segmentation of an object of interest, how can one obtain the correct image segmentation of the same object from a new viewpoint? The main problem here, as with traditional object tracking scenarios, is that certain parts of the object of interest may be occluded from the first viewpoint; however, in our case, since the local user is not necessarily viewing the scene from the same viewpoint as that of the remote user, we cannot immediately apply traditional object tracking methods to obtain the new segmentation. We believe our novel 2D-3D co-segmentation algorithm is a first step towards solving this problem. By considering 2D image features from multiple viewpoints, interactive multi-view object segmentation (*e.g.*, [16]) also aims at separating the object of interest from the background in all of the given views. However, it requires buffering frames from multiple views for performing segmentation for every incoming live frame. In contrast, our proposed 2D-3D co-segmentation only requires a sparse 3D point cloud and the segmentation only needs to be performed once.

Multimodal approaches to inferring the appropriate anchoring of annotations is also an interesting avenue for further exploration. For example, by using natural language processing to detect the words "pull" or "push" while the user is drawing an arrow may help the system to decide how to anchor the arrow in 3D space.

Finally, we emphasize again that our method is not only applicable to AR-based remote collaboration but can also be used in a wide range of mixed-reality scenarios, including asynchronous collaboration.

7 Conclusion

We presented a novel method for interpreting and rendering 2D gesture annotations in 3D augmented reality. This approach was motivated by an experiment that used Amazon Mechanical Turk with 88 participants to better understand the types of 2D annotations users draw for remote collaboration scenarios and to see if there are any differences in types of drawing based on task and viewpoint. The results showed that participants tend to draw arrows, circles, and outlines more than other types of drawings, and that arrows are used more with action tasks and at oblique angles to the objects of interest.

Based on these findings, we developed a novel real-time "gesture enhanced" approach to anchoring 2D arrow and circle annotations in 3D space, describing the main problems and ambiguities a system needs to handle in each case. For arrows, we identified their anchor points and used scene surface normals for better perspective rendering. For circles, we designed a novel 2D-3D co-segmentation energy function to help infer the object of interest using both 2D

image cues and 3D geometric cues.

Our results demonstrate that our method can better convey the user's original intention when rendering 2D gesture annotations from different viewpoints compared to previous methods. Specifically, participants from Amazon Mechanical Turk rated 243 out of 320 (76%) gesture enhanced arrows as better conveying the user's intentions both in action and referencing tasks compared to the median depth plane interpretation [5]. In addition, our novel 2D-3D co-segmentation circle annotation transfer method was able to increase the intersection-over-union score by an average of 167% compared to the median depth plane interpretation [5].

8 Acknowledgments

We thank Markus Tatzgern, Denis Kalkofen, Steffen Gauglitz, and the anonymous reviewers for their valuable feedback. We also thank Yuqi Chen and Anna Luo for help with the statistical analyses. This work was supported by NSF grant IIS-1219261 and ONR grant N00014-14-1-0133.

References

[1] D. Bourguignon, M.-P. Cani, and G. Drettakis. Drawing for Illustration and Annotation in 3D. In *Computer Graphics Forum*, volume 20, pages 114–123. Wiley Online Library, 2001.

[2] A. W. Fitzgibbon and R. B. Fisher. A Buyer's Guide to Conic Fitting. In *Proceedings of the 6th British Conference on Machine Vision (Vol. 2)*, BMVC '95, pages 513–522, Surrey, UK, 1995. BMVA Press.

[3] S. R. Fussell, L. D. Setlock, J. Yang, J. Ou, E. Mauer, and A. D. I. Kramer. Gestures Over Video Streams to Support Remote Collaboration on Physical Tasks. *Human-Computer Interaction*, 19(3):273–309, Sept. 2004.

[4] S. Gauglitz, C. Lee, M. Turk, and T. Höllerer. Integrating the physical environment into mobile remote collaboration. In *Proc. ACM MobileHCI*, pages 241–250, New York, NY, USA, 2012. ACM.

[5] S. Gauglitz, B. Nuernberger, M. Turk, and T. Höllerer. In Touch with the Remote World: Remote Collaboration with Augmented Reality Drawings and Virtual Navigation. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pages 197–205, New York, NY, USA, 2014. ACM.

[6] S. Gauglitz, B. Nuernberger, M. Turk, and T. Höllerer. World-stabilized Annotations and Virtual Scene Navigation for Remote Collaboration. In *Proceedings of the 27th ACM Symposium on User Interface Software and Technology*, pages 449–459, New York, NY, USA, 2014. ACM.

