SNAP&TELL: A Vision-Based Wearable System To Support 'Web-On-The-World' Applications

Trish Keaton[1]
Information Sciences Lab.
HRL Laboratories LLC
Malibu, CA 90265, USA
pakeaton@hrl.com

Sylvia M. Dominguez[1]
HRL Laboratories LLC &
University of California
Los Angeles, CA 90095, USA
sylvia@ee.ucla.edu

Ali H. Sayed[2]
Electrical Engineering Dept.
University of California
Los Angeles, CA 90095, USA
sayed@ee.ucla.edu

Abstract

This paper gives an overview of a vision-based wearable computer system 'SNAP&TELL'[1] , which performs real-time gesture tracking for recognizing objects in the scene including outdoor landmarks. Our system uses a single camera to capture images which are processed using several algorithms to perform segmentation based on color, fingertip shape analysis, robust tracking, and invariant object recognition, in order to quickly identify the objects encircled by a user's pointing gesture. In turn, the system returns information concerning the object such as its classification, historical facts, etc. This system provides enabling technology for the design of intelligent assistants to support "Web-On-The-World" applications, with potential uses such as travel assistance, business advertisement, the design of smart living and working spaces, and pervasive wireless services and internet vehicles.

1. Introduction

In the future, computing technology is expected to greatly impact our daily activities. One recent computing trend is mobile wearable computing for the design of intelligent assistants to provide location-aware information access which can help users more efficiently accomplish their tasks. Thus imagine a user driving by a hotel or a restaurant while on a foreign trip. By pointing at either establishment, the assistant would be able to convey to the driver recommendations about the hotel or the restaurant menu and its hours of operation. In a not so distant future, a paramedic using a wearable system will be able to receive assistance from a Virtual Medical Aid by pointing at the injures on a victim, and getting suggestions on the most suitable treatment to apply to the particular situation. Computing and sensing in such environments must be reliable, persistent (always remains on), easy to interact with, and configured to support different needs and complexities. The success of such systems will rely upon the ability to quickly process the sensory data captured from all sensors, and automatically extract the relevant information for analyzing and understanding the objects and activities occurring within the environment. For scene understanding within wearable environments, we have developed a real-time gesture tracking system 'SNAP&TELL' for recognizing objects in the scene.

Visual tracking and recognition of pointing and hand gestures are essential to interacting with a wearable system. Therefore, the 'SNAP&TELL' system uses several computer vision algorithms to extract color-based segmentations, and shape information from the machine's camera view in order to identify the user's hand and fingertip position. These algorithms, however, are complex and computationally intensive, and thus tend to slow down the response of the machine to a great extent. In order to perform real-time acquisition and tracking, 'SNAP&TELL' uses a robust state-space estimation algorithm to predict the future position of the user's pointing fingertip. Then, the system uses these predicted coordinates to center a smaller search window during the next video frame. This reduces the search space from the full camera view to a smaller area in a dynamic fashion.

The need for a robust prediction algorithm arises from the desire to control the influence of uncertain environmental conditions on our system's performance. For a wearable computer system, these uncertainties arise from the camera moving along with the user's head motion, the background and object moving independently of each other, the user standing still then randomly walking, and the user's pointing finger abruptly changing directions at variable speeds. All these factors give rise to uncertainties that can influence

[2]The work of A. H. Sayed was partially supported by NSF award ECS-9820765.

the design of reliable trackers, therefore we have incorporated data uncertainty modeling into SNAP&TELL's robust tracking algorithm. Once the user has finished encircling the object of interest, our system uses an invariant object recognition algorithm to identify the desired subject, and provide the user with all pre-stored information concerning that particular object.

2 Previous work

In the past, the applicability of computer vision algorithms aimed at real-time pattern recognition and object tracking has been hindered by the excessive memory requirements and slow computational speeds. Some recent computer vision approaches for tracking applications speed up their computation time by reducing the image search area into a smaller window. The window is centered around the last known position of the moving object [1], [10]. The main drawback of these methods is that when the object moves faster than the frame capture rate of the algorithm, the object will move out of the window range. This possibility leads to a loss in tracking ability and forces the algorithm to reset the image search area to the full view of the camera in order to recover the position of the object. The repeated reduction and expansion of the image search area slows down the system performance considerably. Some tracking solutions have attempted an improvement by gradually varying the search window's size according to the moving object speed [1]. The faster the object moves, the larger the search window becomes, while still centering the window around the last known position of the object. Therefore, if the object is moving fast, the search window is large and the computation time for the vision algorithm increases, thus further slowing down the system's response time.

