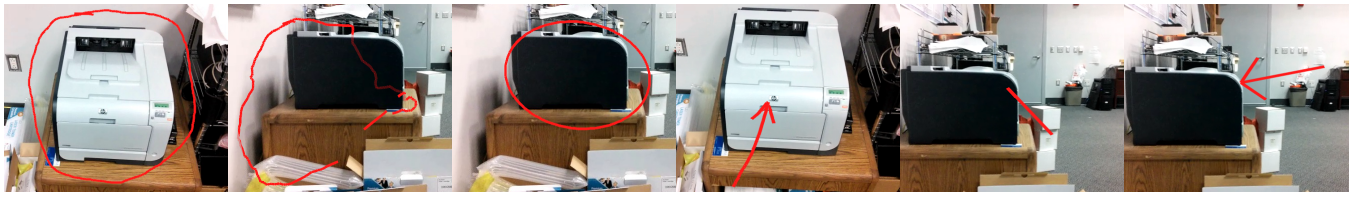
Anchoring 2D Gesture Annotations in Augmented Reality

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

University of California, Santa Barbara



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

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

ABSTRACT

Augmented reality enhanced collaboration systems often allow users to draw 2D gesture annotations onto video feeds to help collaborators to complete physical tasks. This works well for static cameras, but for movable cameras, perspective effects cause problems when trying to render 2D annotations from a new viewpoint in 3D. In this paper, we present a new approach towards solving this problem by using gesture enhanced annotations. By first classifying which type of gesture the user drew, we show that it is possible to render 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 remote collaboration by running an Amazon Mechanical Turk study with 88 participants. Next, we designed a novel system to automatically handle the top two 2D gesture annotations—arrows and circles. Arrows are handled by identifying their anchor points and using surface normals for better perspective rendering. For circles, 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 approach outperforms previous methods in terms of better conveying the original drawing's meaning 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

1 INTRODUCTION

Initially, research using 2D drawing annotations for remote collaboration employed statically positioned cameras and used screen-stabilized annotations [1]. 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.

*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 Virtual Reality Conference 2016
19–23 March, Greenville, SC, USA
978-1-5090-0836-0/16/$31.00 ©2016 IEEE

To overcome this limitation, newer research has employed computer vision tracking in augmented reality to avoid statically positioned cameras, thus enabling 2D drawing annotations to be world-stabilized [2]. The main challenge then becomes how to render such 2D annotations in 3D space such that they still convey the intended information when seen from different viewpoints.

Previous approaches have largely focused on either a graffiti/spray-paint approach (Figure 1b) [2, 3]; or a planar approach, such as using a statistic of the points (e.g., median depth, Figure 1e) [2]. However, both of these approaches suffer from perspective effects and therefore do not optimally convey the intended information.

In this paper, we take a fresh approach toward solving this problem. We argue that not all 2D drawing annotations should be handled the same way, and therefore we use a gesture classifier [7] to first determine what the user has drawn. Based on this classification, the system then takes the appropriate steps to enable a meaningful rendering of such 2D annotations in augmented reality.

2 AMAZON MECHANICAL TURK STUDY

In order to know what types of 2D gestures our system should handle, we conducted a user study with Amazon Mechanical Turk (AMT) with 88 participants. 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; we had 4 referencing and 3 action tasks, worded as simple questions such as, "Where is the printer door?"

There was a total of 1,847 drawings after removing outliers (sometimes participants stated images did not load, or they did not draw on the image, etc.). Across all tasks, participants drew arrows 53.38% of the time, circles 41.20% of the time, and all other gesture types less than 13% of the time. Based on these results, we chose to handle the top two gesture annotations—arrows and circles.

3 ANCHORING ARROWS IN 3D

We make the simple assumption that most times users will want the arrow to be anchored at what its head is pointing to. If the user wants the arrow to be anchored at where its tail is, such as when indicating a "pull" gesture, we require the user to draw another annotation (*e.g.*, a circle) near the desired anchor point. To determine the location of the arrow's head, we search for large changes in the direction of the 2D drawing. We also noticed that the vast majority of arrows in the AMT study were drawn directly onto the object

(89.6%). Based on this, we use the closest foreground object within a small search region near the arrow head.

