Personal and Ubiquitous Computing
© Springer-Verlag London Limited 2005
10.1007/s00779-004-0316-5

Original Article

Browsing the environment with the SNAP&TELL wearable computer system

Trish Keaton[1], Sylvia M. Dominguez[2] and Ali H. Sayed[2]

(1) Information Sciences Laboratory, HRL Laboratories, LLC, 3011 Malibu Canyon Road, Malibu, CA 90265, USA

(2) Electrical Engineering Department, University of California Los Angeles, Los Angeles, CA 90095, USA

Trish Keaton
Email: keaton@vision.caltech.edu
Phone: +1-818-674-2277

Sylvia M. Dominguez
Email: sylvia@ee.ucla.edu
Phone: +1-818-3994560

Ali H. Sayed
Email: sayed@ee.ucla.edu
Phone: +1-310-2672142

Received: 5 April 2003 **Accepted:** 6 May 2003 **Published online:** 4 May 2005

Abstract This paper provides an overview of a multi-modal wearable computer system, SNAP&TELL. The system performs real-time gesture tracking, combined with audio-based control commands, in order to recognize objects in an environment, including outdoor landmarks. The system uses a single camera to capture images, which are then processed to perform color segmentation, fingertip shape analysis, robust tracking, and invariant object recognition, in order to quickly identify the objects encircled and SNAPped by the user's pointing gesture. In addition, 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.

Keywords Wearable computing - Finger tracking - Object recognition - Robust estimation - Gesture interfaces

1 Introduction

A recently emerging computing trend is mobile wearable computing, whereby users can rely on intelligent assistants to provide them with location-aware information. Thus, imagine a tourist using a wearable assistant while on a foreign trip. The tourist could point at a hotel, a landmark, or a restaurant and retrieve information concerning the hotel's rating, its room rates and availability, the landmark's highlights and hours of operation, the restaurant's menu/prices and opening hours, or even multilingual translations of street signs. Example applications illustrated in Fig. 1. In another example, a firefighter using a wearable assistant could receive information concerning the temperature at various locations within his surroundings, his own body temperature and blood oxygen levels, and even about a recommended direction to proceed while inside a building with low visibility conditions. In a third example, a soldier using a wearable system could point at his surroundings and snap landmark images such as buildings, mountains, warning signs, billboards, etc. The wearable assistant would then convey recommendations about friendly and enemy locations.

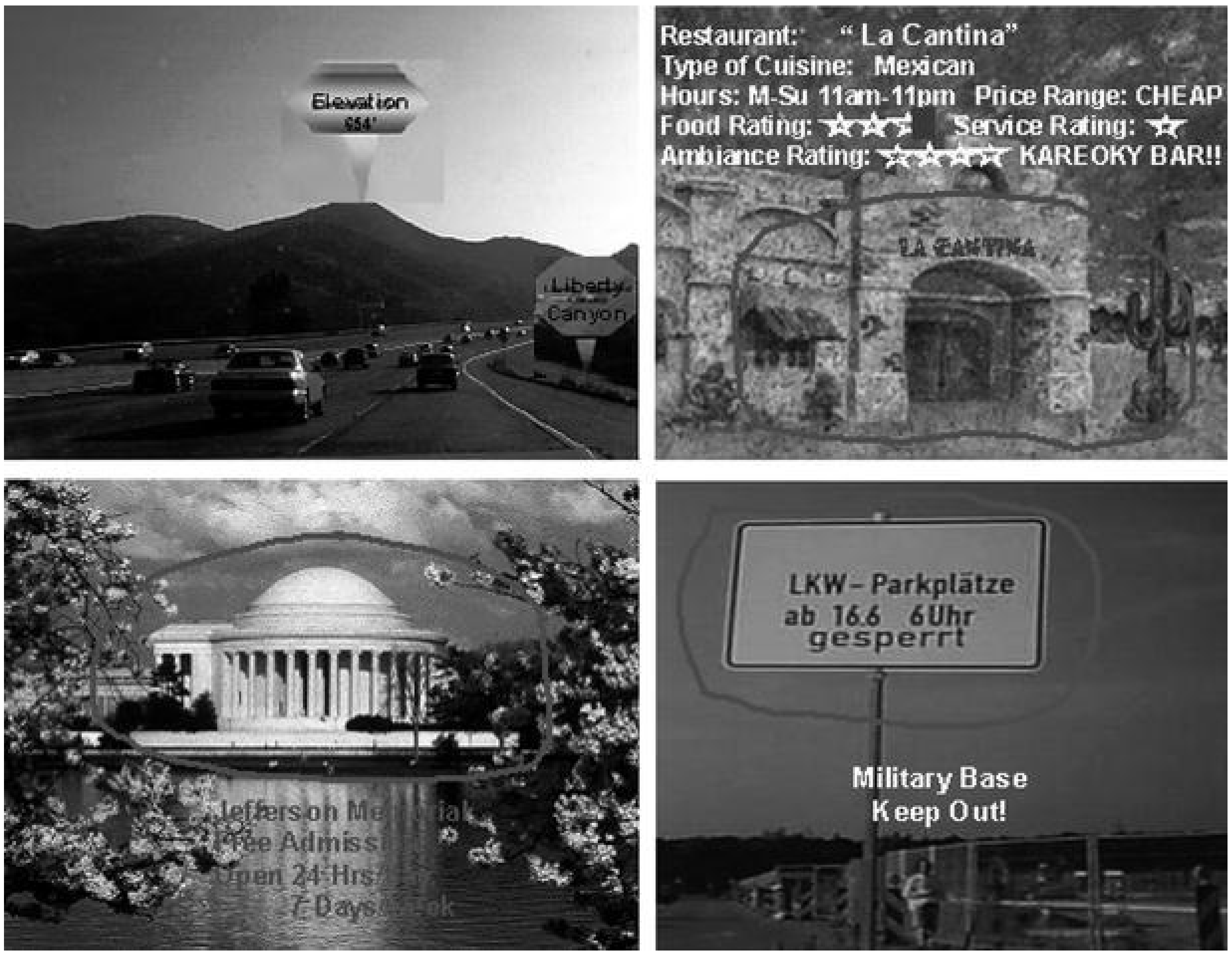

Elevation
654'
Liberty
Canyon
Restaurant: “ La Cantina”
Type of Cuisine: Mexican
Hours: M-Su 11am-11pm Price Range: CHEAP
Food Rating: ☆☆☆ Service Rating: ☆☆
Ambiance Rating: ☆☆☆☆☆ KAREOKY BAR!!
LA CANTINA
LKW - Parkplätze
ab 16.6 6Uhr
gesperrt
Military Base
Keep Out!

Fig. 1 Examples of applications for the SNAP&TELL wearable computer system

Regardless of the application, computing and sensing for wearable computers must be reliable, persistent (i.e., always remains on), easy to interact with, and easily configurable to support different needs and complexities. The success of such systems will rely, to great extent, on their ability to quickly process the sensory data captured from all sensors, and to 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 called SNAP&TELL for recognizing objects in the scene. The operation of the system is controlled by a small set of audio commands and through hand gestures. The use of pointing and hand gesturing is a natural way of interaction between a human and the machine. Visual tracking and recognition of the user's pointing and fingertip movements are important ingredients of the SNAP&TELL interface. The 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. Once the user has finished encircling an object of interest, a verbal command can be used to invoke the system's invariant object recognition module to identify the object, and to provide the user with an audio narration of all previously stored information concerning that particular object.

