NATURAL INTERACTION WITH VIRTUAL PET ON YOUR PALM

Junyeong Choi, Jae-Hyek Han, Byung-Kuk Seo, Hanhoon Park, and Jong-Il Park

Department of Electronics and Computer Engineering
Hanyang University
Seoul, Korea
E-mail: {hooeh, turbostar, nwseoweb, hanuni}@mr.hanyang.ac.kr, jipark@hanyang.ac.kr

ABSTRACT

We present an augmented reality (AR) application for cell phone where users put a virtual pet on their palms and play/interact with the pet by moving their hands and fingers naturally. The application is fundamentally based on hand/palm pose recognition and finger motion estimation, which is the main concern in this paper. We propose a fast and efficient hand/palm pose recognition method which uses natural features (e.g. direction, width, contour shape of hand region) extracted from a hand image with prior knowledge for hand shape or geometry (e.g. its approximated shape when a palm is open, length ratio between palm width and pal height). We also propose a natural interaction method which recognizes natural motion of fingers such as opening/closing palm based on fingertip tracking. Based on the proposed methods, we developed and tested the AR application on an ultra-mobile PC (UMPC).

Keywords: - Augmented Reality, Mobile Application, Hand Pose Estimation

1. INTRODUCTION

The processing power of the mobile devices such as cell phone and personal digital assistant (PDA) has been steadily increased and their portability has provoked augmented reality (AR) researchers to consider them as a powerful platform. In this regard, recently, a number of mobile AR applications have been developed, e.g. providing the direction and location of crosswalks [7], showing the location of underground water pipes [8], and tracking multiple people for robot navigation [11].

In this paper, we present a new mobile AR application, called *virtual pet on your palm*. As shown in Fig. 1, a virtual pet is put on one's palm, and then he/she interacts with it by moving his/her hand or fingers, e.g. touching, stroking, etc. A few years ago, Lee and Höllerer proposed a similar application [2] where users can inspect an AR object put on his/her hand. However, they paid less attention on providing user interaction with the AR object because user interaction was not an important issue in their application. In contrast, in our application, natural interaction with a virtual pet is an important factor and we use finger motions as a natural interface.

Our application is fundamentally based on two methods. One is a palm pose estimation method for estimating the virtual pet's pose. The other is a fingertip tracking method for interacting with the virtual pet. These two methods are the main concern of this paper.

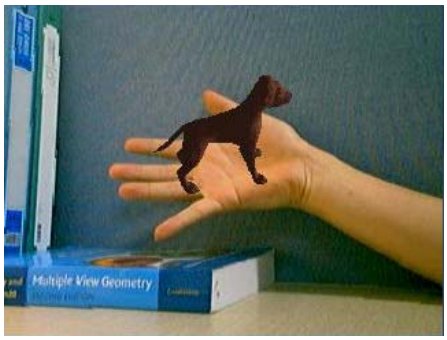

Fig. 1. Virtual pet on your palm.

There have been lots of palm pose estimation methods and fingertip tracking methods in the literature. The palm pose estimation methods can be divided into 3-D model based methods [13, 15, 16] and model-free methods. 3-D model based methods allow free finger motion but are not suitable for mobile devices because they are based on complicated processes in many steps and thus are difficult to work in real-time [9]. One of model-free methods uses a color band for fast and easy hand tracking [12]. However, they may make users feel inconvenient by enforcing wearing band on his/her wrist. Some model-free methods [14, 17, 18, 19] use hand silhouette images for efficient palm pose estimation. However, they suffer from self-occlusion problem. Also, finger's motion will change the hand silhouette and thus influence the palm pose in a wrong way.

The fingertip tracking methods can be divided into device-based methods [20, 21] and vision-based methods [10, 22]. Device-based methods can track user's fingers robustly in uncontrolled environments but may make users feel inconvenient by enforcing wearing devices on his/her fingers. Most vision-based methods can track user's fingers fast and accurately in well-controlled environments but suffer from the inherent problems such as self-occlusion and cluttered background in less-controlled environments.

