Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking

Taehee Lee, Tobias Höllerer
Four Eyes Laboratory, Department of Computer Science
University of California, Santa Barbara, California 93106 USA

Abstract

We present markerless camera tracking and user interface methodology for readily inspecting augmented reality (AR) objects in wearable computing applications. Instead of a marker, we use the human hand as a distinctive pattern that almost all wearable computer users have readily available. We present a robust real-time algorithm that recognizes fingertips to reconstruct the six-degree-of-freedom camera pose relative to the user's outstretched hand. A hand pose model is constructed in a one-time calibration step by measuring the fingertip positions in presence of ground-truth scale information. Through frame-by-frame reconstruction of the camera pose relative to the hand, we can stabilize 3D graphics annotations on top of the hand, allowing the user to inspect such virtual objects conveniently from different viewing angles in AR. We evaluate our approach with regard to speed and accuracy, and compare it to state-of-the-art marker-based AR systems. We demonstrate the robustness and usefulness of our approach in an example AR application for selecting and inspecting world-stabilized virtual objects.

1. Introduction

Augmented reality (AR) is a powerful human-computer interaction paradigm for wearable computing applications. The world around a mobile computer user can directly serve as the user interface, presenting a location-specific 3D interaction space where the user can display, examine, and manipulate information [4]. A successful standard approach for viewing AR content and registering it with the world is via vision-based tracking of cardboard fiducial markers [16][8], which can be used as a hand-held tangible user interface for inspecting and manipulating the augmentations [25].

Mobile AR research [12] has produced many useful user interface options for wearable computing [7][27][24][32]. For direct manipulation of AR content, these applications have so far relied on special interaction device technologies, such as pinch gloves with fiducial markers [31] or head-to-hand tracking equipment such as the WearTrack solution [9].

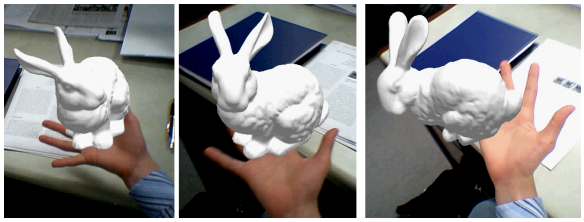

Figure 1. Inspecting a virtual bunny on top of the user's hand from different viewing angles.

In this paper, we present and evaluate a method to use a user's bare outstretched hand in the same way a cardboard AR marker would be used, enabling spontaneous tangible user interfaces in mobile settings. The human hand is a thoroughly ubiquitous input device for all kinds of real-world applications. Our work broadens the applicability of tangible AR interfaces by eliminating the need for hand-held markers. With our method, wearable computing users can conveniently inspect and manipulate AR objects relative to their own body, well within the user's comfort zone [17]. The trend in AR libraries points towards increased use of markerless tracking. Our technique allows a wearable user to retain the undisputed tangible UI benefits of cardboard markers without the inconvenience of having to carry one around at all times.

1.1. Related Work

Hand-gesture interfaces have been widely investigated from the perspectives of computer vision, augmented reality and human-computer interaction. Many early hand-based UIs rely on tracking the hand in a 2D viewing plane and use this information as a mouse replacement [18][19]. Some systems demonstrated fingertip tracking for interactions in a desktop environment [21][23], while interactions with wearable computing systems were introduced using hand movements and gestures [22]. Tracking a hand in 3D space was implemented using a stereoscopic camera [2][10]. We focus on standard wearable (miniature) cameras in our work. While there is very promising real-time work in recognizing dynamic gestures from approximated hand shapes over time using Hid-

1-4244-1453-9/07/$25.00

den Markov Models (e.g., [28]), none of the many proposed approaches to reconstruct an articulated 3D hand pose in detail (e.g., [29][30]) is currently feasible at 30 frames per second. We focus on real-time real life AR interfaces that should still leave a wearable computer sufficient computing power for the execution of application logic. A variety of fingertip detection algorithms have been proposed [21][34][2], each one with different benefits and limitations, especially regarding wearable computing environments with widely varying backgrounds and variable fingertip sizes and shapes. Careful evaluation of this work led us to implement a novel hybrid algorithm for robust real-time tracking of a specific hand pose.

Wearable computers are important enabling technology for "Anywhere Augmentation" applications [13], in which the entry cost to experiencing AR is drastically reduced by getting around the need for instrumenting the environment or creating complex environment models off-line. Hand interfaces are another important piece of the Anywhere Augmentation puzzle, as they help users establish a local coordinate systems within arm's length and enable the user to easily jump-start augmentations and inspect AR objects of interest.