Presently, general-purpose computer vision algorithms tend to be complex and computationally intensive, hence, they can slow down the response of a wearable machine to a great extent. Therefore, to perform real-time acquisition and tracking, we have developed a robust state-space estimation algorithm for use with the SNAP&TELL computer. The algorithm predicts the future position of the user's pointing fingertip in a robust manner, and uses this information to reduce the search space from the full camera view to a smaller area in a dynamic and robust fashion.

The reason for using a *robust* prediction algorithm is to better control the influence of uncertain environmental conditions on the performance of the system. In a wearable computer environment, uncertainties are abound and they arise, for example, from the camera moving along with the user's head motion, the background and object moving independently of each other, the user standing still and 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. For this reason, we have incorporated data uncertainty modeling into the SNAP&TELL system's robust tracking algorithm.

In the following sections, we describe in some detail the operation of the different modules of the system, including the fingertip robust tracker.

2 SNAP&TELL system overview

At HRL Laboratories, a wearable computer system named SNAP&TELL has been designed, which is shown in Fig. 2. The system 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. 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. The system employs a robust algorithm to track the position of the tip of a user's pointing finger. This finger tracker acts as an interface to the 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.

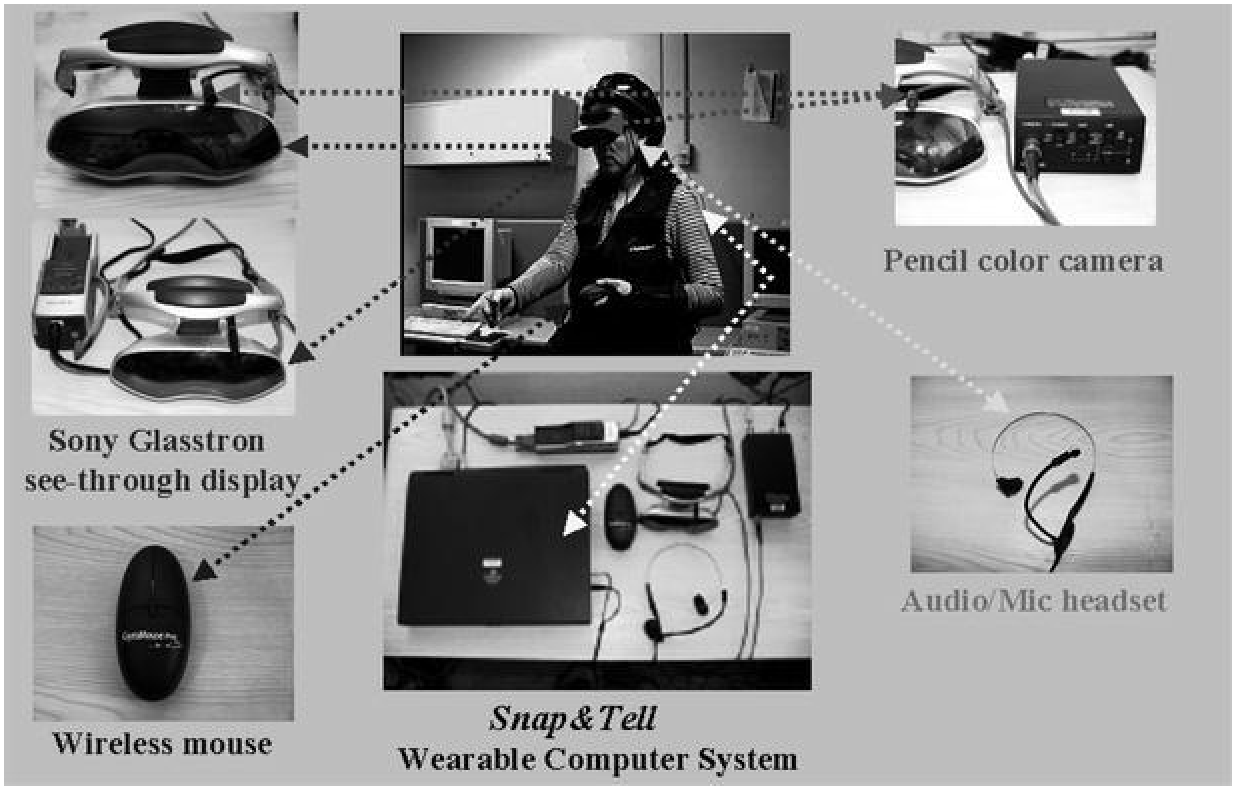

Pencil color camera
Sony Glasstron see-through display
Audio/Mic headset
Wireless mouse
Snap&Tell Wearable Computer System

Fig. 2 SNAP&TELL wearable computer system's hardware

Once the user is ready to point to an 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 through an audio narration. The system block diagram for the SNAP&TELL system is shown in Fig. 3.

