A Robust Finger Tracking Method for Multimodal Wearable Computer Interfacing

Sylvia M. Dominguez, Trish Keaton, *Member, IEEE*, and Ali H. Sayed, *Fellow, IEEE*

***Abstract*—Mobile wearable computers are intended to provide users with real-time access to information in a natural and unobtrusive manner. Computing and sensing in these devices must be reliable, easy to interact with, transparent, and configured to support different needs and complexities. This paper presents a vision-based robust finger tracking algorithm combined with audio-based control commands that is integrated into a multimodal unobtrusive user interface, wherein the interface may be used to segment out objects of interest in the environment by encircling them with the user's pointing fingertip. In order to quickly extract the objects encircled by the user from a complex scene, this unobtrusive interface uses a single head-mounted camera to capture color images, which are then processed using algorithms to perform: color segmentation, fingertip shape analysis, perturbation model learning, and robust fingertip tracking. This interface is designed to be robust to changes in the environment and user's movements by incorporating a state-space estimation with uncertain models algorithm, which attempts to control the influence of uncertain environment conditions on the system's fingertip tracking performance by adapting the tracking model to compensate for the uncertainties inherent in the data collected with a wearable computer.**

***Index Terms*—Finger tracking, genetic algorithm, human–machine interface, Kalman filter, robust filtering, state-space model, wearable computing.**

Fig. 1. Examples of applications of a wearable computer assistant with a pointing interface.

I. Introduction

A RECENT computing technology trend is mobile wearable computing whereby intelligent assistants provide users with location-aware information in order to help them accomplish their tasks more efficiently. Wearable computers can provide multiple public safety and service applications such as virtual medical assistance, fire-fighting assistance, tourist assistance [1], virtual mouse abilities [2], and a virtual three-dimensional (3–D) blackboard [3]. For instance, a paramedic using a wearable system will be able to receive assistance from a virtual medical aid by encircling and extracting images of specific injuries on a victim, and getting feedback on the most suitable treatment to apply in those particular situations. Wearable computers are also useful in military scenarios. For example, a soldier using a wearable computer during a mission may be able to point at his/her surroundings and extract landmark images such as buildings, mountains, warning signs, billboards, etc. The wearable assistant could convey recommendations about where enemy troops are located; it could also convey information about which direction to proceed to and about the surrounding terrain such as a mountain's elevation or a river's depth. The wearable computer could also provide language translation of foreign signs. Furthermore, a wearable system could be equipped with face recognition capabilities, in which case it could provide users with the ability to identity personnel of interest. Fig. 1 depicts additional examples of applications of wearable computers.

Computing and sensing in a wearable computer environment must be reliable, persistent (always remains on), easy to interact with, and configured to support different needs and complexities. Therefore, a critical factor for the success of such systems is their user interface. Visual tracking and recognition of pointing and hand gestures are a natural way of interacting with a wearable system. For this reason, we have opted to rely on a real-time gesture tracking interface for segmenting objects in a scene. The interface enables the user to extract an object of interest by simply encircling the object with his or her pointing finger. In addition, the interface contains a small set of audio commands that allows the user to unobtrusively control the start and end of the object segmentation.

Manuscript received March 19, 2004; revised September 27, 2005. The work of A. H. Sayed was supported in part by the National Science Foundation under Grants ECS-9820765, CCR-0208573, and ECS-0401188. The work of S. M. Dominguez and T. Keaton was supported by HRL Laboratories, LLC. Snap&Tell™ is a trademark of HRL Laboratories, LLC. The associate editor coordinating the review of this manuscript and aproving it for publication was Prof. Tsuhan Chen.

S. M. Dominguez and A. H. Sayed are with the Electrical Engineering Department, University of California Los Angeles, Los Angeles, CA 90095 USA (e-mail: sylvia@summavision.com; sayed@ee.ucla.edu).