An important problem in AR is how to determine the camera pose in order to render virtual objects in correct 3D perspective. When seeing the world through a head-worn [12] or magic-lens tablet display [26], the augmentations should register seamlessly with the real scene. When a user is inspecting a virtual object by "attaching it" to a reference pattern in the real world, we need to establish the camera pose relative to this pattern in order to render the object correctly. Camera calibration can be done with initialization patterns [35] for both intrinsic and extrinsic parameters. In order to compute extrinsic parameters of camera pose on-the-fly, metric information is required for the matching correspondences. In AR research, marker-based camera pose estimation approaches [16][8] have shown successful registration of virtual objects with the help of robust detection of fiducial markers. We replace such markers with the user's outstretched hand.

The rest of this paper is structured as follows: In Section 2, fingertip tracking and camera pose estimation are described in detail. In Section 3, we show experimental results regarding the speed and robustness of the system and present examples of AR applications employing the hand user interface. In Section 4, we discuss benefits and limitations of our implemented method. We present our conclusions and ideas for future work in Section 5.

2. Method Description

Wearable computing users cannot be expected to carry fiducial markers with them at all times. We developed a vision-based user interface that can track the user's outstretched hand robustly and use it as the reference pattern for AR inspection. To this end, we present a method of tracking the fingertip configuration of a single hand posture, and use it for camera pose estimation. In Figure 2, the overall data flow for the system is illustrated. In a one-time calibration step, we measure a user's hand, storing the positions of the outstretched fingertips relative to each other. This needs to be done only once in a user's lifetime. For online real-time tracking, we segment the hand region in the captured camera image and then detect and track fingertips. We recognize and track an outstretched hand in arbitrary position and rotation as seen by a wearable camera, and we derive a six-degree-of-freedom (6DOF) estimate for the camera, relative to the hand.

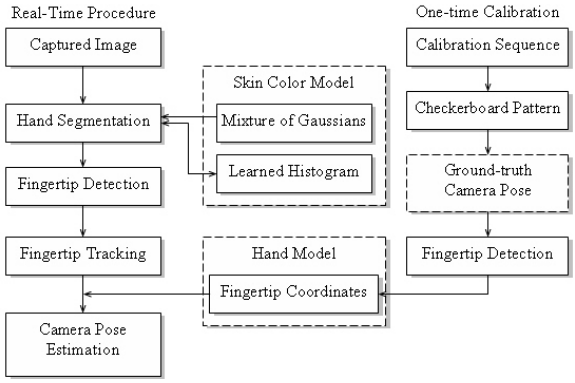

Figure 2. Flowchart of one-time calibration and real-time camera pose estimation using fingertip tracking.

2.1. Adaptive Hand Segmentation

Given a captured frame, every pixel is categorized to be either a skin-color pixel or a non-skin-color pixel. An adaptive skin color-based method [18] is used to segment the hand region. According to the generalized statistical skin color model [15], each pixel is determined to be in the hand region if the skin color likelihood is larger than a constant threshold. In order to adapt the skin color model to the illumination change, a color histogram of the hand region is learned for each frame and accumulated with the ones from the previous n frames ($n = 5$ works well in practice). Then the probability of skin color is computed by combining the general skin color model and the adaptively learned histogram.

The histogram is learned only when the hand is in view and its fingertips are detected. For example, when the hand is moved out of sight, the histogram keeps the previously learned skin color model, so that the system can segment correctly when the hand comes back into the scene.

The segmentation result, as shown in Figure 3, is used for tracking the main hand region. Since we are talking about wearable computing scenarios, where a body-mounted camera sees the hand, which is by necessity within arm's reach, we can assume that the majority portion of the skin color segmented image is the hand region. In order to find the largest blob, we retrieve the point exhibiting the maximum distance

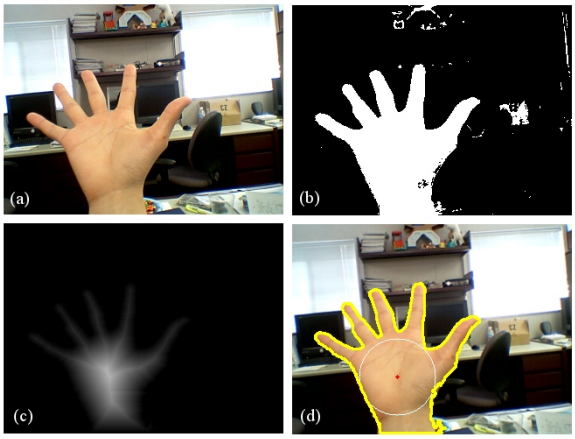

Figure 3. Hand segmentation procedure. Given a captured image (a), skin color segmentation is performed (b). Then the distance transform (c) is used to extract a single connected component of the hand (d).

