

SNAP&TELL: A Multi-Modal Wearable Computer Interface For Browsing The Environment

Trish Keaton¹
Information Sciences Lab
HRL Laboratories LLC
Malibu, CA 90265, USA
pakeaton@hrl.com

Sylvia M. Dominguez
Electrical Engineering Dept.
University of California
Los Angeles, CA 90095, USA
sylvia@ee.ucla.edu

Ali H. Sayed²
Electrical Engineering Dept.
University of California
Los Angeles, CA 90095, USA
sayed@ee.ucla.edu

Abstract

This paper gives an overview of a multi-modal wearable computer system 'SNAP&TELL'¹, which performs real-time gesture tracking combined with audio-based system control commands to recognize objects in the environment including outdoor landmarks. Our system uses a single camera to capture images which are then 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 (SNAPshot) by the user's pointing gesture. In turn, the system returns an audio narration, TELLing the user information concerning the object's classification, historical facts, usage, 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 that can help users more efficiently accomplish their tasks. Thus imagine a soldier using a wearable system during a mission in a foreign country. By pointing at his/her surroundings and snapping landmark images such as buildings, mountains, warning signs, billboards, etc, the wearable assistant could convey up to the minute recommendations about where enemy troops are located with respect to

this current position, including which direction to proceed to, or the language translation of foreign signs. Furthermore, a tourist using a wearable assistant while on a foreign trip, could point at a hotel, a landmark, or a restaurant, in order to obtain information concerning a hotel's rating, room rates and availability, a landmark's highlights and hours of operation, a restaurant's menu/prices and opening hours, or multilingual translations of street signs. In addition, wearable computer technologies can also provide information about the individual that wears the computer, and the world in which this individual interacts (situational awareness). Fire-fighters using a wearable assistant could be provided information concerning the temperature outside of their suit at various positions within their surroundings, their own body temperature and blood oxygen levels, and the recommended direction to proceed while inside a building under low visibility conditions. 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, where the system processes are controlled by the user using a small set of audio commands.

Visual tracking and recognition of pointing and hand gestures are a natural way of 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

¹Copyright(©) 2002 HRL Laboratories, LLC. All rights reserved.)

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

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 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, a verbal command is issued to invoke the system's invariant object recognition algorithm to identify the object, and provide the user with an audio narration of all the previously 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], [11]. 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's 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 the 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 down to a minimum, thus freeing memory space that can be used by other simultaneous processes. 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. The system performs real-time pointing gesture tracking to allow the user to encircle an object of interest in the scene, then a SNAPshot of the object is captured and passed on to a recognition module, which TELLS or outputs audio information concerning the object to the user through the use of IBM's ViaVoice speech recognition and text-to-speech software. 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, by simply pointing at and encircling them with their fingertip. The 'SNAP&TELL' system accepts a constant video input stream from a Toshiba color pencil camera, which is attached to the side hinge of a Sony Glasstron see-through personal LCD monitor. The pencil camera is positioned pointing towards the user's field of view. Once the user is ready to point to the object of interest, he gives the verbal command "start", which activates the finger tracking routine. While tracking the user's fingertip, the system applies color segmentation to the input video 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 user gives the verbal command "stop", which terminates the tracking algorithm. At this point, the currently segmented object encircled by the user is displayed on the see-through personal LCD glasses, and the user is given the choice to either accept this segmented object by using the command "snap", or to "reset" the system and "start" a new snapshot of the object. Once the object has been properly segmented, the user initiates the recognition phase by issuing the verbal command "tell". The recognition algorithm extracts the segmented object 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 object class is displayed and any additional information associated with the object is described to the user thru an audio narration. The system block diagram for 'SNAP&TELL' is shown in Figure 1.

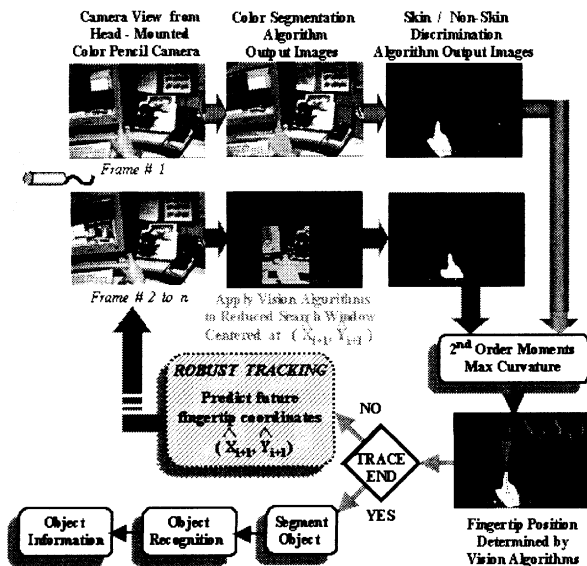


Figure 1. Block diagram of gesture-based interface for the 'Snap&Tell' system.