Next, we utilize the surface normal at the 3D anchor point for the arrow head to determine the 3D direction **d** of the rendered tail, keeping it to be pointing closely to how the user originally drew it whenever the live viewpoint is close to the original drawing's viewpoint. To handle cases where the angle θ between the rendering viewpoint's principal axis and the surface normal **n** is close to 180° (*i.e.*, parallel), we further adjust the rendered tail after projection to 2D by moving it vertically below the arrow head in screen-space whenever $\theta > 150°$.

4 Anchoring Circles in 3D

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 user 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 [4]); 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:

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

Here, the optimization goal is to label **P** and **Q** to be foreground or background, where **P** are the 2D points in the user-annotated frame I and **Q** are the 3D points in the model. T is a transformation that projects **Q** to the image plane, and T^{-1} projects **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 [6].

We adopt a piecewise optimization strategy to efficiently solve Equation (1), i.e., iteratively minimizing one term and then the other. Note that although we do not pre-train the color models, the convex hull obtained from fitting user's input drawing helps make a good initialization such that minimizing the first term can resort to expectation-maximization style solvers. GrabCut [5] 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. [6] directly, which only locally evaluates the convexity based on current labeling and one neighboring 3D point each time to determine if the foreground should expand to that particular point.

5 Implementation & Results

We followed the approach by Gauglitz et al. [2], using a monocular SLAM system on a Nexus 7 tablet, streaming a video to a commodity desktop PC, which then builds a 3D model. To evaluate our novel arrow and circle annotation methods, we compared our methods against the median depth plane interpretation since this was what we considered the most competitive method introduced in previous work [2].

To evaluate our arrow anchoring method, we conducted another AMT study where we showed each participant 16 images of a user-drawn arrow indicating a particular object in the scene. Based on this image, we showed an additional 2 images side by side, showing the gesture enhanced and median depth plane annotation transfer interpretations. We asked the participants, "Which image (left or right) best conveys the same meaning as the drawing in the first picture?" With 20 participants, the results were that 270 (84%) votes chose the gesture enhanced arrows to better convey the meaning of the original drawing, whereas only 50 (16%) chose the median depth interpretation.

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

Table 1: IoU score (in %) for the proposed gesture enhanced circle and the median depth circle methods.

To evaluate our 2D-3D co-segmentation method for anchoring circle annotations, we manually marked the ground-truth objects in 5 drastically different viewpoints in each of three 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 1 shows a quantitative comparison between the 2D-3D co-segmentation method and the median depth method.

6 Conclusion

We developed a novel "gesture enhanced" approach to anchor 2D arrow and circle annotations in 3D space for remote collaboration. For arrows, we identified their anchor points and used the 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 system can better convey the user's original intention when rendering 2D gesture annotations from different viewpoints compared to previous methods. Specifically, participants in a small study on Amazon Mechanical Turk rated 270 (84%; out of 320) gesture enhanced arrows over the median depth plane interpretation [2]. 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 [2].

7 Acknowledgments

We thank Markus Tatzgern, Denis Kalkofen, Steffen Gauglitz, and the anonymous reviewers for their valuable feedback. This work was supported by NSF grant IIS-1219261 and ONR grant N00014-14-1-0133.

References

[1] 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. *HCI*, 19(3):273–309, Sept. 2004.

[2] 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 ACM VRST*, pages 197–205, New York, NY, USA, 2014. ACM.

[3] P. Gurevich, J. Lanir, B. Cohen, and R. Stone. TeleAdvisor: A Versatile Augmented Reality Tool for Remote Assistance. In *Proceedings of CHI*, pages 619–622, New York, NY, USA, 2012. ACM.

[4] K.-C. Lien, B. Nuernberger, M. Turk, and T. Höllerer. 2D-3D Co-segmentation for AR-based Remote Collaboration. In *ISMAR*, 2015.

[5] 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.

[6] S. C. Stein, F. Worgotter, J. P. Markus Schoeler, and T. Kulvicius. Convexity based object partitioning for robot applications. In *Proceedings of IEEE ICRA*, 2014.

[7] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. of CHI*, pages 159–168, New York, NY, USA, 2007. ACM.