value from the Distance Transform [5] of the segmentation image. Among skin-colored regions as shown in Figure 3b, a single connected component of the hand contour is extracted using OpenCV's implementation [14] by checking which region the centroid from the previous frame lies in. Since some large skin-colored object may come into the scene while we are tracking the hand, constraining the previous centroid of the hand to be in the current frame's hand region prevents the tracking system from jumping to a different region outside of the hand. This makes the assumption that a user's hand motion from frame to frame is not big enough for the centroid to leave the hand blob entirely. This is true for even very rapid hand motions at 30 frames per seconds. The contour of the tracked hand region, as shown in Figure 3d, is then used for next fingertip detection step.

2.2. Accurate Fingertip Detection

Fingertips are detected from the contour of a hand using a curvature-based algorithm similar to the one described in [2]. We then fit the curvature points to ellipses in order to increase the accuracy. The curvature of a contour point is measured on multiple scale levels in order to detect fingertips with various sizes as follows: The points with higher curvature values than a threshold (on any scale level) are selected as candidates for fingertips by computing a dot product of $\overrightarrow{P_iP_{i-l}}$ and $\overrightarrow{P_iP_{i+l}}$ as in

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

where P_i is the ith point in the contour, and P_{i-l} and P_{i+l} are preceding and successing points, with displacement index l on the contour representing the scale. In practice, a range for l that includes every integer between 5 and 25 works well. In addition to the high curvature value, the direction of the curve is considered to determine that the point is a fingertip and not a valley between fingertips. Directions are indicated by the cross product of the two vectors.

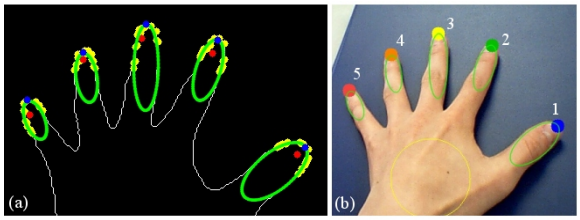

Figure 4. Fingertip detection procedure: Ellipses are fitted to contour, based on candidate curvature points (a). Fingertips are ordered based on detected thumb (b)

From the candidate fingertip points as shown in Figure 4a, an accurate fingertip point is computed by fitting an ellipse to the contour around the fingertip using least-squares fitting as provided by OpenCV [14]. We then compute the intersection points of the ellipse's major axis with its edge and pick the one closer to the fingertip estimated from curvature (cf. Figure 4). Experimental results show that this ellipse fitting method increases the camera pose estimation accuracy (see Table 2 and discussion in Section 4) compared to the point of largest curvature.

Since the detection algorithm may produce false positives of fingertips for initial detection, we choose the most frequently detected points above the center of the hand for a certain number of consecutive frames as our final fingertips. Thus, for *initial* detection, the hand has to be held *fingers up*, which is the most convenient and by far the most common pose anyway. After fingertips have been detected, we eliminate false positives by tracking a successful configuration over time. In our experiments, 10 frames are used for this initial fingertip detection period, which is far less than a second for a real-time application. The fingertips are then ordered based on the index of the thumb, which can be determined as the farthest fingertip from the mean position of all fingertips as shown in Figure 4b. The order of fingertips is later used for tracking the fingertips and estimating the camera pose.

2.3. Fingertip Tracking

Once fingertips are detected, we track them based on matching the newly detected fingertips to the previously tracked fingertips. Similar to [23], we track the fingertip trajectory by a matching algorithm that minimizes the displacement of pairs of fingertips over two frames. In addition, we use our knowledge about the centroid of the hand blob to effectively handle large movements of the hand as follows: The