More advanced systems, such as in [5], use state-space estimation techniques to center the smaller search window around a future predicted position of the user's fingertip, rather than around its current position. In this way, as the moving object speed increases, the predicted window position will accompany the speeding object thereby keeping it inside the window's view. The window size thus remains small and centered around the object of interest regardless of its speed. This in turn keeps the memory allocations to a minimum, thus freeing memory space that can be used by other simultaneous applications. However, if the object abruptly changes its movement patterns (which introduces modeling uncertainties), such systems breakdown, and tracking of the user's hand is lost. Therefore, a robust estimation algorithm such as [4], which models the uncertainties created by the user's random ego motion, is more effective in keeping the user's hand inside the small search window and in reducing the number of times the image search area has to be expanded to full view, thus increasing the system's response time.

3 SNAP&TELL system overview

At HRL, we have designed a wearable computer system 'SNAP&TELL', which aims at providing a gesture-based interface between the user and the mobile computer. With this goal in mind, we have developed a robust algorithm to track the position of the tip of a user's pointing finger. This finger tracker acts as an interface to our wearable computing system, which enables a user to specify, segment, and recognize objects of interest such as landmarks, by simply pointing at and encircling them with their fingertip. The 'SNAP&TELL' system accepts input from a color pencil camera, then applies color segmentation to each input stream. The color segmented image is then fed into a skin/non-skin discrimination algorithm to detect likely skin toned regions, then shape and curvature analysis is used to extract the hand and to determine the coordinate position of the fingertip. The sequence of successive detected fingertip positions identifies the trajectory that the user's fingertip is following while encircling the object of interest. At the conclusion of the hand motion gesture, the algorithm determines if an object has been selected by the user, and extracts it from the scene, by cropping the region of interest. The segmented object is then compared against a database of pre-stored objects, by using an invariant object recognition algorithm which recognizes the object despite small variations in pose, scale, rotation, and translation. Once the object is recognized, the information associated with it is made available to the user. The system block diagram for 'SNAP&TELL' is shown in Figure 1.

This problem is particularly difficult because we need to recognize the user's hands and objects from images taken from head-mounted cameras in real time. When the user's head moves so does the camera, thus introducing image jitters, and dramatical changes in the unrestricted background and the lighting conditions. Therefore, in order to track the user's fingertip position in the presence of ego-motion, we incorporate the knowledge of the dynamics of human motion to create uncertainty models, which are used with a robust estimation algorithm to make the tracking model less sensitive to the random motion produced by the head-mounted camera and temporary occlusions. Furthermore, we use the coordinates of the robust predicted fingertip position as the center of a smaller image search window. From this point onwards, only the input image inside the smaller search window is analyzed by the vision algorithms, thus speeding up the response time of the system, and making it memory and computationally efficient. If, for some reason, the search window fails to display the user's hand, the system resets back to the full camera view.

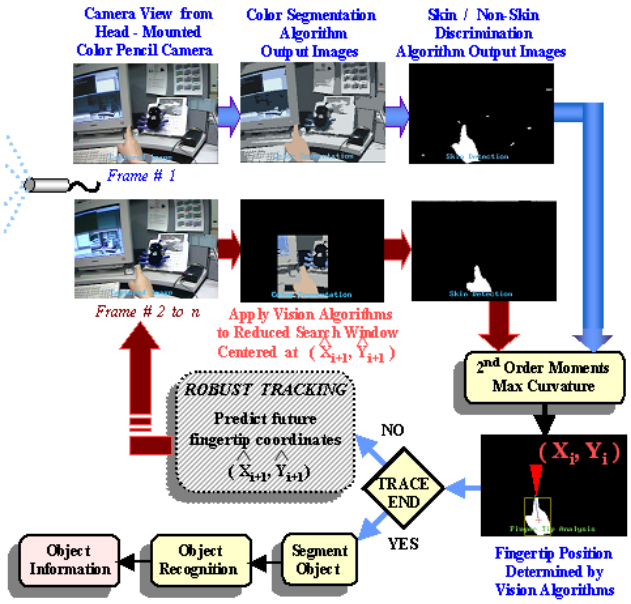

Figure 1. Block diagram of gesture-based interface for the 'Snap&Tell' system.