Keeping the pros and cons of the palm pose estimation methods and fingertip tracking methods in mind, we propose a new palm pose estimation method. Our method is basically a model-free and silhouette-based one and estimates palm pose based on natural features (starting

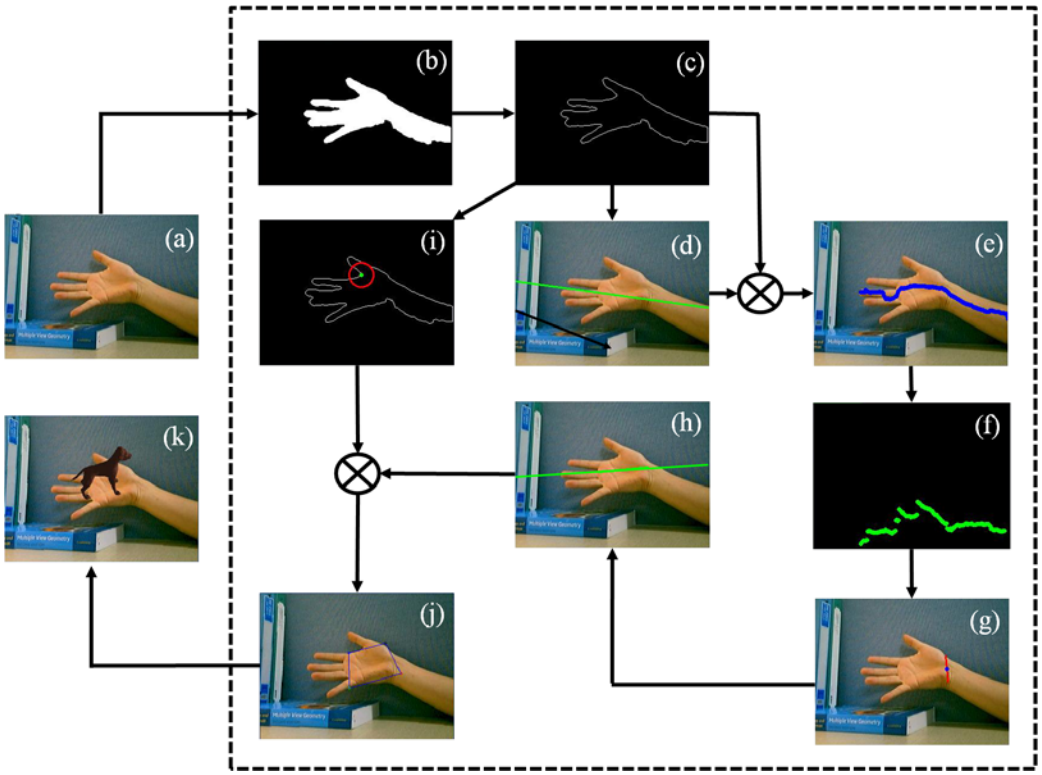

Fig. 2. Process flow for palm pose estimation. (a) captured scene, (b) segmented hand region silhouette, (c) contour of the hand region, (d) mean direction of the hand region (green line), (e) mean points of the hand region (blue points), (f) distance map, (g) separation the palm from the forearm, (h) dominant direction of the palm, (i) detection of the convexity defect point between the thumb and the indexfinger, (j) pose estimation of the hand. (k) rendering a virtual pet.

point of the forearm, direction, and convexity defect point between the thumb and the indexfinger) which are rarely influenced by finger's motions. Because the proposed method minimizes the influence of finger's motions, it exactly estimates palm pose in situation of freely moving fingers. This advantage can also allow natural interaction based on finger motions. To track finger motions, we propose a fast and efficient fingertip tracking method which is also a silhouette-based one and estimates the partial contour angles of hand silhouette.