T. Keaton is with the Information Science Laboratory, HRL Laboratories, LLC, Malibu, CA 90265 USA (e-mail: trish.keaton@gmail.com).

Color versions of Figs. 1, 2, 4, 5, 7–10, and 15–18 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2006.879872

The multimodal interface uses several computer vision algorithms to extract color-based segmentation and shape information from the machine's camera view in order to identify the user's hand and fingertip position. These algorithms, in their generic forms, are complex and computationally intensive, and they tend to slow down the response of the machine. In order to perform real-time acquisition and tracking, we have opted to use a robust state-space estimation algorithm that predicts the future position of the user's pointing fingertip in a robust manner. Then, the system uses the predicted coordinates to center a smaller search window during the next video frame. This scheme 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 the interface's performance, wherein the interface's performance refers to the fingertip tracking accuracy and the faster computation of the vision algorithms using the reduced search window. 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 accuracy performance of the finger trackers.

II. Multimodal Wearable Computer Interface

In order to allow a user to communicate with a wearable system in a natural and efficient manner that is easy to interact with, we have designed a multimodal robust wearable computer interface comprised of a vision-based robust finger tracking algorithm combined with simple audio-based control commands. Then, in order to test the performance of our multimodal robust interface in a wearable environment, we integrated our interface with the Snap&Tell[1] wearable computer system described in [11], [12], and in Fig. 2. In this section, we review the basic features of the multimodal interface described in [12] and provide additional details as needed.

The audio commands are input into the system through the use of IBM's ViaVoice speech recognition and text-to-speech software, where in order to make the wearable interface transparent to the user, as well as easy to interact with, the user must wear a headset consisting of head phones and a microphone to verbally communicate with the system in a natural and efficient manner. Furthermore, in order to avoid speech recognition errors generated by continuous word spotting in an open-air acoustic environment, our multimodal interface allows the user to selectively activate the speech recognition software on and off. Therefore, the IBM ViaVoice software is not continuously attempting to recognize (i.e., spot) the verbal control commands from a constant input audio stream. In addition, the user utters a single command at a time, and then the software gives audio feedback to the user on the recognized command before the system is allowed to continue the command protocol. Thus, the user is given the opportunity to utter the verbal control command repeatedly until the system recognizes the word properly.

[1] Snap&Tell is a registered trademark of HRL Laboratories, LLC, Malibu, CA.

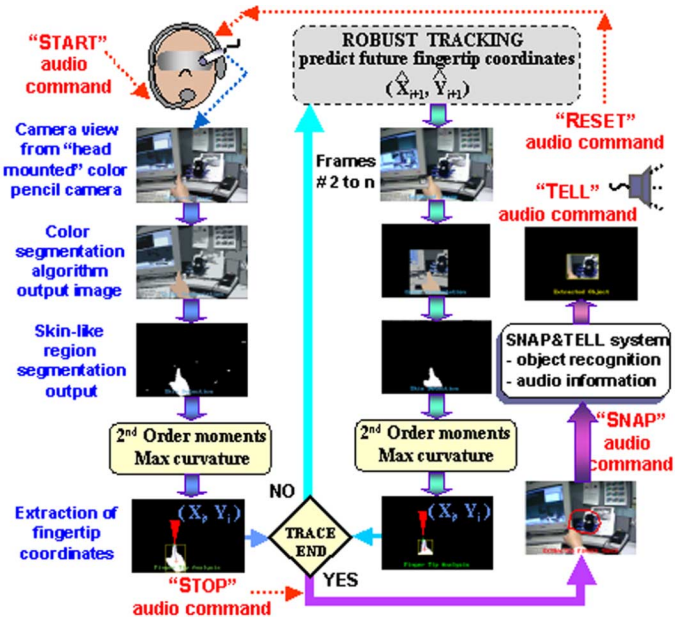

Fig. 2. Block diagram of the multimodal robust gesture-tracking interface integrated with the Snap&Tell™ wearable system.