3.1. Skin/non-skin color segmentation

To determine the skin-like regions in the current frame, we first perform a color segmentation based on the fast and robust mean shift algorithm [2]. By using the mean shift algorithm the number of dominant colors can be determined automatically, unlike the k-means clustering method where the initial number of classes must be chosen. Here, the intensity distribution of each color component in the current frame is viewed as a probability density function. The mean shift vector is the difference between the mean of the probability function on a local area and the center of this region. Mathematically, the mean shift vector associated with a region $S_{\vec{x}}$ centered on $\vec{x}$ can be written as:

$$\vec{V}(\vec{x}) = \frac{\int_{\vec{y}\epsilon S_{\vec{x}}} p(\vec{y})(\vec{y}-\vec{x})d\vec{y}}{\int_{\vec{y}\epsilon S_{\vec{x}}} p(\vec{y})d\vec{y}} \quad (1)$$

where $p(\cdot)$ is the probability density function. The mean shift algorithm states that the mean shift vector is proportional to the gradient of the probability density $\nabla p(\vec{x})$, and reciprocal to the probability density $p(\vec{x})$, such that

$$\vec{V}(\vec{x}) = c\frac{\nabla p(\vec{x})}{p(\vec{x})} \quad (2)$$

where c is a constant. Since the mean shift vector is along the direction of the probability density function maximum, we can exploit this property to find the actual location of the density maximum by searching for the mode of the density. One dominant color can be located by moving search windows in the color space using the mean shift vector iteratively. After removing all color inside the converged search window, one can repeat the mean shift algorithm again to locate the second dominant color. This process is repeated several times to identify a few major dominant colors which segment the image into like-color regions. The dominant colors of the current frame are used as the initial guess of dominant colors in the next frame, thus speeding up the computational time (adjacent frames are usually similar). After segmenting the current frame into homogeneous regions, we determine whether each region is skin-like by considering the mean hue and saturation values and geometric properties of the region. This region-based skin detection procedure is more robust to varying illumination conditions than pixel-based approaches.

3.2. Shape analysis

Once the skin-like regions have been segmented, we clean up this image by applying morphological operations to minimize the number of artifacts being considered as having skin-like color properties. Geometric properties of the skin-like regions are used to identify the hand. Then the user's hand orientation with respect to the x-axis (i.e. pointing direction) is derived using central 2^{nd} order moments, and the fingertip position is determined as the point of maximum curvature along the contour of the hand.

3.3. Robust state-space fingertip tracking

To achieve computational efficiency, memory savings and real-time tracking, a robust state-space estimation algorithm is used to reduce the search area to a smaller search window centered around the predicted position of the fingertip. This robust finger tracker [4] is based on the principles of state-space estimation with uncertain models, see Sayed [7]. The tracker attempts to predict the fingertip coordinate positions $\{x_{i+1}, y_{i+1}\}$ in the next video frame by using the following robust state-space model with state vector s_i and measurement vector z_i.

$$s_i \triangleq \begin{bmatrix} x_i & y_i & v_{x,i} & v_{y,i} & \alpha_{x,i} & \alpha_{y,i} \end{bmatrix}^T \quad (3)$$

$$z_i \triangleq \begin{bmatrix} x_i & y_i \end{bmatrix}^T \quad (4)$$

$$s_{i+1} = (F+\delta F_i)s_i + (G+\delta G_i)u_i \quad (5)$$

$$z_i = Hs_i + v_i \quad (6)$$

where u_i and v_i denote uncorrelated zero-mean white gaussian process and measurement noises, with corresponding covariance matrices Q and R. Moreover, $\{\alpha_{x,i}, \alpha_{y,i}\}$ denote the accelerations along the x and y directions (measured in pixels per second^2), and $\{v_{x,i}, v_{y,i}\}$ denote the speeds along these same directions during the i^{th} frame (measured in pixels/second).