2. PALM POSE ESTIMATION

To augment a 3-D virtual pet on user's palm, the hand region silhouette should be detected first in a scene image. Note that in our method, the hand region includes both a hand and a part of a forearm. The proposed method detects the hand region by segmenting the scene image using the generalized statistical color model [1] and eliminates skin-colored background using Distance Transform [3]. With the segmented hand region, hand pose (i.e. position and orientation) is estimated by the following procedure.

a. Find a mean direction of the hand region by applying least square line fitting to its contour points. Then, separate the palm from the forearm using the mean points which are computed by two points where the hand region contour meets with the line orthogonal to the mean direction. The starting point of the forearm is determined by a point where differentials of the mean points are starting to be constant.

b. Find the palm direction by applying least square line fitting to the mean points of the palm region.

c. Compute the ratio of the palm's reference lengths by using the convexity defect point between the thumb, the indexfinger, and the starting point of the forearm.

d. Determine a projection model using a ratio of the palm's reference lengths. Then, the hand pose is estimated and a 3-D virtual pet is rendered on the palm of the hand

This process flow is also depicted in Fig. 2 and the detailed explanation is given in the following sub-sections.

2.1 Hand Region Segmentation

For segmenting the hand region from a scene image, we use the generalized statistical color model for classifying skin color pixels from non-skin color pixels. Based on the model, each pixel is determined to be in the hand region if the skin color likelihood is larger than a constant threshold. Fig. 3-(b) shows a result of skin color detection. As shown in Fig. 3-(b), background which has a pixel value similar to skin is classified to skin. Because we assume that users see the right hand through the mobile screen at a close distance holding a mobile device on the left hand, the majority

portion of the skin color segmented image is the hand region. In order to detect the majority portion, we use Distance Transform [3] (Fig. 3-(c)). Fig. 3-(d) shows the detected hand silhouette image by using the Distance Transform.

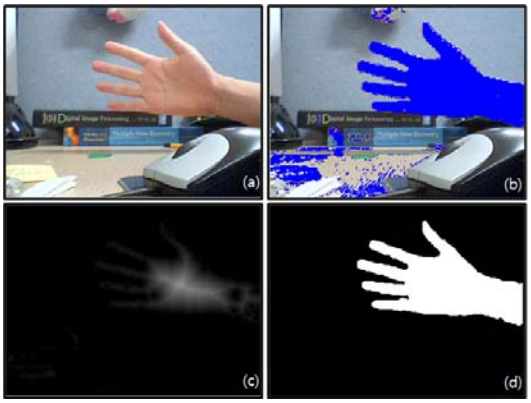

Fig. 3. (a) Captured scene, (b) skin color segmentation result, (c) distance transform, (d) detected hand region silhouette.

2.2 Pose Estimation

Detecting the starting point of the forearm is the first step for estimating palm pose. The starting point detection consists of four steps. First, we find the mean direction of the hand region (green line in Fig. 4-(a)). The mean direction is found by applying line fitting to the points on silhouette contour as shown in Fig. 4-(b). Second, we find the mean points (blue points in Fig. 4-(a)) from two orthogonal points where the hand silhouette contour meets with the line orthogonal to the mean direction. Then, we compute the distances (Fig. 4-(c)) between two orthogonal points where the hand silhouette contour meets with the line orthogonal to the local direction of the mean points. Here, the local direction is computed by line fitting of a target point and its neighborhood points (when the number of neighborhood points are about 15, it works well in practice). Because of the difference between the palm direction (blue line in Fig. 4-(d)) and the mean direction of hand region, using the local direction of the mean points is more correct than using the global mean direction of hand region. Finally, the starting point of the forearm is determined as a point where differentials of the distances (fig.4-(c)) are starting to be constant (blue arrow between Figs. 4-(a) and 4-(c)) after starting from the max point (red arrow between Figs. 4-(a) and 4-(c)).