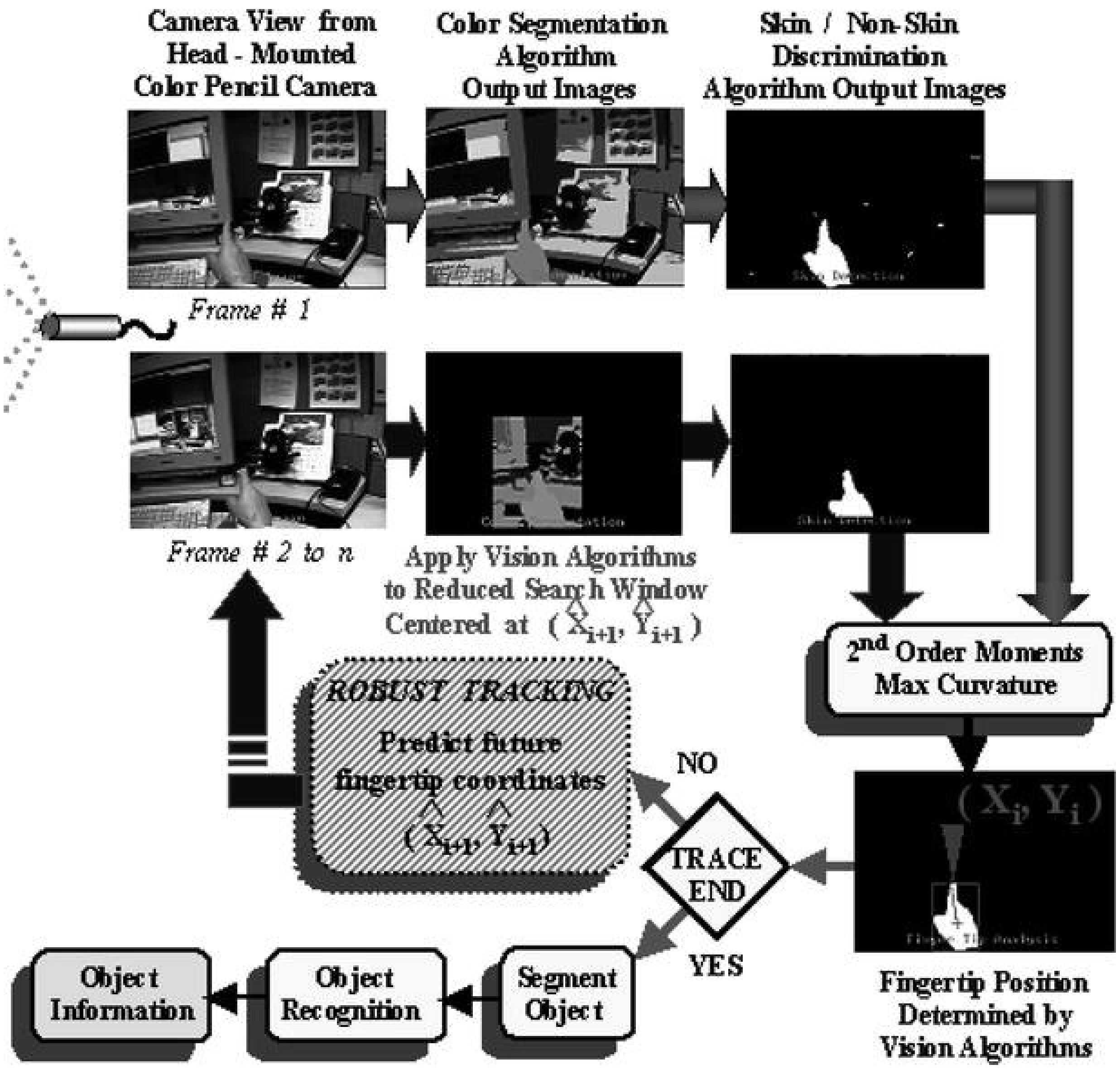

Camera View from Head - Mounted Color Pencil Camera
Color Segmentation Algorithm Output Images
Skin / Non-Skin Discrimination Algorithm Output Images
Frame # 1
Frame # 2 to n
Apply Vision Algorithms to Reduced Search Window Centered at $(\hat{X}_{i+1}, \hat{Y}_{i+1})$
2nd Order Moments Max Curvature
ROBUST TRACKING
Predict future fingertip coordinates $(\hat{X}_{i+1}, \hat{Y}_{i+1})$
NO
TRACE END
YES
(X_i, Y_i)
Fingertip Position Determined by Vision Algorithms
Segment Object
Object Recognition
Object Information

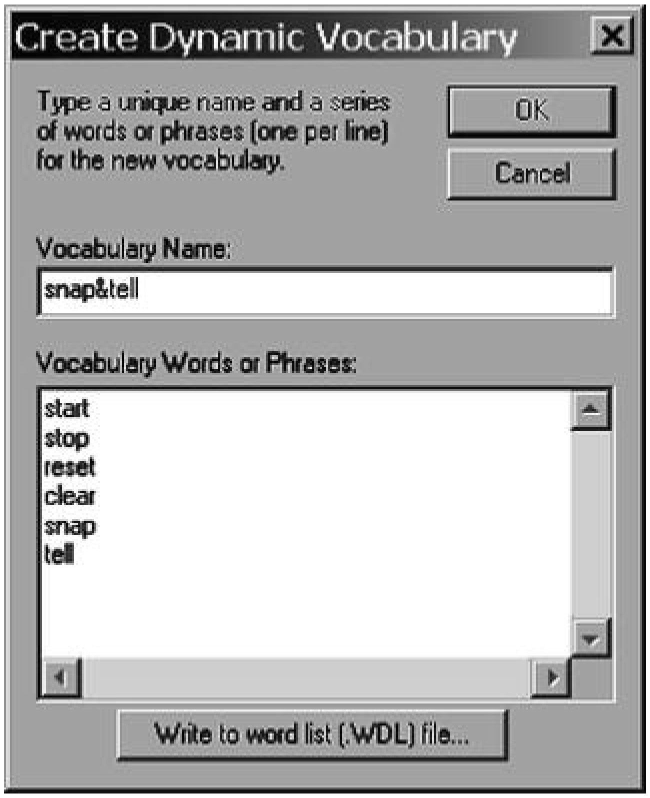
Fig. 3 Block diagram of gesture based interface for the SNAP&TELL system

This problem is particularly difficult because we need to recognize the user's hands and objects from images taken by the head-mounted cameras in real time. When the user's head moves, so does the camera, thus, introducing image jitters and dramatically significant 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 tracking model less sensitive to the random motion produced by head/camera motion and temporary occlusions. Furthermore, we use the coordinates of the robustly 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 computationally memory efficient. If, for some reason, the search window fails to display the user's hand, the system resets back to the full camera view.

2.1 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 headphones 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 through the microphone, while at the same time, it allows him to hear the information communicated back to him through audio feedback.

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 4 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 individual user, where each user is trained by having him/her say the commands in the list. Figure 5 shows the speech recognition results of a user speaking the various SNAP&TELL commands.

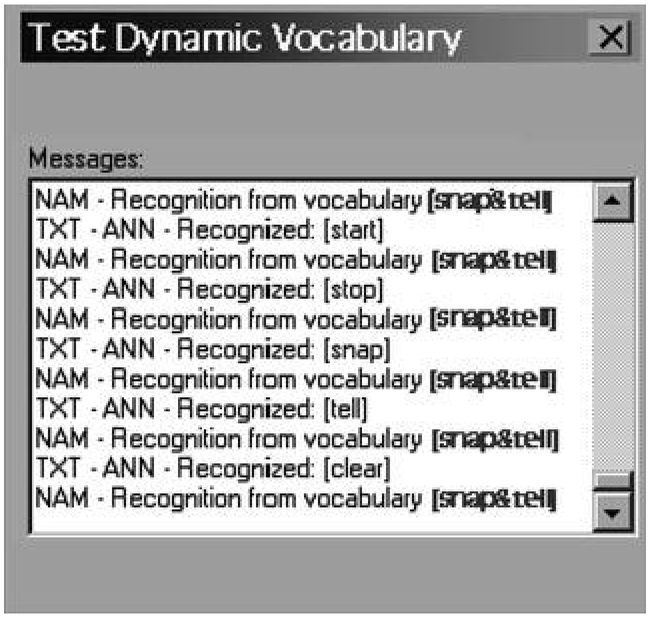


Type a unique name and a series of words or phrases (one per line) for the new vocabulary.

Vocabulary Name:
snap&tell

Vocabulary Words or Phrases:
start
stop
reset
clear
snap
tell

Fig. 4 Creating the verbal command list using the IBM's ViaVoice software



```
NAM - Recognition from vocabulary [snap&tell]
TXT - ANN - Recognized: [start]
NAM - Recognition from vocabulary [snap&tell]
TXT - ANN - Recognized: [stop]
NAM - Recognition from vocabulary [snap&tell]
TXT - ANN - Recognized: [snap]
NAM - Recognition from vocabulary [snap&tell]
TXT - ANN - Recognized: [tell]
NAM - Recognition from vocabulary [snap&tell]
TXT - ANN - Recognized: [clear]
NAM - Recognition from vocabulary [snap&tell]
```

Fig. 5 Voice recognition results of the verbal command list for the SNAP&TELL system

2.2 Video interface

The video interface encompasses two main parts. First, the user's pointing finger must be located in the image using algorithms for both skin segmentation and shape analysis. Second, the user's fingertip must be tracked during the pointing gesture.

2.2.1 Human perceptual color space

In order to locate the user's hand within the field of view of the pencil camera, we need to find all the image areas that are colored using a skin tone. This is accomplished by using a color representation for each pixel, which is similar to the way that humans perceive and store colors on their visual cortex. The human eye contains two types of photoreceptors located in the retina, the rods and the cones. The rods are more numerous, about 120 million, and are more sensitive to low light intensity than the cones (useful for night vision); however, they are not sensitive to color. The cones, which are about 6–7 million in number, provide the eye's color sensitivity. Among the cones, there are three different types of color reception, as experimental evidence suggests, which correspond to the firing of the three different types of cone nerve cells. Therefore, it follows that visible color can be mapped in terms of three numbers, called "tristimulus values," and color perception can be successfully modeled in terms of these values, which roughly correspond to the colors red, green, and blue. Thus, any color that can be produced by the primary colors red, green, and blue can be written as:

$$\text{color} = rR + gG + bB \tag{1}$$

where *r*, *g*, and *b* can be considered to be the "unit values" for the blue, green, and red cone nerve cells, and *R*, *G*, and *B* are the magnitudes, or relative firing intensities of the cones of those primaries, and they are called "tristimulus values."