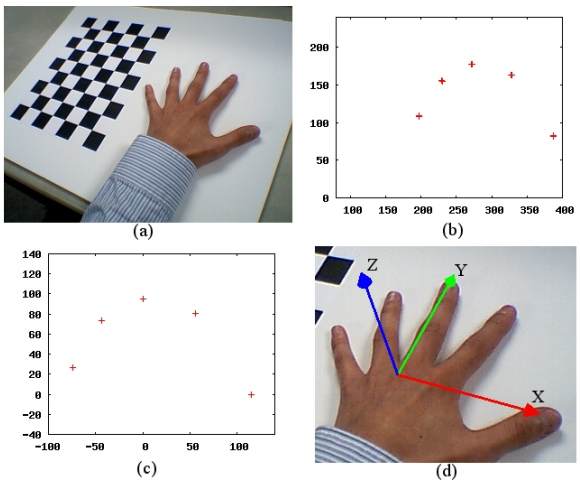

Figure 5. A hand model is constructed by putting the hand next to the checkerboard pattern (a), computing the fingertip positions (b). Then the coordinates are translated (c), yielding the hand coordinate system (d).

matching cost is minimized as

$$f_{i+1} = \arg\min \sum_{j=0}^{N-1} \|(f_{i,j} - C_i) - (f_{i+1,j} - C_{i+1})\| \quad (2)$$

where f_i and f_{i+1} are the sets of N fingertips at the ith and $i+1$th frames respectively, $f_{i,j}$ represents the fingertip of the jth index in f_i, and C_i and C_{i+1} are the centroids of the corresponding frames.

While matching the fingertips in two frames, the order of fingertips on the contour is used to constrain the possible cases of combinations, determining the ordering (front or back) by the position of the thumb.

2.4. One-Time Hand Model Construction

In order to estimate the camera pose from the tracked fingertips, we measure the size of the hand by calculating the position of the fingertips in a one-time initialization process. This measurement can be performed together with calibrating the intrinsic parameters of a camera [35], putting a hand with a wide spread posture next to an initialization pattern, as shown in Figure 5a. While the camera is moving around, we keep both the hand and the checkerboard pattern in view. Given the camera pose estimated from the checkerboard pattern, we unproject the fingertips in the captured image to the parallel plane at finger height:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = P_{3\times4} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (3)$$

where $(x\ y\ 1)^T$ and $(X\ Y\ Z\ 1)^T$ are homogeneous representations of the fingertip coordinates in the image plane and the world coordinate system respectively, and $P_{3\times4}$ is the projection matrix of the camera. By setting the Z coordinate as the average thickness of a finger, we can derive an equation to compute the X and Y positions of the fingertips given the (x, y) image points that are tracked as described in the previous section. In Figure 5b, the locations of fingertips are plotted with the XY plane, with Z coordinates assumed as 5mm above the initial pattern's plane.

From the measured fingertip positions relative to the origin corner point of the initialization pattern, we translate the coordinate system to the center of the hand. As shown in Figure 5c, the center point is defined with the X coordinate of the middle finger and the Y coordinate of the thumb. This results in the hand's coordinate system's y axis being aligned with the middle finger and the x axis going through the thumb tip as in Figure 5d. This coordinate system, centered to the hand, is then used for rendering virtual objects on top of the hand, providing direct control of the view onto the target.

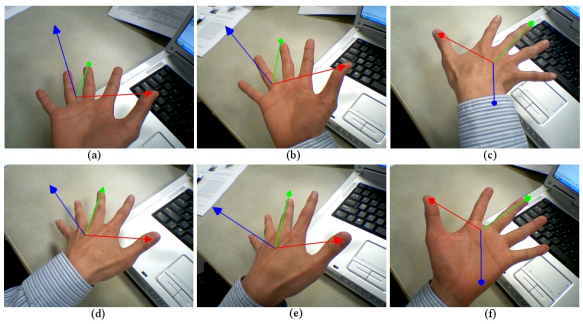

Figure 6. Estimating the camera pose from different viewing angles. The similarity of the right hand (a), (b), (c) and the left hand (d), (e), (f) enables the user to use both hands for object inspection.

2.5. Pose Estimation from Fingertips

Using the hand model and the extrinsic camera parameter estimation from [35], a 6DOF camera pose can be estimated. As long as the five fingertips of the fixed hand posture are tracked successfully, the pose estimation method has enough point correspondences, as four correspondences are the minimum for the algorithm. In order to inspect an AR object on top of the hand from different viewing angles, the user may rotate or move the hand arbitrarily as illustrated in Figure 6. Based on the symmetry of one's left and right hands, the hand model that is built from the left hand can be used for the right hand as well. In practice, one would likely measure the non-dominant hand palm-down next to the checkerboard pattern and use the dominant hand palm-up for inspecting objects.