The wearable computer uncertainties are modeled by treating the given parameters $\{F, G\}$ as nominal values, and assuming that the actual values lie within a certain set around them. Thus the perturbations in $\{F, G\}$ in equation (5) are modeled as

$$\left[\begin{array}{cc} \delta F_i & \delta G_i \end{array}\right] = M\Delta_i \left[\begin{array}{cc} E_f & E_g \end{array}\right] \qquad (7)$$

for some matrices $\{M, E_f, E_g\}$ and for an arbitrary contraction Δ_i, $\|\Delta_i\| \leq 1$. For generality, one could allow the quantities $\{M, E_f, E_g\}$ to vary with time as well. This is useful in the case when our model changes dramatically in a particular time instance, such as when the user starts walking, coughing, or moving his/her head abruptly while being distracted. Then one can assign different levels of distortion by selecting the entries of $\{E_f, E_g\}$ appropriately, [4], [7]. The authors are currently investigating adaptive models for modeling the uncertainties associated with user's head motion, walking, and changes in lighting conditions. One such case is when the user starts walking while pointing at an object of interest. In this situation, the uncertainties δF_i and δG_i will have larger values than when the user is standing still. The 'SNAP&TELL' system would then detect constant movement in the camera view, hinting walking motion, and would switch the robust tracker's perturbation model to the "walking" uncertainty model.

Applying the time- and measurement-update form of our robust filter to the uncertainity model (5)–(6), where $\Pi_0 > 0$, $R > 0$, $Q > 0$ are given weighting matrices, yields the following equations, which attempt to minimize the estimation error at the worst case possible created by the bounded uncertainties δF_i and δG_i, see Sayed [7]:

Initial conditions: Set $\hat{s}_{0|0} = P_{0|0} H^T R^{-1} z_0$ and $P_{0|0} = \left(\Pi_0^{-1} + H^T R^{-1} H\right)^{-1}$.

Step 1. If $HM = 0$, then set $\hat{\lambda}_i = 0$ (non robust filter). Otherwise, select α (typically $0 < \alpha < 1$) and set

$$\hat{\lambda}_i = (1 + \alpha) \cdot \|M^T H^T R^{-1} H M\|.$$

Step 2. Replace $\{Q, R, P_{i|i}, G, F\}$ by:

$$\begin{aligned}
\widehat{Q}_i^{-1} &= Q^{-1} + \hat{\lambda}_i E_g^T \left[I + \hat{\lambda}_i E_f P_{i|i} E_f^T\right]^{-1} E_g \\
\widehat{R}_{i+1} &= R - \hat{\lambda}_i^{-1} H M M^T H^T \\
\widehat{P}_{i|i} &= \left(P_{i|i}^{-1} + \hat{\lambda}_i E_f^T E_f\right)^{-1} \\
&= P_{i|i} - P_{i|i} E_f^T (\hat{\lambda}_i^{-1} I + E_f P_{i|i} E_f^T)^{-1} E_f P_{i|i} \\
\widehat{G}_i &= G - \hat{\lambda}_i F \widehat{P}_{i|i} E_f^T E_g \\
\widehat{F}_i &= (F - \hat{\lambda}_i \widehat{G}_i \widehat{Q}_i E_g^T E_f)(I - \hat{\lambda}_i \widehat{P}_{i|i} E_f^T E_f)
\end{aligned}$$

If $\hat{\lambda}_i = 0$, then simply set $\widehat{Q}_i = Q$, $\widehat{R}_{i+1} = R$, $\widehat{P}_{i|i} = P_{i|i}$, $\widehat{G}_i = G$, and $\widehat{F}_i = F$.