The RGB color space creates a linear color representation, which, unfortunately, is not suitable for representing a particular range of colors, as opposed to representing a single color. That is, a human hand (even for a single computer user) has different shades of skin color due to the lighting, the shadows cast by objects around the user, and the three dimensionality of the hand. Therefore, in order to extract the user's hand from the camera view, we need to locate all the skin colored regions within a range of colors centered around the skin tone of the user. In order to accomplish this task, we need to encode the pixel's color using a nonlinear representation, which is more suitable for representing a range of colors.

Therefore, we use the hue, saturation, value (HSV) nonlinear color space, which represents colors along human perceptual color dimensions (RGB red cones, green cones, and blue cones) that are familiar to us all. The hue represents the property of a color that varies in passing from red to green, the saturation represents the property of a color that varies in passing from red to pink, and the value represents the brightness or lightness, which is the property of a color that varies in passing from black to white. Thus, if we are interested in checking whether a color lies in a particular range of blues, we might wish to encode the hue of the color directly, and allow the saturation to vary within a certain range. This can be easily seen in Fig. 6, which gives a visual representation of the HSV color space.

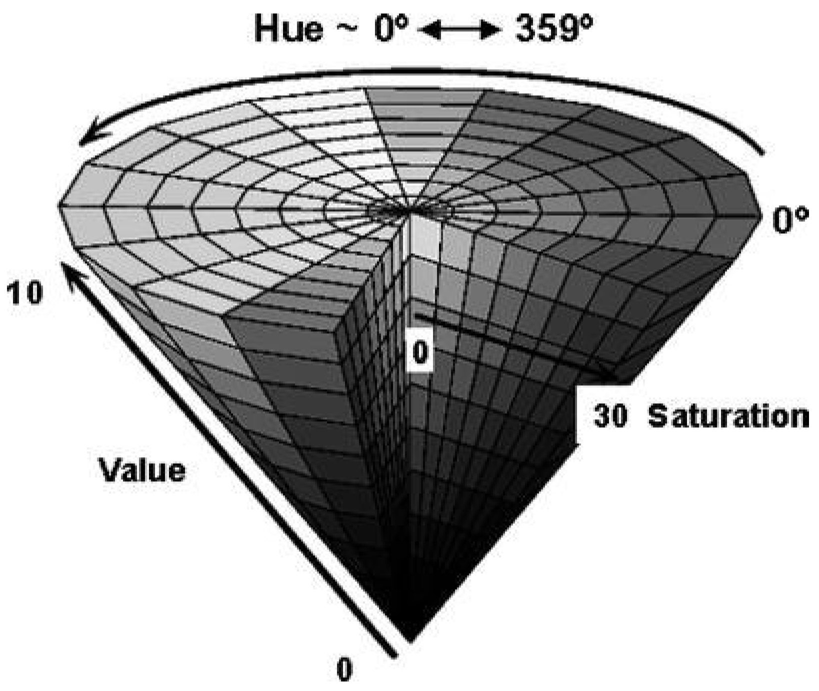


Fig. 6 HSV nonlinear color space representation

The RGB color space is transformed nonlinearly to the HSV color space according to the following transformation:

$$V = \frac{R+G+B}{3}$$
$$S = 1 - \frac{\min(R,G,B)}{V}$$
$$H = 180 - \frac{0.5(R-G)+(R-B)}{\left((R-G)^2+(R-B)(G-B)\right)^{1/2}}$$
$$H = \text{undefined} \quad \text{if } S = 0$$
$$H = 360 - H \quad \text{if } B/V > G/V$$

where the hue (*H*) indicates the color type, the value (*V*) specifies the total amount of light, and the saturation (*S*) shows how much white light is mixed with the pure color. Since the luminance and chromatic components of a color are separated in this space, it is possible to derive an effective model of color that can handle non-uniform illumination.

From this color-space, 12 perceptual color zones have been identified, which are given the following well known names: white, black, red, green, blue, purple, orange, yellow, skin or tan, pink, cyan, and gray. This differs from the Munshell color space by one color (cyan). The color zones are defined by their range of HSV values; for example, the color “skin” corresponds to the following range (hue (in degrees): 36–112, value: 4–9, saturation: 1.5–30). Since the hue and saturation ranges for the “skin” color are broad, they include some shades of adjacent colors such as yellow, tan, orange, red, and pink, in order to account for different tones of skin color, as well as for illumination effects.

2.2.2 Skin/non-skin color segmentation

We now determine the skin-like regions in the current frame by using the HSV color representation and performing 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:

$$\vec{V}(\vec{x}) = \frac{\int_{\vec{y} \in S_{\vec{x}}} p(\vec{y})(\vec{y} - \vec{x})\mathrm{d}\vec{y}}{\int_{\vec{y} \in S_{\vec{x}}} p(\vec{y})\mathrm{d}\vec{y}} \tag{2}$$

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 the reciprocal to the probability density $p(\vec{x})$, such that:

$$\vec{V}(\vec{x}) = c\frac{\nabla p(\vec{x})}{p(\vec{x})} \tag{3}$$

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 the geometric properties of the region. This region-based skin detection procedure is more robust to varying illumination conditions than pixel-based approaches.

2.2.3 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 second-order moments, and the fingertip position is determined as the point of maximum curvature along the contour of the hand.

2.2.4 Search window size

The computational effort demanded by the computer vision algorithm used to locate the user's fingertip is a function of the window search area. Standard computer vision techniques are usually hindered by their excessive memory requirements and slow computational speeds. These deficiencies preclude real-time operation.

However, 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 at the last known position of the moving object [1, 17]. 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's 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 at 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 [8], use state-space estimation techniques to center the smaller search window at the future predicted position of the user's fingertip, rather than around its current position. In this way, as the moving object's 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), the tracking of the user's hand is lost. A robust estimation algorithm that takes into account 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.

2.2.5 Robust state-space fingertip tracking

The robust finger tracker developed in [6] is based on the principles of state-space estimation with uncertain models from [13]. The robust tracker functions as follows.

First, a simplified model is adopted for the fingertip movements. The fingertip coordinate positions $\{x_{i+1}, y_{i+1}\}$ in the next video frame are modeled in terms of the present frame's fingertip pixel coordinates $\{x_i, y_i\}$ as follows:

$$x_{i+1} \approx x_i + v_{x,i}T + \alpha_{x,i}\frac{T^2}{2} \tag{4}$$

$$y_{i+1} \approx y_i + v_{y,i}T + \alpha_{y,i}\frac{T^2}{2} \tag{5}$$

$$v_{x,i+1} \approx v_{x,i} + \alpha_{x,i}T \tag{6}$$

$$v_{y,i+1} \approx v_{y,i} + \alpha_{y,i}T \tag{7}$$

where $\{\alpha_{x,i}, \alpha_{y,i}\}$ denote the accelerations along the x and y directions (measured in pixels per second^2), $\{v_{x,i}, v_{y,i}\}$ denote the speeds along these same directions during the ith 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 s/frame). The above equations motivate the following state-space model with state vector s_i and measurement vector z_i:

$$s_i \triangleq \begin{bmatrix} x_i & y_i & v_{x,i} & v_{y,i} & \alpha_{x,i} & \alpha_{y,i} \end{bmatrix}^{\mathrm{T}} \tag{8}$$

$$z_i \triangleq \begin{bmatrix} x_i & y_i \end{bmatrix}^{\mathrm{T}} \tag{9}$$

$$s_{i+1} = (F + \delta F)s_i + (G + \delta G)u_i \tag{10}$$

$$z_i = Hs_i + w_i \tag{11}$$

(12)

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad H^{\mathrm{T}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{13}$$

where u_i and w_i denote that uncorrelated zero-mean white Gaussian process and measurement noises, with corresponding covariance matrices Q and R, and where $\{\delta F_i, \delta G_i\}$ represent system uncertainties.