Given that there are errors in tracking fingertips, it is advantageous to smooth the estimated camera pose using a Kalman filter [33] modeling the position and orientation of the camera [3]. In our implementation, we define the state x

Table 1. Processing time for different resolutions

Resolution Processing Time	320×240 (msec)	640×480 (msec)
Hand Segmentation	13.24	39.43
Fingertip Detection	4.13	14.80
Fingertip Tracking	0.01	0.01
Pose Estimation	1.54	0.87
Kalman Filtering	0.05	0.04
Total	22.27	69.83

of the Kalman filter as in [23]:

$$x = \begin{pmatrix} t \\ r \\ v_t \\ v_r \end{pmatrix} \tag{4}$$

where t is a 3-vector for translation of the camera, r is a quaternion for rotation, and v_t and v_r are velocities for them. The measurement y is directly modeled as

$$y = \begin{pmatrix} t \\ r \end{pmatrix} \tag{5}$$

where t and r are the same as in (4). Then the state transition, observation, and driving matrices are defined accordingly so that the estimated camera pose is smoothed and predicted to be robust against abrupt errors.

3. Results

We experimented with various configurations considering the robustness and the speed of the system. The results show that our method is applicable for real-time applications with sufficient accuracy to be a user interface alternative to fiducial marker tracking.

The experiments were performed on a small laptop computer with a 1.8GHz CPU, using a USB 2.0 camera with 640×480 resolution. These specs are in line with currently available ultra mobile PC and high-end wearable computing platforms. Intrinsic camera parameters were measured by the calibration method from [35] using the implementation of OpenCV [14] with a 6×8 checkerboard pattern. We used ARTag [8] to compare marker-based camera pose estimation with our fingertip tracking approach.

3.1. Speed and Accuracy

The goal for our system was to achieve real-time performance of at least 15 frames per second (a conservative minimum for interactive systems), and to strive for 30fps, which is considered real-time. Table 1 lists the speed of the fingertip tracking and the camera pose estimation procedures. Our first experiment shows that the system runs at around 15fps for 640×480 resolution, which meets the interactive system constraint. In order to increase the speed, we have tested it with 320×240 resolution for the hand segmentation and fingertip detection steps, while keeping the capturing and display resolution at 640×480. As a result, the system satisfies the real-time constraint, running over 30fps.

Table 2. Reprojection errors for different resolutions and fingertip detection methods.

Fingertip Detection	Resolution	RMS error (pixel)
Curvature only	320×240	8.97
	640×480	7.96
Ellipse fitting	320×240	5.86
	640×480	5.76

The increase in performance comes at a slight cost of tracking accuracy. In Table 2, we list the accuracy according to the two choices of resolutions. The accuracy is measured by computing the mean reprojection error at the 48 internal corner points of the initializing checkerboard pattern. The checkerboard is detected by OpenCV's implementation [14], which is considered to be the ground truth for our experiment (cf. Table 3). After building the hand model and determining the coordinates of the fingertips relative to the checkerboard, we reproject the checkerboard's corners from the estimated camera pose and then compute the RMS error of the reprojected points (i.e. per-corner pixel distances on the 640×480 viewing plane). The result, as shown in Table 2, shows that the accuracy at 320×240 is only marginally smaller than for 640×480.

We also assessed the accuracy improvement that we are getting by employing ellipsoid fitting to our fingertip detection, as described in section 2.2. As shown in Table 2, the ellipse fitting method helps to reduce the error of camera pose estimation considerably.

3.2. Comparison with Markers

In this experiment, we compared camera pose estimation accuracy based on markers and fingertips. As shown in Figure 7a, the user's hand and an ARTag marker of similar size are placed next to the ground-truth checkerboard pattern. The camera pose is then computed separately based solely on the marker, the hand, and the checkerboard pattern, respectively. We compare the reprojection errors at the checkerboard pattern's corners. As part of the experiment, we move the camera around while keeping the hand, the marker, and the checkerboard pattern all in the same view in order to fairly compare the estimation accuracy.

