Hand Inspection of Augmented Reality Objects

Kyle Balousek, Yanqi Liu, Max Smith
University of Michigan, College of Engineering
{kbalouse, liuyanqi, mxsmith}@umich.edu

***Abstract*—Interacting with augmented reality (AR) objects robustly is an unsolved problem. Current methods focus on using tags as a an interaction tool; however, restricting systems to always have tags is a major constraint. This work seeks to remove this constraint by using a hand to directly interact with these objects. The method involves segmenting the hand, detecting and tracking fingertips, and estimating the camera pose. Placing AR objects into the hands frame allows us to no longer require tags, and users may directly interface with the objects.**

***Keywords*—Augmented, Reality, Inspection**

I. Introduction

Suppose a medical student was learning about the inner mechanisms of a heart. It would prove much more informative if they were able to tangibly have a heart to inspect. This is a very unrealistic circumstance - hearts are a scarce resource, so students are shown diagrams of projections of the heart. This loses lots of potentially valuable information. Augmented reality (AR) poses a solution to this problem by allowing three-dimensional information to be stored and represented in the world without losing information through digital projection. Allowing medical students to directly interact with human anatomy at no cost has the potential to greatly improve their understanding by conveying the most detailed information possible. Solving this problem does not end at students, full dimensional information sharing has countless other applications from showing house designs to remote maintenance assistance.

Canonical AR does not natively come with a human-computer interaction paradigm to interact with the augmented objects. This work seeks to span this gap and create a system for inspection of augmented reality objects in a users hand. The proposed methodology segments the hand's skin from the background, extracts the hand contour, then detects fingertips based on curvature of the edges of the contour. The fingertips' coordinates are then used to calibrate the cameras extrinsic parameters allowing mapping of images onto the hands reference frame. Figure 1, shows the final result where an AR object is inspected through hand movement.

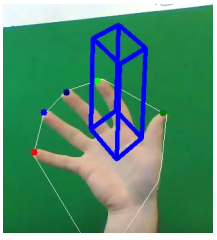

Fig. 1: AR object inspection through hand manipulation.

II. Related Work

Marker tracking is the most frequently used method in systems for AR object inspection [1]. This involves putting unique identifiers (tags) on points of interest; namely, in this problem fingers would be given identifiers (e.g., color coded tape) for trivial information extraction. This method puts a major constraint on the system, namely the markers, which does not translate into the applied world where these conditions might not be possible to impose.

The novelty this work contributes is the integration of several state-of-the-art computer vision techniques to remove the tag constraint. Instead of using tags, our method uses the users hand as the interaction medium. This allows anyone to leverage this technology without any additional resources. Removing the need for tags allows the application of this technology in a many more scenarios where marking objects may not be possible.

III. Methodology

Our solution involves a five-step process: hand contour detection, fingertip recognition, fingertip tracking, pose estimation, and projection. We begin by detecting the hand contour from the original image, followed by fingertip detection on the contour. We additionally use fingertip tracking to achieve more stable fingertip positions. By finding the corresponding positions of fingertips in world and image coordinates, we are able to establish enough correspondences to solve for the extrinsic parameters of the camera (intrinsic parameters are calculated from the canonical checkerboard [2], [3]). This allows us to devise a mapping from the cameras frame of reference onto the center of the hands coordinates. The AR object is then projected into the hand space and rendered on the image.

A. Hand Contour Detection

Hand contour detection is done in three parts: skin detection, edge detection and contour extraction. The process for segmenting skin from the background is more involved than one would might think. We begin by converting the original image in the RGB color space into an image in the YCrCb color space and threshold the colored image to produce a binary image which differentiates skin and background pixels. Previous research has found that certain ranges of chrominance, namely $C_r = [133, 173]$ and $C_b = [77, 127]$, are good indications of a pixel being skin [4]. These thresholds were used in our implementation and performed well.