The values for $\{Q, R\}$ are determined empirically as follows. We assume initially large uncertainties in the x and y locations, say, of the order of three pixels, and smaller uncertainties in the displacements, say, of the order of one pixel. Then, these initial values are checked for optimality by testing the whiteness of the resulting innovations process of a Kalman filter implementation following the method of [12], and the values are adjusted until the whiteness test is passed. The chosen values for Q and R used in the SNAP&TELL wearable system, and which meet a 95% confidence whiteness test, are:

$$Q = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 \end{bmatrix} \qquad R = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \tag{14}$$

The wearable computer uncertainties are modeled by treating the given parameters $\{F, G\}$ as nominal values, and by assuming that the actual values lie within a certain set around them. Thus, the perturbations in $\{F, G\}$ in Eq. 10 are modeled as:

(15)

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

for some matrices $\{M, E_f, E_g\}$ and for an arbitrary contraction Δ_i, $\|\Delta_i\| \leq 1$. For generality, we could also allow the quantities M, E_f, and E_g to vary with time. This can be useful for cases when the model is expected to change dramatically at a particular time instant, such as when the user starts walking, is coughing, or is moving his/her head abruptly while being distracted [6]. The authors are currently investigating alternative models for modeling the uncertainties associated with the 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 switch the robust tracker's perturbation model to the "walking" mode.

Applying the time- and measurement-update form of the robust filter of [13] to the uncertainty model (Eqs. 10 and 11) yields the following equations (where Π_0 is a positive definite matrix chosen by the designer, usually a large multiple of the identity):

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

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

$$\hat{\lambda}_i = (1+\alpha)\left\|M^{\mathrm{T}}H^{\mathrm{T}}R^{-1}HM\right\|$$

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

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

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:

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

We applied this robust algorithm to a typical user's fingertip trajectory. The results are displayed in Fig. 7. Note that the reduced search window is centered at the previously predicted fingertip position, and that it very closely overlaps with the actual finger position.

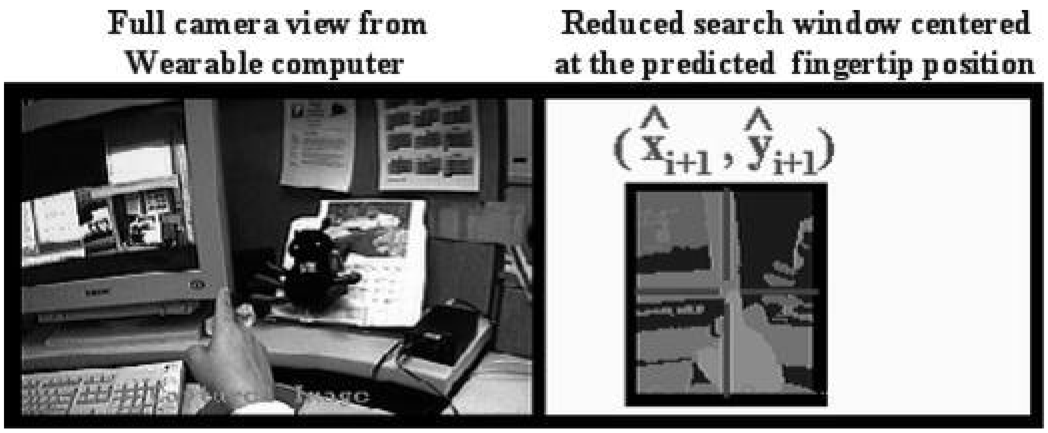

Full camera view from Wearable computer
Reduced search window centered at the predicted fingertip position
$(\hat{x}_{i+1}, \hat{y}_{i+1})$

Fig. 7 Successfully tracked fingertip using a robust state-space Kalman filter

2.3 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. The literature in [4] 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 and Kanade [14], 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 [10]. 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 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 sub-regions extracted from the training set for each object class. The training sets for each particular object class contain images of the object captured at different orientations, thus, allowing the system to be trained to recognize the object from different views. Then, by allowing a collection of subspaces to be learned, each subspace can become specialized to the variety of structures present in the data ensemble. Therefore, in order to find a probabilistic representation of the cropped image containing the object of interest, we first decompose the image into three color bands (YCrCb), which correspond to the luminance (Y), red chrominance (Cr), and blue chrominance (Cb) bands. For each band, we find the salient points, or corners in the image, by computing the following gradient matrix in a local neighborhood around each point in the image:

$$C_{\mathrm{grad}} = \begin{bmatrix} \sum Ix^2 & \sum I_x I_y \\ \sum I_y I_x & \sum Iy^2 \end{bmatrix} \tag{16}$$

Taking the gradient in the *x* direction (*Ix*) detects the vertical edges, and the double gradient (*Ix*^{2}) finds the points of maximum curvature (salient points) in vertical features of the image. After C_{grad} is found for a pixel, we compute the eigenvalues of C_{grad} and test if $\lambda_1 > \lambda_2$ is constant. If this condition is met for a particular point in the image, then that point is classified as a salient point (see Fig. 8).