The next step is finding the palm direction. Because the mean direction of hand region is not determined by only palm direction but also forearm direction, using the mean direction of hand region as it is may cause wrong palm direction. Thus, palm direction is computed by using only the mean points (blue points in Fig. 4-(a)) which are upper than the starting point of the forearm. As shown in Fig. 4-(d), the palm direction is accurately computed even if the palm direction significantly differs from the mean direction of hand region.

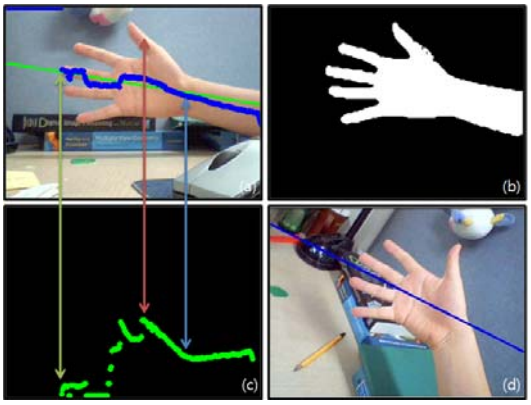

Fig. 4. (a) Mean direction (green line) and mean points (blue points), (b) hand region silhouette, (c) the y-values of green points indicate the distances between the two orthogonal points and their mean points, (d) palm direction (blue line).

The third step is detecting the convexity defect point between the thumb and the indexfinger. The candidate points (Fig. 5-(a)) of the convexity defect point are detected from the contour of hand silhouette using a curvature-based algorithm similar to the one described in [4, 10]. If the angle between the line P_iP_{i+l} and the line P_iP_{i-l} is lower than a threshold value, the point P_i is classified to a candidate point of the convexity defect point. Here, the points P_{i+l} and P_{i-l} indicate the points which are l-indices distant from the point P_i, respectively. The angle of the point P_i is computed by dot product of the line P_iP_{i+l} and the line P_iP_{i-l} as

$$\theta_l(p_i) = \frac{\overrightarrow{P_iP_{i-l}} \cdot \overrightarrow{P_iP_{i+l}}}{\left\|\overrightarrow{P_iP_{i-l}}\right\| \left\|\overrightarrow{P_iP_{i+l}}\right\|} \quad (1)$$

For adapting to scaling, we compute the angles using various l values. The points (blue points in Fig. 5-(a)) which are relatively close to the starting point of the forearm are neighborhoods of the convexity defect point. For detecting the accurate location of the convexity defect point, we fit the relatively close candidate points to an ellipse (Fig. 5-(b)) by using least square fitting offered by OpenCV [5]. We then compute the intersection points of the ellipse's major axis with its edge and pick the one closer to the starting point of forearm.

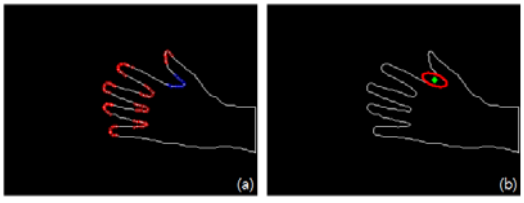

Fig. 5. (a) Candidate points of the convexity defect point, (b) ellipse fitting.

Finally, we estimate the palm pose using the palm direction,

the starting point of forearm, and the convexity defect point found before. A palm width (L_2 in Fig. 6-(a)) is computed by the distance between two orthogonal points where the line, which is orthogonal to the palm direction (L_3 in Fig. 6-(a)) and passes through the convexity defect point, intersects with the hand contour. A palm height (L_1 in Fig. 6-(a)) is computed by the distance between height (L_2 in Fig. 6-(a)) and the starting point of forearm. Then, the palm pose is estimated from the palm height and width by the following procedures.

a. Consider a virtual 3-D rectangle model for estimating the palm pose. 3-D coordinates (x, y, z) of the rectangle's four corners are computed using the palm width and height.
   a-1. 3-D rectangle model's height and width have to equal in the front view. However, as shown in Fig.6-(a), the palm height (L_1) and the palm width (L_2) are different in the front view of palm. Thus, multiply the palm height by a constant a (set to 1.4742 in our experiments) for making the width and the height same.
   a-2. Decide the coordinates by the follow equation.