The multimodal interface waits for the user to be ready to point to an object of interest before being activated. Once the user is ready to start selecting the desired object, he gives the verbal command "*start*", which enables the system to begin tracking the user's pointing fingertip, as indicated in Fig. 2. While tracking the user's fingertip, the interface color segments the input video frames and then applies a skin/nonskin discrimination algorithm to the color segmented images in order to detect all the likely skin toned regions. Once all the skin tone regions are found, the interface performs shape and curvature analysis to find the user's hand and to determine the user's fingertip coordinate position. The trajectory that the user's fingertip is following while encircling the object of interest is determined by the sequence of successively detected fingertip positions. At any given time during the pointing gesture, the user may delete any partial tracking points computed by using the verbal command "*clear*", which allows the user to erase defective tracking paths such as the ones created when the user moves his head abruptly causing the field of view of the camera to move away from the object of interest. Once the user is done encircling the object, the user gives the verbal command "*stop*", which signals the end of the pointing gesture and terminates the robust tracking algorithm. Then, the previously identified trajectory is used to extract the object of interest. Then, the coarsely segmented object encircled by the user is sent to the wearable system for further processing, where the system displays the segmented object for the user to visually inspect the extracted object. Then, the user is given the choice to either accept this coarsely segmented object by using the audio command "*snap*", or to "*reset*" the system in cases when the final encircled region does not coarsely segment the object properly, and "*start*" a new snapshot of the object.

Once the object has been properly segmented, the extracted object maybe sent to a wearable computer to be further analyzed, recognized, and classified. In the case of the Snap&Tell

system, the multimodal interface uses the audio command "*tell*" to recognize the coarsely segmented object and to send audio feedback to the user through the head phones, providing him with audio information pertaining to the recognized object. Ultimately, the user clears all the recognition results and sets the system into the "*wait for start command*" mode by uttering the audio command "*reset*".

Tracking the user's fingertip is a particularly difficult problem because we need to recognize the user's hands and objects from images taken from head-mounted cameras in real time. While the user is in motion, either walking or moving his head, the head-mounted camera also moves, thus introducing image jitters, and dramatic 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 integrated a state-space estimation with uncertain models algorithm [15] to the wearable interface, which attempts to control the influence of the wearable environment uncertain conditions on the system's performance, by making the tracking model less sensitive to the random motion produced by head/camera motion. In addition, in order to perform real-time acquisition and tracking, the robust interface uses the coordinates of the robust predicted fingertip position to center a smaller image search window on the next video frame for locating the user's hand. Then, 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.

For the unobtrusive video interface, we first need to locate the user's hand within the field of view of the head-mounted camera by finding all the image areas which are colored with a skin-tone. This is accomplished by using a "color segmentation" algorithm and a "skin-like regions segmentation" algorithm. Then, we need to determine which of the skin-tone areas in the image, if any, corresponds to the user's hand and pointing finger. Next, if a user's pointing finger has been found, we need to determine its fingertip coordinates, and then we need to predict the fingertip position during the next video frame by using our robust tracker. Finally, we need to reduce the input camera view for the next video frame by centering a smaller search window at the predicted fingertip coordinates provided by the robust tracker, thus relieving the computational burden of the color segmentation algorithms used to extract the user's fingertip from the rest of the scene in the subsequent video frame.

A. Color Segmentation

There are different ways in which to represent color. One of the most widely used color representations is the RGB color space, which creates a linear color representation suitable for representing an individual color, as illustrated in Fig. 3. However, to locate all the image areas which are colored in a skin-tone, we need to consider that a human hand, even for the case of a single computer user, has different shades of skin color due to the lighting, the shadows casted 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 a user. This task is accomplished