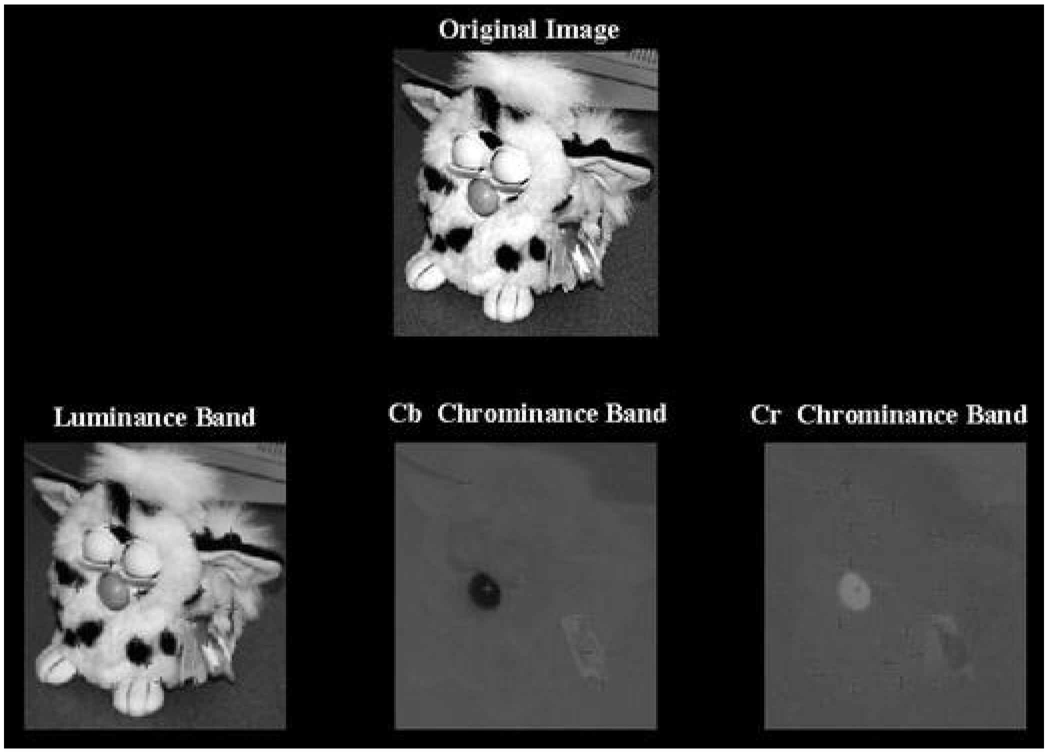


Fig. 8 Detection of salient points

Once all the salient points have been found in the three chrominance band images, for each individual salient point, we extract a set of 8×8 image patches centered at a particular salient point's position in each of the band images, and at different scale sizes for each of the bands. For three different scale sizes, this creates a set of nine salient patches for each individual salient point, as can be seen in Fig. 9. Thus, 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. This process could be repeated for all the salient points previously found; however, in order to reduce the amount of data we must process, we extract the image patches only at selected points in the image. Salient points are local features where the signal changes two-dimensionally. We use a technique described by Tomasi and Kanade [15] for finding salient features. Once all the salient patches have been found, an estimate of the probability density function, characterizing the cropped object of interest, is found by using an MFA framework and the collection of salient patches.

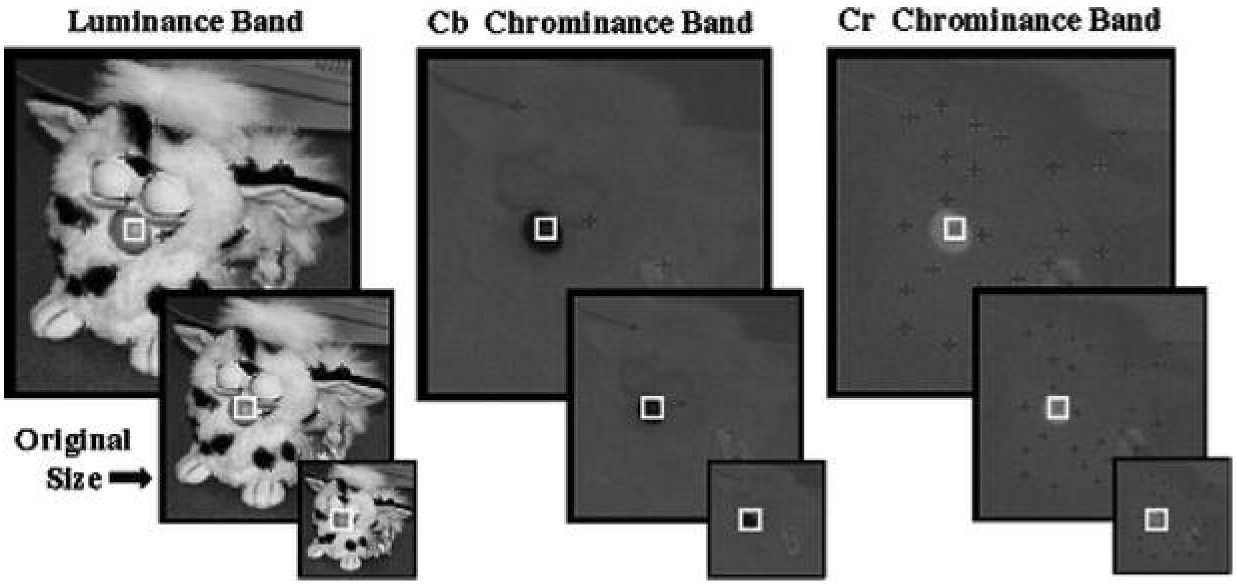


Fig. 9 Creation of salient patches, at different scales, corresponding to a single salient point

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~N(0, *I*). The additive noise *u* is also normally distributed with zero mean and a diagonal covariance matrix Ψ, *u*~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 the 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 expectation–maximization (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:

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

See [10] 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 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:

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

where Θ_i is the set of MFA model parameters for the *i*th object class, and P_{im} is the mixing proportion for the *m*th model of 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 a 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 that 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:

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

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 10 illustrates the object recognition framework for the SNAP&TELL wearable system.

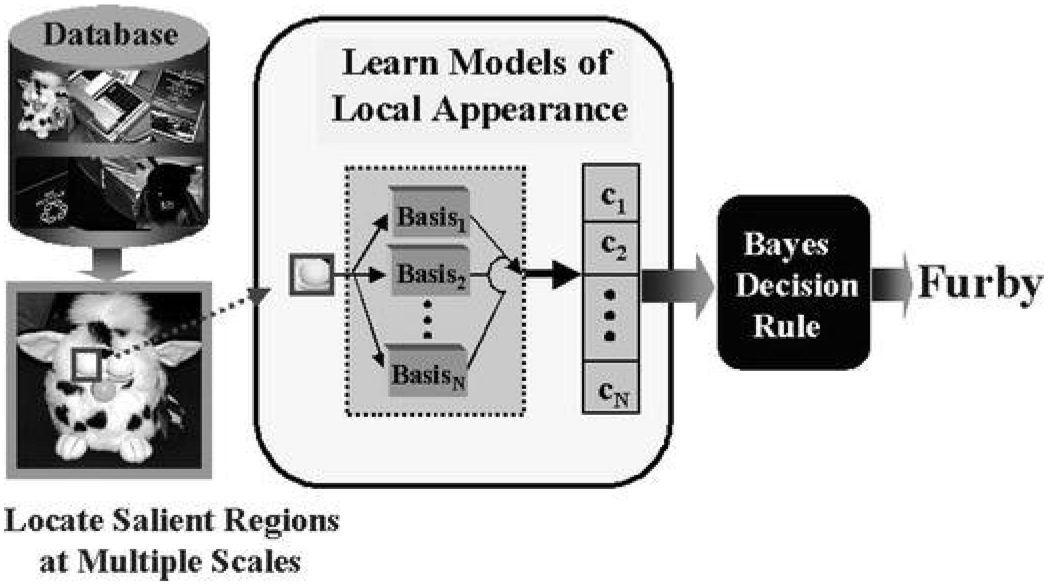


Fig. 10 Object recognition framework

3 Experimental results

Figure 11 illustrates the fingertip tracking results obtained using a 1.2 GHz laptop computer over a sequence of frames, where the last frame 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).