After applying the initial segmentation using the previously mentioned thresholds, there was still some background noise in the image. We tried a variety of different methods to reduce

this noise, and in the end devised a more reliable system. Prior to segmenting the image, we apply a Gaussian blur on the image. We then segment, erode, dilate, and lastly apply a median blur over the image. This process caused noise to be reduced, allowing our recognition system to focus just on the hand and giving us a smooth hand border. While this process worked for our purposes, it may have different results depending on the type of camera used. In the future, it would be useful to determine a method with universal robust performance.

While this model might be relatively simple and at times have poor segmentation results, it was chosen for it's very quick performance. The desired toolkit needs to have fast performance to allow for real-time feedback for the user. Further work may build-upon this particular component through different color spaces, more complex models, probabilistic models, and machine learning.

With the segmented image we can produce an image of the outline of the hand using the canny [5] edge detector algorithm. The pixels in the hand contour are then extracted for later use in fingertip recognition using OpenCVs [3] implementation of the border following algorithm [6] (a contour extraction algorithm). These methods were chosen as they are the state-of-the-art solutions, and provided very reliable results; the result of these steps are shown in Figure 2.

Fig. 2: Left picture shows the original hand image; the middle picture shows the binary image after skin color segmentation; the right image shows the hand contour

B. Fingertip Recognition

From the vector of edge points that were obtained from the canny edge detection we can identify candidates for fingertips by checking the curvature at each edge point. Areas of high curvature are flagged as potentials for fingertips. This is done by measuring the angles as described in [2] via (here P_i is the ith a point along the hand edge and l is the displacement index):

$$Curvature(P_i, l) = \frac{\overrightarrow{P_iP_{i-l}}\overrightarrow{P_iP_{i+l}}}{||\overrightarrow{P_iP_{i-l}}|| \cdot ||\overrightarrow{P_iP_{i+l}}||} \quad (1)$$

Candidates were identified if they had a angle of less than 30° using a displacement index of 15 (chosen by cross-validation). We then performed clustering on the candidates and identified five main clusters. We distinguished between valleys of hand and fingertips by choosing the cluster that has the closest distance to the convex hull of our hand. The centroids of these clusters are our fingertips, and the cluster that had the furthest respective L2 norm was marked as the thumb. The remaining fingertips are identified by distance from the thumb.

C. Fingertip Tracking

We implement fingertip tracking to prevent erroneous fingertip correspondences. Since fingertips are sorted in order, starting from the thumb to tiny finger, we calculate the displacement for corresponding fingertips in two consecutive frames. If the displacement is larger than a threshold, we know that there is a mismatch of fingertips or not all five fingertips are detected. Then we keep the previous fingertips' positions to indicate the current position. After implementing tracking, we found that the detected fingertips were stable and did not jump to random locations in the image if a frame was poorly segmented.

D. Pose Estimation and Projection

The hand and fingertip positions are measured with the middle of the palm as the origin in order to calibrate the fingertips' world coordinates. Using the two-dimensional measurements, an assumption is made that the palm is 5mm above the checkerboard - approximation of hand depth. Given the world and image coordinate of the fingertips and the camera intrinsic and distortion parameteres, we use the OpenCV solvePnP functions, which uses an iterative method based on Levenberg-Marquardt optimization to minimize the reprojection error, to estimate the cameras extrinsic parameters [2], [3]. We begin the process by assuming the hand is pointing upwards, with the thumb to the right. This allows us to specify some default behaviours; namely, the x-dimension in the hand space points towards the thumb, the y to the fingers, and the z upwards from the palm. The hands coordinate system may be seen in Figure 3.

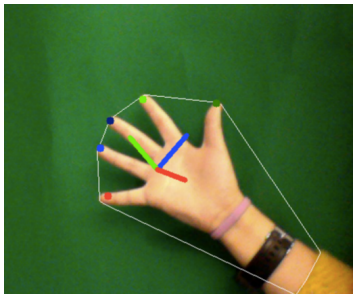

Fig. 3: Coordinate system projected onto hand's coordinate system.