$$
\begin{aligned}
&\textit{if } W > H, \\
&P_1 = (-W, H, -Z),\ P_2 = (W, H, -Z), \\
&P_3 = (-W, -H, Z),\ P_4 = (W, -H, Z), \\
&\textit{otherwise}, \\
&P_1 = (-W, H, Z),\ P_2 = (W, H, -Z), \\
&P_3 = (-W, -H, Z),\ P_4 = (W, -H, -Z), \\
&\textit{where} \\
&W = width/2,\ \ H = height * a/2, \\
&Z = \sqrt{(\max(W,H)^2 - \min(W,H)^2}.
\end{aligned}
\tag{2}
$$

b. Project four corner points onto a plane z=0. Then, rotate the points about the angle between the palm direction (L_3) and the north direction (N).

c. Translate the center of the four projected points into that of the palm's center where the width and height intersects with each other. The projected rectangle model is correctly located on a user's palm (Fig. 6).

d. Use the function arGetTransMat() in ARToolKit [6] for computing the palm pose from the center point, four corner points, and four edge lines of the projected model.

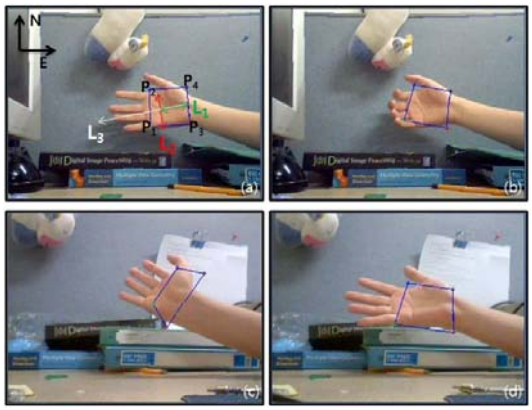

Fig. 6. Rectangle model projected onto a palm with different poses.

As shown in Fig. 7, a virtual teapot is correctly drawn on a user's palm once the palm pose is estimated.


Fig. 7. Virtual teapot on a palm with different poses.

3. Finger Motion Detection

We use a simple method for tracking finger motions, which is similar to the convexity defect point detection method in Section 2.2. First, we detect the candidate fingertip points using the curvature-based algorithm as shown in Fig. 8. Next, we find the points (blue points) which are relatively away from the starting point of the forearm. From the blue points, we find the exact location of fingertips using ellipse fitting. When the user stretches his/her hand, fingertips are easily detected using the method explained here. However, if the user clenches his/her fist, fingertips will not be detected. Therefore, we can recognize if the user stretches his/her hand or clenches his/her fist using the number of blue points.

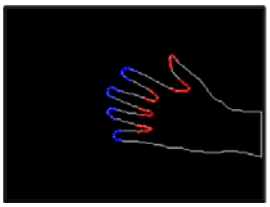
Fig. 8. Candidate fingertip points.

The preliminary results are shown in Fig. 9. In our demonstration on a UMPC (Sony, VGN-UX27LN), when the palm of a hand is opened, a flower is opened and a bee is coming out and bussing around it. On the other hand, when the palm of the hand is closed, the flower is closed and the bee disappears

Fig. 9. Interaction with a virtual flower using finger motion.

4. Conclusion and Future Works

This paper proposes a fast and efficient palm pose estimation method and a natural interaction method for mobile AR application called virtual pet on your palm. Our palm pose estimation method could estimate palm pose exactly and fast on mobile platform. Our interaction method could provide users natural interaction with virtual pet on their palm by moving their fingers.

Our current system is limited to basic interactions such as open/closing palm, but we will try to provide more natural interactions in the near future. Also, occlusion processing will be provided when users put their fingers over the virtual pet.