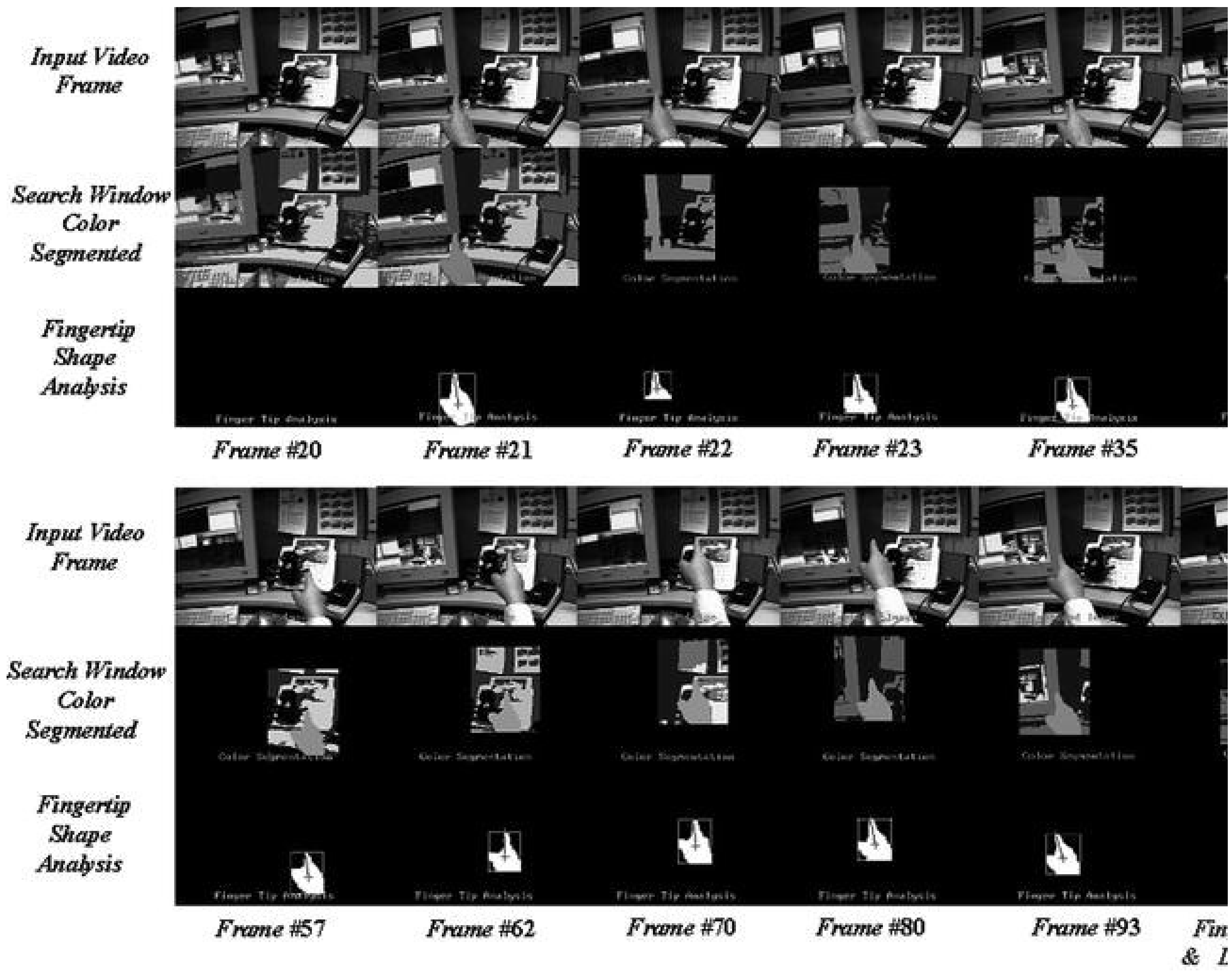


Fig. 11 Sample frames from a real-time fingertip tracking sequence using our robust tracker

Furthermore, 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 of the robust tracker in the *x* and *y* directions $(\Delta Wx \geqslant 2\bar{x}_{\max},\ \Delta Wy \geqslant \bar{y}_{\max})$, where the performance of this tracker was

estimated using a training sequence of a typical pointing finger trajectory. Therefore, the more accurate the 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. Experimental and empirical evidence suggests that possible choices for the preliminary robust tracker's matrices $\{M, E_f, E_g\}$ are:

$$E_f = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad E_g = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

$$M = \begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}^{\mathrm{T}} \quad (18)$$

These choices account for possible violations of the constant speed assumption and the acceleration instabilities present in a wearable system. The performance of the robust fingertip tracker has been compared to a tracker that relies on a plain Kalman filter for tracking a typical fingertip trajectory of a user encircling an object of interest. The mean-square-error (MSE) results for both filters are shown in Fig. 12 for the estimation error of the *x* and *y* pixel coordinates, and the estimation error in the Δx and Δy displacements. The robust algorithm, using the perturbation models (Eqs. 17 and 18), shows smaller magnitudes of the MSE for all variables, leading to an average improvement of 10% over the performance of the Kalman filter. 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.

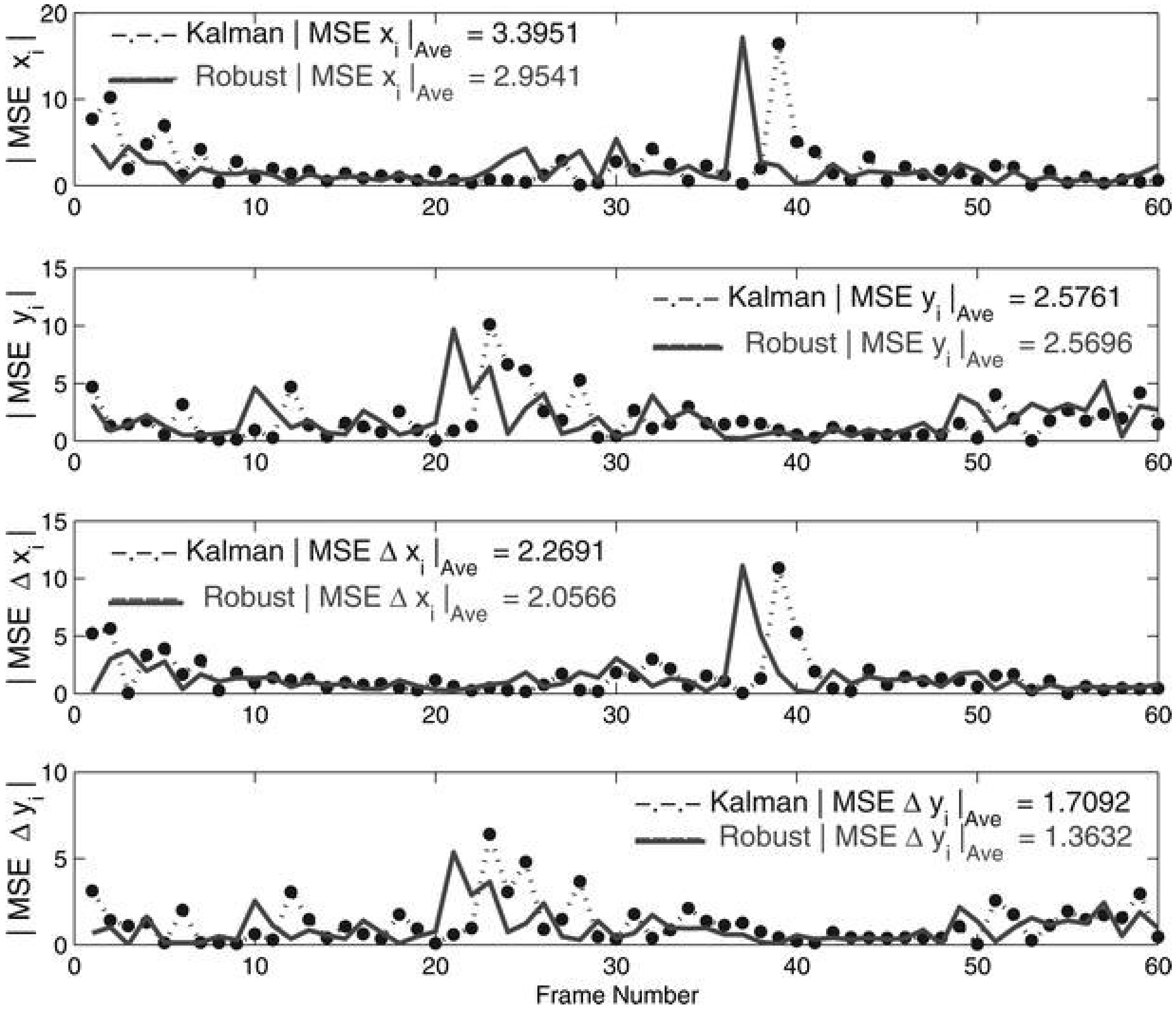


Fig. 12 Comparison of the fingertip estimation errors between the Kalman filter and the robust tracker

using the perturbation models (Eqs. 17 and 18) with α=0.5

Finally, our object recognition approach has been found to be robust to small changes in illumination, viewpoint, 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. Figure 13 illustrates the object recognition results obtained with the SNAP&TELL wearable system.