[7] P. Gurevich, J. Lanir, B. Cohen, and R. Stone. TeleAdvisor: A Versatile Augmented Reality Tool for Remote Assistance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 619–622, New York, NY, USA, 2012. ACM.

[8] J. D. Hincapié-Ramos, X. Guo, P. Moghadasian, and P. Irani. Consumed Endurance: A Metric to Quantify Arm Fatigue of Mid-air Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1063–1072, New York, NY, USA, 2014. ACM.

[9] D. D. Hoffman and W. A. Richards. Parts of recognition. In *Cognition*, pages 65–96. Elsevier, 1984.

[10] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental Surface Extraction from Sparse Structure-from-Motion Point Clouds. In *Proceedings of 24th British Machine Vision Conference*, BMVC '13, pages 94.1–94.11, Bristol, UK, 2013. BMVA Press.

[11] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[12] T. Jung, M. D. Gross, and E. Y.-L. Do. Annotating and Sketching on 3D Web Models. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pages 95–102, New York, NY, USA, 2002. ACM.

[13] S. Kasahara, V. Heun, A. S. Lee, and H. Ishii. Second Surface: Multi-user Spatial Collaboration System Based on Augmented Reality. In *SIGGRAPH Asia 2012 Emerging Technologies*, SA '12, pages 20:1–20:4, New York, NY, USA, 2012. ACM.

[14] S. Kim, G. A. Lee, and N. Sakata. Comparing Pointing and Drawing for Remote Collaboration. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–6, Oct. 2013.

[15] D. Kirk and D. S. Fraser. Comparing Remote Gesture Technologies for Supporting Collaborative Physical Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1191–1200, New York, NY, USA, 2006. ACM.

[16] A. Kowdle, Y.-J. Chang, D. Batra, and T. Chen. Scribble Based Interactive 3D Reconstruction Via Scene Co-Segmentation. In *Proceedings of the 18th IEEE International Conference on Image Processing (ICIP)*, pages 2577–2580, 2011.

[17] P. Mohr, B. Kerbl, M. Donoser, D. Schmalstieg, and D. Kalkofen. Retargeting Technical Documentation to Augmented Reality. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3337–3346, New York, NY, USA, 2015. ACM.

[18] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[19] J. Ou, S. R. Fussell, X. Chen, L. D. Setlock, and J. Yang. Gestural Communication over Video Stream: Supporting Multimodal Interaction for Remote Collaborative Physical Tasks. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 242–249, New York, NY, USA, 2003. ACM.

[20] P. Paczkowski, J. Dorsey, H. Rushmeier, and M. H. Kim. Paper3D: Bringing Casual 3D Modeling to a Multi-touch Interface. In *Proceedings of the 27th ACM Symposium on User Interface Software and Technology*, UIST '14, pages 23–32, New York, NY, USA, 2014. ACM.

[21] J. S. Pierce, A. S. Forsberg, M. J. Conway, S. Hong, R. C. Zeleznik, and M. R. Mine. Image plane interaction techniques in 3D immersive environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 39–43, New York, New York, USA, Apr. 1997. ACM Press.

[22] I. Poupyrev, N. Tomokazu, and S. Weghorst. Virtual Notepad: Handwriting in Immersive VR. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 126–132, 1998.

[23] G. Reitmayr, E. Eade, and T. W. Drummond. Semi-automatic Annotations in Unknown Environments. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 67–70, Nov 2007.

[24] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, Aug. 2004.

[25] R. S. Sodhi, B. R. Jones, D. Forsyth, B. P. Bailey, and G. Maciocci. BeThere: 3D Mobile Collaboration with Spatial Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 179–188, New York, NY, USA, 2013. ACM.

[26] S. C. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius. Convexity based object partitioning for robot applications. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3213–3220, May 2014.

[27] M. Tait and M. Billinghurst. The Effect of View Independence in a Collaborative AR System. *Computer Supported Cooperative Work (CSCW)*, 24(6):563–589, 2015.

[28] M. Tsang, G. W. Fitzmaurice, G. Kurtenbach, A. Khan, and B. Buxton. Boom Chameleon: Simultaneous Capture of 3D Viewpoint, Voice and Gesture Annotations on a Spatially-aware Display. In *Proceedings of the 15th ACM Symposium on User Interface Software and Technology*, UIST '02, pages 111–120, New York, NY, USA, 2002. ACM.

[29] J. Wither, S. DiVerdi, and T. Höllerer. Annotation in outdoor augmented reality. *Computers & Graphics*, 33(6):679–689, 2009.

[30] J. O. Wobbrock, A. D. Wilson, and Y. Li. 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*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.