Figure 8 shows the results plotted over time. The y axis represents the reprojection error of the marker and the fingertip pose estimations, while the x axis shows the elapsed camera frames. The average and variance of the reprojection error is shown in Table 3, together with the ground-truth checkerboard's reprojection error. The peaks in the reprojection error from the fingertip pose estimation can be explained by abrupt inaccurate locations of fingertips. Since the five point correspondences from the fingertips are close

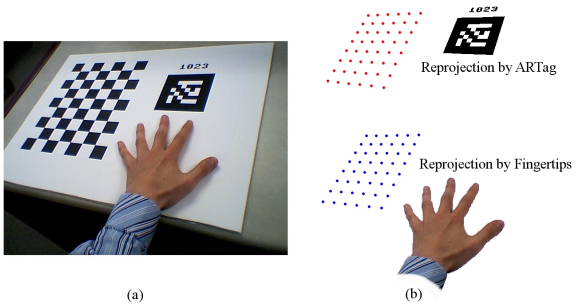

Figure 7. (a) A checkerboard, an ARTag marker, and a hand are seen in the same view. (b) The internal corners of the checkerboard are reprojected based on marker and hand tracking, respectively.

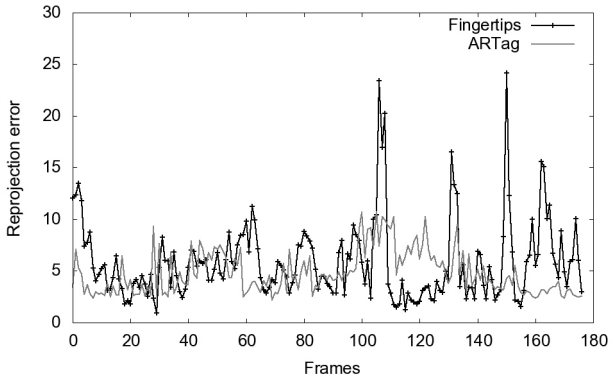

Figure 8. The reprojection errors of the camera pose estimation from fingertip and ARTag tracking

to the minimum number for the camera pose estimation algorithm, small deviations can cause error spikes in the results. In order to make the system more robust, occasional jitters are filtered out by applying our Kalman filter with fine-tuned noise covariance matrices and thresholds for the camera motion. In summary, this experiment demonstrates that the proposed fingertip tracking method can be used for camera pose estimation without losing significant accuracy compared to a state-of-the-art fiducial marker system.

3.3. Applications

We have implemented an augmented reality proof-of-concept application, in which a user can select a world-stabilized object and inspect it using their hand. In order to make an AR environment, we used ARTag markers [8] for stabilizing virtual objects in the environment as shown in Figure 9a: teapots with red, green, and blue colors, from left to right. The user can select a teapot by putting his or her hand close to the desired one as determined by pixel distance on the image plane. As the fingertips are detected and tracked successfully, the selected teapot is moved from its original position to the hand. While the world coordinate system is

Table 3. Average and variance of reprojection errors

Method	Average RMS error (pixel)	Variance
ARTag marker	4.79	4.40
Fingertips	5.86	14.77
Checkerboard	0.26	0.002

defined by the markers' transformation matrix, the local coordinate system of the camera pose estimated via fingertips is used for inspecting the virtual object on top of the hand.

In order to seamlessly transfer the object from the world coordinate to the local coordinate, a linear combination of the transform matrices based on the marker and the hand is calculated with a weight coefficient function over time. As shown in Figure 9, the virtual object is brought to the hand to allow the user to inspect it, and is released to its original position when the inspection ends.

4. Discussion

Our tests and experiments indicate that the hand segmentation is somewhat sensitive to changes in illumination, even though our adaptively learned color model helps robustness a great deal. In outdoor scenes, as in Figure 10, hand color can change quite drastically over short periods of time and even spatially within one frame due to more pronounced surface normal shading and shadowing. Also the hand color can get saturated to white, which does not distinguish well from other bright objects. Most of these effects can be diminished by using a high quality camera featuring rapid auto-gain control.

Because the hand region becomes an input to the fingertip detection, the first step of detecting and tracking the hand is very important. The assumption for our hand blob tracking, classifying the largest skin color area as a hand, works effectively for the area within a user's reach. However, it may fail when a larger region with skin color comes into the scene and overlaps with the hand. For example, our implementation will not successfully track a hand when the view shows two hands of similar area, overlapping with each other. Tracking multiple skin-colored objects can be performed by a more sophisticated algorithm [1].