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 to 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 integrated into a robust estimation algorithm to make the track-

ing model less sensitive to the random motion produced by head/camera motion and temporary occlusions. Furthermore, we use the coordinates of the robust predicted fingertip position to center a smaller image search window for locating the hand. 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 the routine 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.

3.1. Audio interface

For a wearable system to be practical and fully functional, the user interface must be transparent to the users, as well as easy to interact with. Therefore, the 'SNAP&TELL' system uses a headset consisting of head phones and a microphone to verbally communicate with the user in a natural and efficient manner. While the user wears this headset, it enables him to give verbal commands to the computer thru the microphone, while at the same time allows him to hear the information communicated back to him thru audio feedback.

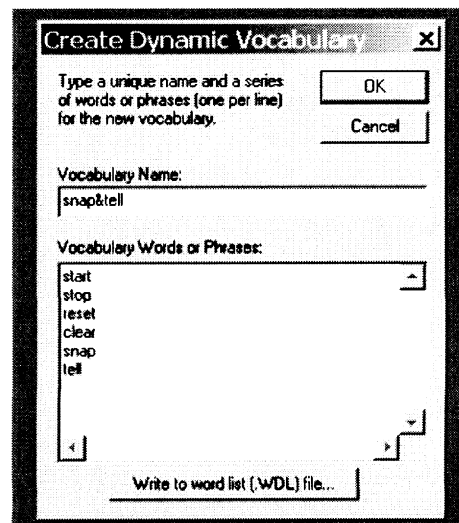


Figure 2. Creating the verbal command list using the IBM's ViaVoice software.

As part of our wearable user interface, we incorporated a series of verbal commands into the system using IBM's ViaVoice software. These commands allow the user to turn the tracking system on and off, as well as give him/her the choice to accept or reject a captured object before it is recognized. Figure 2 shows the ViaVoice dialog window used to create the dynamic vocabulary for

the 'SNAP&TELL' system. The current list of verbal commands include: "start" which enables the system to begin tracking the user's pointing fingertip; "stop" which signals the end of the pointing gesture; "clear" which deletes the partial tracking points computed at any given time thereby allowing the user to erase defective tracking paths; "snap" which extracts the object of interest encircled by the fingertip track; "tell" which activates the recognition routines and ultimately sends the object's audio information to the user; and "reset" which deletes a defective snapshot and clears all the recognition results and sets the system into the "wait for start command" mode. We expect the set of verbal commands to increase as our system continues to grow. Multiple users are supported by training the speech recognition engine on each user, by having him/her say the commands in the list. Figure 3 shows the speech recognition results of a user speaking the various 'SNAP&TELL' commands.

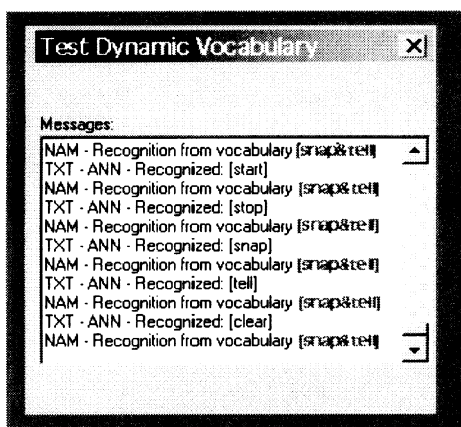


Figure 3. Voice recognition results of the verbal command list for the 'Snap&Tell' system.

3.2. 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:

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

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.

3.3. 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 (e.g., elongatedness, boundary curvature) 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.