Step 3. Update $\{\hat{s}_{i|i}, P_{i|i}\}$ as follows:

$$\begin{aligned}
\hat{s}_{i+1} &= \widehat{F}_i \hat{s}_{i|i} \\
\hat{s}_{i+1|i+1} &= \hat{s}_{i+1} + P_{i+1|i+1} H^T \widehat{R}_{i+1}^{-1} e_{i+1} \\
e_{i+1} &= z_{i+1} - H \hat{s}_{i+1} \\
P_{i+1} &= F \widehat{P}_{i|i} F^T + \widehat{G}_i \widehat{Q}_i \widehat{G}_i^T \\
P_{i+1|i+1} &= P_{i+1} - P_{i+1} H^T R_{e,i+1}^{-1} H P_{i+1} \\
R_{e,i+1} &= \widehat{R}_{i+1} + H P_{i+1} H^T
\end{aligned}$$

We applied this robust algorithm to a typical user's finger tip trajectory and display the results in Figure 2. Note that the reduced search window is centered around the previously predicted fingertip position, and very closely overlaps the actual finger position.

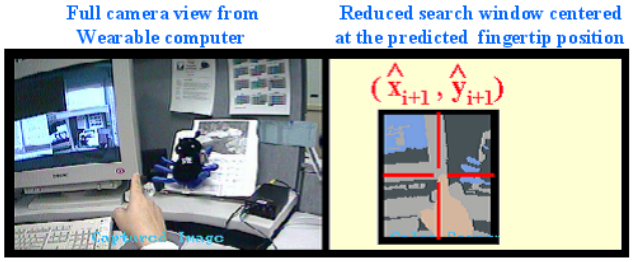

Figure 2. Successfully tracked fingertip using a robust state-space Kalman filter.

3.4. Invariant object recognition

Having located the scene object or landmark of interest, we would like to recognize it irrespective of pose, scale, rotation, and translation variations. Our current approach to object recognition involves a multi-dimensional indexing scheme based on characterizing its local appearance by a vector of features extracted at *salient points*. Local descriptors should be stable to slight changes in viewpoint, illumination, and partial occlusion. It is also desirable that the descriptors be highly discriminant so that objects may be easily distinguished. Crowley *et al.* [3] represented physical objects by an orthogonal family of local appearance descriptors obtained by applying principal component analysis (PCA) to image neighborhoods. The principal components with the largest variance were used to define a space for describing local appearance. Recognition is achieved by projecting local neighborhoods from newly acquired images onto the local appearance space and associating them to descriptors stored in a database. A similar approach to local appearance modeling was proposed by Schneiderman *et al.* [8], where the pattern space was first discretized by applying clustering using Vector Quantization (VQ), and

then a projection basis was learned for each cluster. The approach we take improves upon these methods of modeling local appearance by learning the collection of patterns within a mixture of factor analyzers (MFA) framework, see Keaton *et al.* [6]. The advantages of this approach are that the clustering and dimensionality reduction steps are performed simultaneously within a maximum-likelihood framework. In addition, the MFA model explicitly estimates the probability density of the class over the pattern space. Therefore, it can perform object detection based on the Bayes decision rule.

In our object recognition approach, MFA modeling is used to learn a collection, or mixture, of local linear subspaces over the set of image patches or subregions extracted from the training set for each object class. By allowing a collection of subspaces to be learned, each can become specialized to the variety of structures present in the data ensemble. The cropped image containing the object of interest is first decomposed into a set of 8×8 image patches extracted at salient points. We extract the image patches at only selected points in the image, in order to reduce the amount of data we must process. Salient points are local features where the signal changes two-dimensionally. We use a technique by Tomasi and Kanade [9] for finding salient features. In order to detect an object at any size, we repeat the process of extracting image patches at salient points over a range of magnification scales of the original image.

Factor analysis is a latent variable method for modeling the covariance structure of high dimensional data using a small number of latent variables called factors, where Λ is known as the factor loading matrix. The factors **z** are assumed to be independent and Gaussian distributed with zero-mean unit variance, **z** $\sim \mathcal{N}$ **(0,I)**. The additive noise **u** is also normally distributed with zero-mean and a diagonal covariance matrix Ψ, **u** $\sim \mathcal{N}$ **(0,**Ψ**)**. Hence, the observed variables are independent given the factors, and **x** is therefore distributed with zero mean and covariance $\Lambda'\Lambda + \Psi$. The goal of factor analysis is to find the Λ and Ψ that best model the covariance structure of **x**. The factor variables **z** model correlations between the elements of **x**, while the **u** variables account for independent noise in each element of **x**. Factor analysis defines a proper probability density model over the observed space, and different regions of the input space can be locally modeled by assigning a different mean μ_j, and index ω_j (where *j* = 1,...,M), to each factor analyzer.