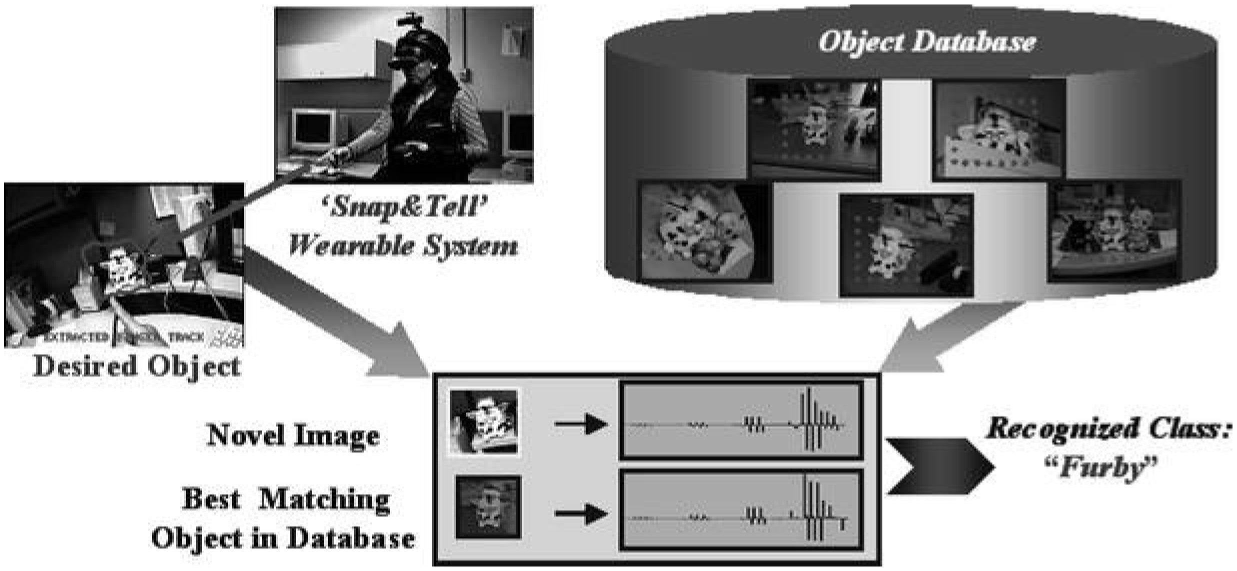


Fig. 13 SNAP&TELL invariant object recognition

4 Future work

The current performance results are encouraging and merit future exploration. The authors are investigating learning methods to develop more sophisticated models for uncertainties associated with the user's head motion, walking, and changes in lighting conditions, as well as more elaborate state-space models to account for additional information, such as depth information, hand size, and skin tone.

Acknowledgements The work of S.M. Dominguez and A.H. Sayed were partially supported by NSF grants ECS-9820765 and CCR-0208573.

References

1. Brown T, Thomas RC (2000) Finger tracking for the digital desk. In: Proceedings of the 1st Australasian user interface conference (AUIC 2000), Canberra, Australia, January 2000, pp 11–16

2. Comaniciu D, Meer P (1997) Robust analysis of feature space: color image segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR’97), San Juan, Puerto Rico, June 1997, pp 750–755

3. Crowley JL, Coutaz J (1995) Vision for man–machine interaction. In: Proceedings of the IFIP working conference on engineering for human–computer interaction (EHCI’95), Grand Targhee, Wyoming, August 1995

4. de Verdiere VC, Crowley JL (1998) Visual recognition using local appearance. In: Proceedings of the 5th European conference on computer vision (ECCV’98), Frieburg, Germany, June 1998

5. Dominguez SM, Keaton T, Sayed AH (2001) Comparison of robust estimation and Kalman filtering applied to fingertip tracking in human–machine interfaces. In: Proceedings of the 35th Asilomar conference on signal, systems, and computers, pp 342–346, Pacific Grove, California, November 2001

6. Dominguez SM, Keaton T, Sayed AH (2001) Robust finger tracking for wearable computer interfacing. In: Proceedings of the workshop on perceptual/perceptive user interfaces (PUI 2001), Orlando, Florida, November 2001

7. Ghahramani Z, Hinton GE (1996) The EM algorithm for mixtures of factor analyzers. Technical report CRG-TR-96–1, University of Toronto, Ontario, Canada

8. Jennings C (1999) Robust finger tracking with multiple cameras. In: Proceedings of the IEEE international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems (RATFG-RTS’99), Corfu, Greece, September 1999, pp 152–160

9. Kailath T, Sayed AH, Hassibi B (2000) Linear estimation. Prentice Hall, Upper Saddle River, New Jersey

10. Keaton T, Goodman R (1999) A compression framework for content analysis. In: Proceedings of the IEEE workshop on content-based access of image and video libraries (CVAIVL’99), Fort Collins, Colorado, June 1999, pp 68–73

11. Imagawa K, Shan L, Igi S (1998) Color-based hands tracking system for sign language recognition. In: Proceedings of the 3rd IEEE international conference on automatic face and gesture recognition (FG’98), Nara, Japan, April 1998, pp 462–467

12. Mehra RK (1970) On the identification of variances and adaptive Kalman filtering. IEEE Trans Automat Contr 15:175–183

13. Sayed AH (2001) A framework for state-space estimation with uncertain models. IEEE Trans Automat Contr 46(7):998–1013

14. Schneiderman H, Kanade T (1998) Probabilistic modeling of local appearance and spatial relationships for object recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'98), Santa Barbara, California, June 1998, pp 45–51

15. Tomasi C, Kanade T (1991) Detection and tracking of point features. Technical report CMU-CS-91–132, Carnegie Mellon University, Pittsburg, Pennsylvania

16. Wu A, Shah M, da Vitoria Lobo N (2000) A virtual 3D blackboard: 3D finger tracking using a single camera. In: Proceedings of the 4th IEEE international conference on automatic face and gesture recognition (FG 2000), Grenoble, France, March 2000, pp 536–543

17. Yang J, Yang W, Denecke M, Waibel A (1999) Smart sight: a tourist assistant system. In: Proceedings of the 3rd international symposium on wearable computers (ISWC'99), San Francisco, California, October 1999, pp 73–78

18. Zhu X, Yang J, Waibel A (2000) Segmenting hands of arbitrary color. In: Proceedings of the 4th IEEE international conference on automatic face and gesture recognition (FG 2000), Grenoble, France, March 2000, pp 446–453