Regarding accurate fingertip detection, we have tested other approaches than the proposed curvature-based detection and ellipse fitting algorithm. A template-based approach could apply shape filters [23][21][34] to detect a skin-colored area with fingertip model constraints, for example, having a circle attached to a cylinder. Such a method has the benefit of detecting the fingertip without the assumption of the hand being a single connected component. However, the computational cost is much higher than for the contour-based algorithm when detecting fingertips at multiple scales ranging from "very close" to "at arm's length" as is normal in wearable computer applications. Additionally, shape filters tend to produce more false negatives than the contour-based approach, which makes it hard to track the fingertip continuously with a moving camera. Another potential benefit of

Figure 9. Selecting and inspecting augmented objects. (a) ARTag is used for stabilizing teapots in the world. (b) Selecting the green teapot in the middle, and then (c) inspecting it from (d) several angles. (e) The user releases the object by breaking the fingertip tracking.

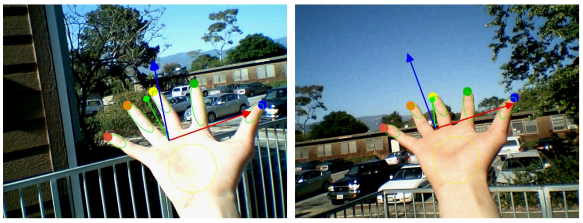

Figure 10. Illumination changes causing different hand skin color (a) and (b) for outdoor scenes.

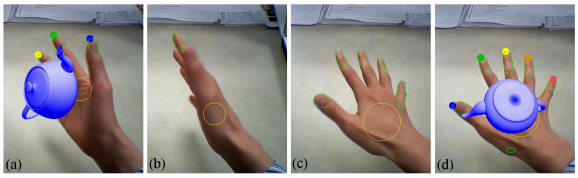

Figure 11. Flipping the hand from (a) to (d). (b) Fingertips are not detected because of self-occlusion. In (c), fingertips are detected again, and the object is rendered again in (d).

template-based detection is that it locates the detected fingertip point at the center of the finger, while the contour-based approach locates it on the outer edge, which may be inaccurate for some unfavorable viewing angles. In order to cope with that issue, we introduced the ellipse fitting algorithm to accurately locate the fingertip independently of the viewing direction. The experimental results in section 3.1 show that the ellipse fitting effectively and efficiently locates the fingertip for establishing the hand coordinate system.

Since we are using only fingertips as point correspondences for camera pose estimation, we have limitations due to possible self occlusions, as shown in Figure 11. When fingertips are not visible, our system determines that it has lost tracking the fingertips as in Figure 11b, and tries to detect fingertips again as in Figure 11c. While this recovery happens promptly enough to not be overly disruptive in wearable applications, we have started to investigate use of more features on the hand and silhouette-based approaches such as active shape models [6] or smart snakes [11] in order to deal with more articulated hand poses and self occlusions.

5. Conclusions and Future Work

We introduced a real-time six-degree-of-freedom camera pose estimation method using fingertip tracking. We segment and track the hand region based on adaptively learned color distributions. Accurate fingertip positions are then located on the contour of the hand by fitting ellipses around the segments of highest curvature. Our results show that camera pose estimation from the fingertip locations can be effectively used instead of marker-based tracking to implement a tangible UI for wearable computing and AR applications.

For future work, we want to investigate how to improve accuracy by including more feature points on the hand, and, especially for the outdoor case, we want to improve the hand segmentation algorithm. In addition to a single hand pose, we would like to experiment with different poses, triggering different actions, and combining our method with other gesture-based interfaces. We also want to investigate how to best handle the occlusion between virtual objects and the hand, as [20] has started to do. Moreover, we are currently extending our method to initialize a global AR coordinate system to be used with markerless natural feature tracking for larger 3D spaces, such as a tabletop AR environment. Finally, we are planning to distribute Handy AR as an Open Source Library in the near future.

6. Acknowledgements

This research was supported in part by the Korea Science and Engineering Foundation Grant (#2005-215-D00316) and by a research contract with the Korea Institute of Science and Technology (KIST) through the Tangible Space Initiative project.

References

[1] A. A. Argyros and M. I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *European Conference on Computer Vision*, pages Vol III: 368–379, 2004.

[2] A. A. Argyros and M. I. A. Lourakis. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Computer Vision in Human-Computer Interaction*, pages 40–51, 2006.

[3] R. T. Azuma. Predictive tracking for augmented reality. Technical Report TR95-007, Department of Computer Science, University of North Carolina - Chapel Hill.

[4] R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, Nov./ Dec. 2001.

[5] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34:344–371, 1986.

[6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[7] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. In *Proc. ISWC '97 (First Int. Symp. on Wearable Computers)*, pages 74–81, Cambridge, MA, Oct. 13–14 1997.

[8] M. Fiala. Artag, a fiducial marker system using digital techniques. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.