The EM learning algorithm is used to learn the model parameters without the explicit computation of the sample covariance which greatly reduces the algorithm's computational complexity:

E-Step: Compute the moments $h_{ij} = E[\omega_j|x_i]$, $E[z|x_i,\omega_j]$, and $E[zz'|x_i,\omega_j]$ for all data points *i* and mixture components *j* given the current parameter values Λ_j, and Ψ_j.

M-Step: This results in the following update equations for the parameters:

$$\widetilde{\Lambda}_j^{new} = \left(\sum_i h_{ij} x_i E[\widetilde{z}|x_i,\omega_j]'\right)\left(\sum_i h_{ij} E[\widetilde{z}\widetilde{z}'|x_i,\omega_j]\right)^{-1}$$
$$\widetilde{\Psi}_j^{new} = \frac{1}{n} diag\left\{\sum_{ij} h_{ij}(x_i - \widetilde{\Lambda}_j^{new} E[\widetilde{z}|x_i,\omega_j])x_i'\right\}$$

See [6] for details on the derivation of these update equations. We iterate between the two steps until the model likelihood is maximized.

In the context of object recognition, we are interested in calculating the probability of the object O_i given a local feature measurement x_k represented by the local image patch or subregion. Once the MFA model is fitted to each class of objects, we can easily compute the posterior probabilities for each subregion x_k. The pdf of the object class O_i is given by

$$p_i(x_k;\theta_i) = \sum_{m=1}^{M} P_{im}\mathcal{N}(\mu_{im}, \Lambda_{im}'\Lambda_{im} + \Psi_{im}),$$

where Θ_i is the set of MFA model parameters for i^{th} object class, and P_{im} is the mixing proportion for the m^{th} model of the object class O_i. The posterior probability of object class O_i given x_k can be calculated by Bayes' rule:

$$P(O_i|x_k) = \frac{P_i p_i(x_k;\Theta_n)}{\sum_{n=1}^{N} P_n p_n(x_k;\Theta_n)}$$

where N is the total number of object classes and P_i is the priori probability of object class O_i which is estimated from the training set of images. Without modeling the dependencies between the local subregions x_k, lets assume we have extracted K independent local feature measurements $(x_1, ..., x_K)$ from an image, then we can compute the probability of each object class O_i given the image patches by

$$P(O_i|x_1,...,x_k) = \frac{P_i p_i(x_1,...,x_k;\Theta_n)}{\sum_{n=1}^{N} P_n p_n(x_1,...,x_k;\Theta_n)} = \frac{\prod_k P_i p_i(x_k;\Theta_n)}{\prod_k \sum_{n=1}^{N} P_n p_n(x_k;\Theta_n)}$$

Then, the optimum object class label $i*$ for the image having a set of local measurements $(x_1, ...x_K)$, is determined by Bayes decision rule as follows:

$$i* = arg\max_i P(O_i|x_1,...,x_K).$$

Figure 3 illustrates the object recognition results obtained with the 'SNAP&TELL' wearable system. Our method has been found to be robust to small changes in viewpoint, scale and 3D rotations.

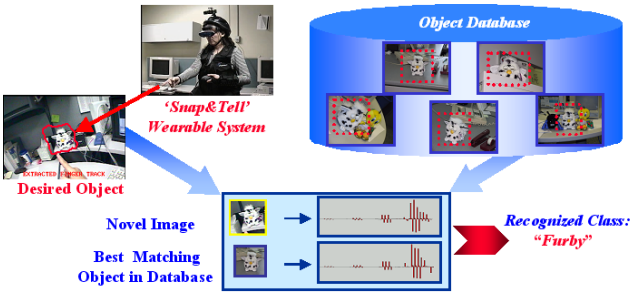

Figure 3. 'Snap&Tell' invariant object recognition.