3.4. 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 [8]. The need for robust methods arises from the desire to control the influence of uncertain environment conditions on the system performance, such as the effect of random variations in the user's motion characteristics. The ro-

bust tracker attempts to predict the fingertip coordinate positions $\{x_{i+1}, y_{i+1}\}$ in the next video frame in terms of the present frame fingertip pixel coordinates $\{x_i, y_i\}$ and the pixel-shift per frame estimated by $\{v_i T, \alpha_i \frac{T^2}{2}\}$ such as,

$$x_{i+1} \approx x_i + v_{x,i}T + \alpha_{x,i} \frac{T^2}{2} \quad (3)$$

$$y_{i+1} \approx y_i + v_{y,i}T + \alpha_{y,i} \frac{T^2}{2} \quad (4)$$

$$v_{x,i+1} \approx v_{x,i} + \alpha_{x,i}T \quad (5)$$

$$v_{y,i+1} \approx v_{y,i} + \alpha_{y,i}T \quad (6)$$

where $\{\alpha_{x,i}, \alpha_{y,i}\}$ denote the accelerations along the x and y directions (measured in pixels per second²), $\{v_{x,i}, v_{y,i}\}$ denote the speeds along these same directions during the i^{th} frame (measured in pixels/second), and T denotes the frame capture rate while tracking the user's hand, (for the 'SNAP&TELL' wearable system, this rate is currently 1/5 seconds/frame). The previous equations motivate the following state-space model with state vector s_i and measurement vector z_i :

$$s_i \triangleq [x_i \ y_i \ v_{x,i} \ v_{y,i} \ \alpha_{x,i} \ \alpha_{y,i}]^T \quad (7)$$

$$z_i \triangleq [x_i \ y_i]^T \quad (8)$$

$$s_{i+1} = (F + \delta F_i)s_i + (G + \delta G_i)u_i \quad (9)$$

$$z_i = H s_i + v_i \quad (10)$$

$$F = \begin{bmatrix} 1 & 0 & T & 0 & 0.5T^2 & 0 \\ 0 & 1 & 0 & T & 0 & 0.5T^2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad H^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (12)$$

where u_i and v_i denote uncorrelated zero-mean white Gaussian process and measurement noises, with corresponding covariance matrices Q and R . The particular values for these covariances are determined empirically by meeting the 95% confidence whiteness test required by Mehra's method [7]. The chosen values for Q and R used in the 'SNAP&TELL' wearable system are:

$$Q = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 \end{bmatrix}, \quad R = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \quad (13)$$

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 (9) are modeled as

$$[\delta F_i \ \delta G_i] = M \Delta_i [E_f \ E_g] \quad (14)$$

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 for the case when our model is expected to change dramatically at 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], [8]. 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 indicating walking motion, and would then 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 uncertainty model (9)–(10), 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 [8]:

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

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:

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

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

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

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

We applied this robust algorithm to a typical user's fingertip trajectory and display the results in Figure 4. Note that the reduced search window is centered around the previously predicted fingertip position, and very closely overlaps the actual finger position.

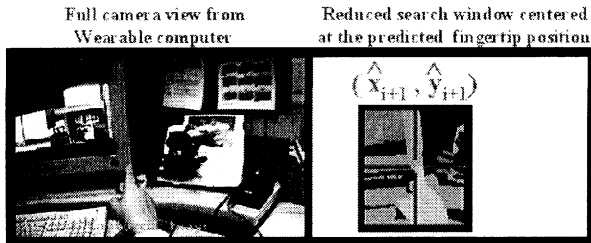


Figure 4. Successfully tracked fingertip using a robust state-space Kalman filter.

3.5. 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

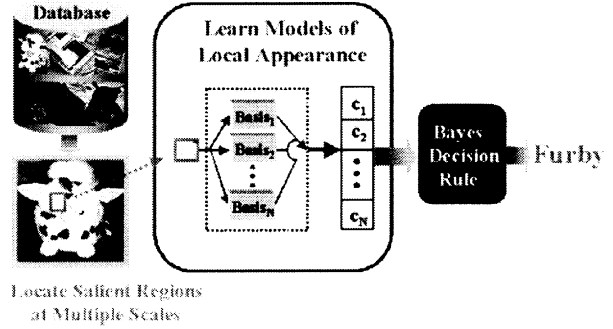


Figure 5. Object recognition framework.