[9] E. Foxlin and M. Harrington. Weartrack: A self-referenced head and hand tracker for wearable computers and portable vr. In *Proc. ISWC '00 (Fourth Int. Symp. on Wearable Computers)*, pages 155–162, Atlanta, GA, Oct. 16–17 2000.

[10] M. Fukumoto, Y. Suenaga, and K. Mase. Finger-Pointer: Pointing interface by image processing. *Computers and Graphics*, 18(5):633–642, 1994.

[11] A. Heap. Real-time hand tracking and gesture recognition using smart snakes. In *Interface to Human and Virtual Worlds*, 1995.

[12] T. Höllerer and S. Feiner. Mobile augmented reality. In H. Karimi and A. Hammad, editors, *Telegeoinformatics: Location-Based Computing and Services*. Taylor and Francis Books Ltd., London, UK, 2004.

[13] T. Höllerer, J. Wither, and S. DiVerdi. *Anywhere Augmentation: Towards Mobile Augmented Reality in Unprepared Environments*. Lecture Notes in Geoinformation and Cartography. Springer Verlag, 2007.

[14] Intel Corporation. Open Source Computer Vision Library reference manual. December 2000.

[15] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *CVPR*, pages 1274–1280. IEEE Computer Society, 1999.

[16] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, Oct. 1999.

[17] M. Kölsch, A. C. Beall, and M. Turk. The postural comfort zone for reaching gestures. Technical report, University of California, Santa Barbara, Computer Science, Aug. 29 2003.

[18] M. Kölsch and M. Turk. Fast 2D hand tracking with flocks of features and multi-cue integration. In *Vision for Human-Computer Interaction*, page 158, 2004.

[19] T. Kurata, T. Okuma, M. Kourogi, and K. Sakaue. The hand mouse: GMM hand-color classification and mean shift tracking. In *Second Intl. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems*, 2001.

[20] W. Lee and J. Park. Augmented foam: A tangible augmented reality for product design. In *ISMAR*, pages 106–109. IEEE Computer Society, 2005.

[21] Letessier, Julien and Berard, Francois. Visual tracking of bare fingers for interactive surfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Interactive surfaces, pages 119–122, 2004.

[22] W. W. Mayol, A. J. Davison, B. J. Tordoff, N. D. Molton, and D. W. Murray. Interaction between hand and wearable camera in 2D and 3D environments. In *British Machine Vision Conference*. BMVA, Sept. 2004.

[23] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, 2002.

[24] W. Piekarski and B. Thomas. Tinmith-Metro: New outdoor techniques for creating city models with an augmented reality wearable computer. In *Proc. ISWC '01 (Fifth Int. Symp. on Wearable Computers)*, pages 31–38, Zürich, Switzerland, Oct. 8–9 2001.

[25] I. Poupyrev, D. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani. Developing a generic augmented-reality interface. *Computer*, 35(3):44–50, March 2002.

[26] G. Reitmayr and T. W. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. In *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 109–118, 2006.

[27] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. Picard, and A. Pentland. Augmented reality through wearable computing. *Presence*, 6(4):386–398, Aug. 1997.

[28] T. Starner, J. Weaver, and A. Pentland. A wearable computing based american sign language recognizer. In *Proc. ISWC '97 (First Int. Symp. on Wearable Computers)*, pages 130–137, Cambridge, MA, Oct. 13–14 1997.

[29] B. D. R. Stenger. Template-based hand pose recognition using multiple cues. In *Asian Conference on Computer Vision*, pages II:551–560, 2006.

[30] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Visual hand tracking using nonparametric belief propagation. In *Workshop on Generative Model Based Vision*, page 189, 2004.

[31] B. H. Thomas and W. Piekarski. Glove based user interaction techniques for augmented reality in an outdoor environment. *Virtual Reality: Research, Development, and Applications*, 6(3):167–180, 2002. Springer-Verlag London Ltd.

[32] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *Proc. ISWC '03 (Seventh Int. Symp. on Wearable Computers)*, pages 127–137, White Plains, NY, Oct. 21–23 2003.

[33] G. Welch and G. Bishop. An introduction to the kalman filter. In *Technical Report*. University of North Carolina at Chapel Hill, 1995.

[34] G. Ye, J. Corso, G. Hager, and A. Okamura. Vishap: Augmented reality combining haptics and vision. In *International Conference on Systems, Man and Cybernetics*, pages 3425–3431. IEEE Computer Society, 2003.

[35] Z. Y. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov. 2000.