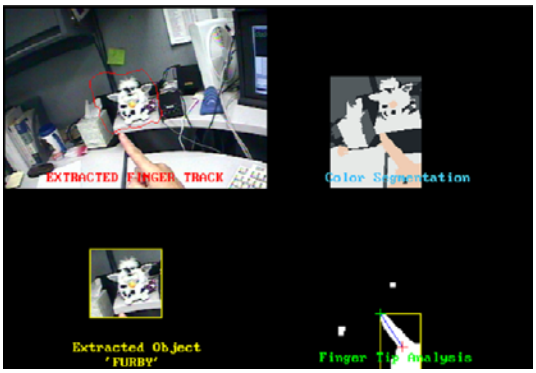

Figure 4. 'Snap&Tell' output display, showing the user's fingertip tracked in 'real-time', and the recognized object of interest.

4 Results

Figure 4 shows the final output display of the 'SNAP&TELL' system, after successfully tracking the user's fingertip, extracting the object of interest at the end of the pointing gesture, and finally recognizing the desired object. This figure also illustrates how the robust tracker helps to reduce the search area into a small window, thereby speeding up the processing of the vision algorithms. In this particular simulation, the response time of our overall system was 68% faster than the response obtained by a system that uses a full camera view to track the user's fingertip, and 23% faster when compared with a system that uses a small search window centered around the previous fingertip position (rather than the predicted future position). It should be noted that the size of the *reduced search window* was chosen to be at least twice the size of the maximum estimation errors in the x and y directions, of our robust Kalman tracker previously applied to a training sequence representative of a typical pointing finger trajectory ($\Delta W_x \geq 2\tilde{x}_{max}, \Delta W_y \geq 2\tilde{y}_{max}$). Therefore, the more accurate the tracker is, the smaller the search window needed, and the faster the overall system response time will be. A comparison of the MSE results between a plain Kalman tracker and our robust Kalman tracker, showed over 15% improvement in the estimation error by using the robust algorithm. These performance results are encouraging and merit future exploration. We are working on an online learning method to develop multiple uncertainty models with an intelligent switching scheme to further speed up our system performance. Finally, our object recognition approach has been found to be robust to changes in scale, illumination, and viewpoint.

References

[1] T. Brown and R. C. Thomas, "Finger tracking for the digital desk", *Proc. Australasian User Interface Conference*, vol. 1, Canberra, Australia, 2000, pp. 11-16.

[2] D. Comaniciu and P. Meer, "Robust analysis of feature space: color image segmentation", *Proc. Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997, pp. 750-755.

[3] V. Colin de Verdiere, and J. L. Crowley, "Visual recognition using local appearance", *Proc. European Conference on Computer Vision*, Frieburg, Germany, 1998.

[4] S. M. Dominguez, T. Keaton, A. H. Sayed, "Robust finger tracking for wearable computer interfacing", *Proc. Perceptive User Interfaces*, Orlando, FL., Nov. 2001.

[5] C. Jennings, "Robust finger tracking with multiple cameras", *Proc. Conference on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, Corfu, Greece, 1999, pp. 152-160.

[6] T. Keaton, and R. Goodman, "A compression framework for content analysis", *Proc. Workshop on Content-based Access of Image and Video Libraries*, Fort Collins, Colo., June 1999, pp. 68-73.

[7] A. H. Sayed, "A framework for state-space estimation with uncertain models", *IEEE Trans. on Automatic Control*, vol. 46, no. 7, July 2001, pp. 998-1013.

[8] H. Schneiderman, and T. Kanade, "Probabilistic modeling of local appearance and spatial relationships for object recognition", *Proc. Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA., 1998, pp. 45-51.

[9] C. Tomasi and T. Kanade, "Detection and tracking of point features", Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburg, PA, April 1991.

[10] J. Yang, W. Yang, M. Denecke, A. Waibel, "Smart sight: a tourist assistant system", *Proc. Intl. Symposium on Wearable Computers*, vol. 1, Oct. 1999, pp.73-78.