Acknowledgements

This work was supported by the IT R&D program of MKE/IITA. [2008-F-042-01, Development of Vision/Image Guided System for Tele-Surgical Robot]

REFERENCES

[1] M. J. Jones and J. M. Rehg, “Statistical color models with application to skin detection,” Proc. of CVPR’99, pp. 1274–1280, 1999.

[2] T. Lee and T. Höllerer, “Handy AR: Markerless inspection of augmented reality objects using fingertip tracking,” Proc. of ISWC’07, 2007.

[3] G. Borgefors, “Distance transformations in digital images,” Computer Vision, Graphics and Image Processing, vol. 34, pp. 344–371, 1986.

[4] A. A. Argyros and M. I. A. Lourakis, “Vision-based interpretation of hand gestures for remote control of a computer mouse,” Computer Vision in Human-Computer Interaction, pp. 40–51, 2006.

[5] Intel Corporation, Open Source Computer Vision Library reference manual, 2000.

[6] ARToolkit, available at: http://www.hitl.washington.edu/artoolkit/.

[7] V. Ivanchenko, J. Coughlan, and H. Shen, “Detecting and locating crosswalks using a camera phone,” Proc. of CVPR’08, 2008.

[8] G. Schall, H. Grabner, M. Grabner, P. Wohlhart, D. Schmalstieg, and H. Bischof, “3D tracking in unknown environments using on-line keypoint learning for mobile augmented reality,” Proc. of CVPR’08, 2008.

[9] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, “Vision-based hand pose estimation : A review,” Computer Vision and Image Understanding, pp. 52-73, 2007.

[10] K. Oka, Y. Sato, and H. Koike, “Real-time fingertip tracking and gesture recognition,” IEEE Computer Graphics and Applications, vol. 22 no. 6, pp. 64–71, 2002.

[11] A. Ess, B. Leibe, K. Schindler, and L. V. Gool, “A mobile vision system for robust multi-person tracking,” Proc of CVPR’08, 2008.

[12] R. Lockton and A. Fitzgibbon, “Real-time gesture recognition using deterministic boosting,” Proc. of BMVC’02, pp. 817-826, 2002.

[13] V. Athitsos and S. Sclaroff, “Database indexing methods for 3D hand pose estimation,” Proc. of Gesture Workshop’03, pp. 288-299, 2003.

[14] L. Bretzner, I. Laptev, and T. Lindeberg, “Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering,” Proc. of FG’02, pp. 405-410, 2002.

[15] B. Stenger, P. R. S. Mendonca, and R. Cipolla, “Model-based hand tracking using an unscented Kalman filter,” Proc. of British Machine Vision Conference’01, vol. I, pp 63-72, 2001.

[16] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis, “Using multiple cues for hand tracking and model refinement,” Proc. of CVPR’03, pp. II: 443-450, 2003.

[17] C. Schwarz and N. Lobo, “Segment-based hand pose estimation,” Proc. of 2nd Canadian Conf. Computer and Robot Vision’05, pp. 42–49, 2005.

[18] J. Segen and S. Kumar, “Shadow gestures: 3D hand pose estimation using a single camera,” Proc. of CVPR’99, pp. 479-485, 1999.

[19] Z. Mo and U. Neumann, “Real-time hand pose recognition using low-resolution depth images,” Proc. of CVPR’06, vol. 2, pp. 1499-1505, 2006.

[20] M. Bezdicek and D. G. Caldwell, “Potable Absolute Position Tracking System for Human Hand Fingertips,” Proc. of VC’06, 2006.

[21] T. Grossman, D. Wigdor and R. Balakrishnan, “Multi-Finger Gestural Interaction with 3D Volumetric Displays,” Proc. of UIST’04, 2004.

[22] J. Letessier and F. Bérard, “Visual Tracking of Bare Fingers for Interactive Surfaces,” Proc. of UIST’04, 2004.