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.* [9], 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, see Figure 5. 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 in the YCrCb color space, see Figure 6. 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

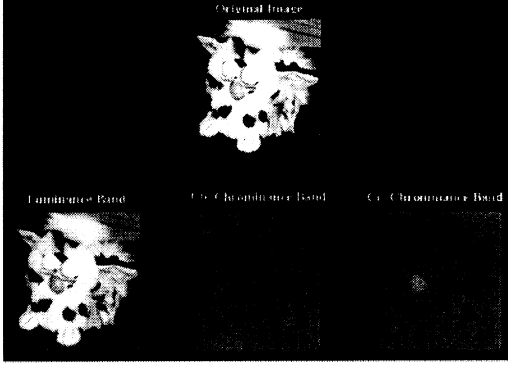


Figure 6. Detection of salient points.

the signal changes two-dimensionally. We use a technique by Tomasi and Kanade [10] 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 \mathbf{z} are assumed to be independent and Gaussian distributed with zero-mean unit variance, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The additive noise \mathbf{u} is also normally distributed with zero-mean and a diagonal covariance matrix Ψ , $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \Psi)$. Hence, the observed variables are independent given the factors, and \mathbf{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 \mathbf{x} . The factor variables \mathbf{z} model correlations between the elements of \mathbf{x} , while the \mathbf{u} variables account for independent noise in each element of \mathbf{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 $j = 1, \dots, M$), 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:

$$\begin{aligned} \tilde{\Lambda}_j^{new} &= (\sum_i h_{ij} x_i E[\tilde{z} | x_i, \omega_j])' (\sum_i h_{ij} E[\tilde{z} \tilde{z}' | x_i, \omega_j])^{-1} \\ \tilde{\Psi}_j^{new} &= \frac{1}{n} \text{diag} \left\{ \sum_i h_{ij} (x_i - \tilde{\Lambda}_j^{new} E[\tilde{z} | x_i, \omega_j]) x_i' \right\} \end{aligned}$$

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:

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

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 , let's assume we have extracted K independent local feature measurements (x_1, \dots, x_K) from an image, then we can compute the probability of each object class O_i given the image patches by

$$\begin{aligned} P(O_i | x_1, \dots, x_K) &= \frac{P_i p_i(x_1, \dots, x_K; \Theta_i)}{\sum_{n=1}^N P_n p_n(x_1, \dots, x_K; \Theta_n)} \\ &= \frac{\prod_k P_i p_i(x_k; \Theta_i)}{\prod_k \sum_{n=1}^N P_n p_n(x_k; \Theta_n)} \end{aligned}$$

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

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

Figure 7 illustrates the object recognition results obtained with the 'SNAP&TELL' wearable system.

4. Results

Figure 8 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

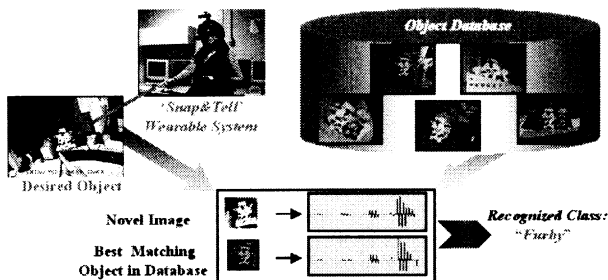


Figure 7. 'Snap&Tell' invariant object recognition.

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, where the performance of our robust Kalman tracker was estimated using a training sequence of a typical pointing finger trajectory ($\Delta W_x \geq 2\hat{x}_{max}$, $\Delta W_y \geq 2\hat{y}_{max}$). Therefore, the more accurate our tracker is in estimating the fingertip position, the smaller the size of the search window needed, and thus 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 on-line 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 small changes in illumination, view-point, and scale. We achieved 96% correct object recognition on a small test database of 5 views of 10 objects captured at different scales and perspectives, using a database of 10 views per object to train the classification models. Further testing is needed to determine how well the recognition algorithm performs as the number of objects in the database increases.

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.

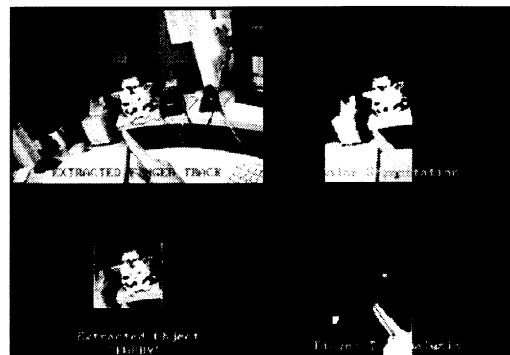


Figure 8. 'Snap&Tell' output display, showing the user's fingertip tracked in 'real-time', and the recognized object of interest.

- [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*, Friburg, 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] R. K. Mehra, "On the identification of variances and adaptive Kalman filtering", *IEEE Transactions on Automatic Control*, AC-15, pp. 175-183, 1970.
- [8] 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.
- [9] 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.
- [10] C. Tomasi and T. Kanade, "Detection and tracking of point features", Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburg, PA, April 1991.
- [11] 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.