After finding all the intrinsic and extrinsic parameters of the camera, we can construct camera matrix P, and project any three-dimensional points onto the hand coordinates.

$$P = K \begin{bmatrix} R & T \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = P \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3)$$

IV. Experiments

As Figure 1 shows, our methodology proved successful in projecting AR objects on still images. However, this alone doesn't allow us to completely observe all dimensions of data. Across successive frames in a video, changing our hand's pose should allow for complete information recovery. Several different hand movements were tested to ensure any form of information inspections may work. In order to have full control of a three-dimensional object we need to be able to: dilate, translate, and rotate the object. Any combination of these transformations would allow us to observe any information at any position surrounding the object.

Through hand pose changes, we could demonstrate all the desired transformations. Table I shows the three experiments, and the corresponding inspection functionality it demonstrates.

TABLE I: Transformation experiment summary.

Experiment	Transformation
Sliding hand across screen	Translation
Rotating hand in place	Rotation
Bringing hand towards camera	Scaling

A. Translation

The hand, while keeping constant orientation, was translated across the video screen. This would provide us spatial invariance, as inspection could be maintained at any point on frame. Figure 4 shows two frames of the recording of this experiment, where the hand was moved from right to left across the screen.

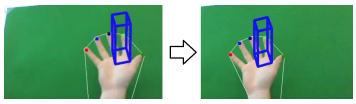

Fig. 4: Translation of hand from the right to left.

The results of this experiment prove that our system successfully enables translation. It was observed that at a few instances enough noise was found in the image to force our system to use the position in previously successful frames. This prevented a completely fluid experience, and could be rectified by further research in more robust fingertip detection and background noise removal.

B. Rotation

The AR object should rotate synchronously with the rotation of the hand. This will allow information that was originally obstructed to be viewable. This is demonstrated by rotating the hand in a counter-clockwise fashion until an previously un-seen edge is visible, shown in Figure 5. The results were successful, with the same lag concerns, as mentioned for Translation.

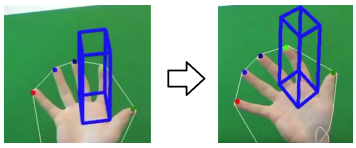

Fig. 5: Rotation of hand clockwise, causing rotation of AR object.

C. Scaling

While not truly scaling, in-order to perceive images larger they're brought closer to the sensor (e.g., eye, camera). The system follows this same intuition, you can't directly stretch or contort an object. Instead, a user would bring their hand closer, causing the projection to feel larger. All while maintaining the same actual size. This is visible in Figure 6, and the same issues were noticed from the previous transformations.

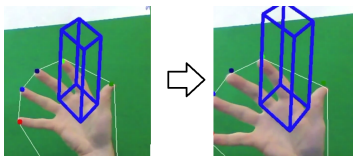

Fig. 6: Scaling of object by bringing hand towards camera.

V. Conclusions and Future Work

Inspection of AR objects through hand manipulation, proves to be a much more robust and less constrained problem then through markered inspection. This was shown in our three experiments where all methods of inspection were proven to work and provide additional previously unobtainable information. Currently our solution makes some simplifying assumptions that would be the points of further improvement. The background is assumed to be nearly uniform, and not of the same color as the hand. Further research into a computationally efficient more robust skin recognition system would allow for this system's constraint to be removed. This improvement will also remedy the slight lag experienced in our tests, since the fingertips would be detected more frequently and to a higher degree of accuracy. With these improvements in place, our system wouldn't need to skip frames due to an incorrect number of detected fingertips or incorrect hand detection.

References

[1] A. Pagani J. Kohler and D. Stricker. Detection and identification techniques for markers used in computer vision.

[2] T. Lee and T. Hollerer. Hand ar: Markerless inspection of augmented reality objects using fingertip tracking. *IEEE ISWC*, 2007.

[3] Intel Corporation. Open source computer vision library reference manual, 2000.

[4] D. Chai and K.N. Ngan. Face segmentation using skin-color map in videophone applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 1999.

[5] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1986.

[6] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *CVGIP*, 1985.