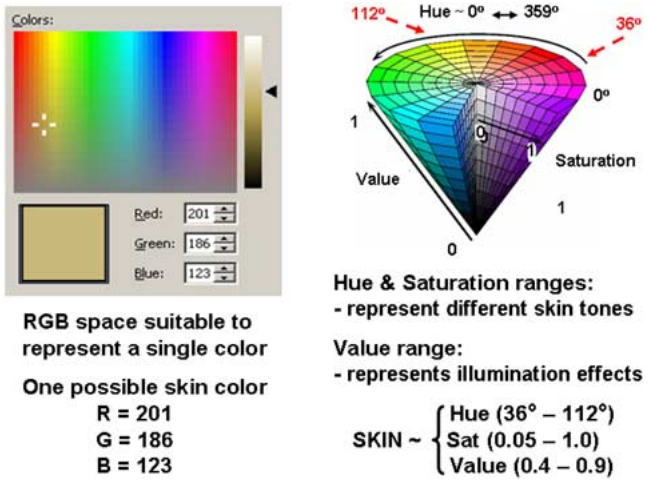

Fig. 3. Skin tone color representations.

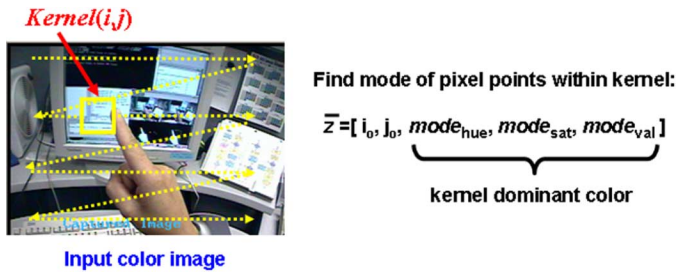

Fig. 4. Finding the mode of pixel points within a local region in an image by systematically applying a Kernel(i, j) window function, wherein the local mode represents the dominant color of the local region.

better by encoding the pixel's color in each image frame using the HSV (Hue, Saturation, Value) nonlinear color space representation, which is suitable for representing a range of colors, as shown in Fig. 3.

The *hue* (H) indicates the color type, the *value* (V) specifies the total amount of light, and the *saturation* (S) tells how much white light is mixed with the pure color. Since the luminant and chromatic components of a color are separated in this space, it is possible to derive an effective model of color that can handle nonuniform illumination. For example, the color "*Skin*" corresponds to the following range [Hue (in degrees): 36°–112°, Value: 0.4–0.9, Saturation: 0.05–1.0], wherein these broad ranges help to account for different tones of skin color. In a similar manner, the range for the *value* parameter of the *skin color* is also broad, thus accounting for variations on the skin color of the user's hands due to multiple illumination effects.

For this multimodal interface, once the current frame's color image has been transformed to the HSV color space, we perform color segmentation based on the "fast mean shift algorithm" [13]. Generally, color segmentation in a two-dimensional (2-D) image $I(i,j)$ is performed by iteratively shifting and convolving a 2-D fixed sized window or kernel function $\mathrm{Kernel}(i,j)$ with radius r_{kernel}, with the color image, to average the image data points, (pixel's color values) within the window, as seen in Fig. 4. In the case of a color image, each pixel is fully represented by a five element vector containing its location within the image "(i,j) coordinates", and its three

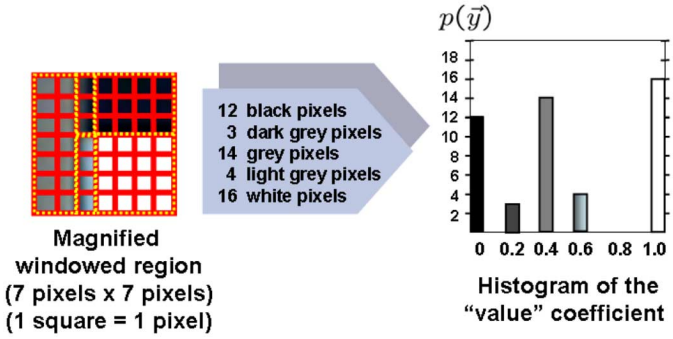

Fig. 5. Finding the probability density function (pdf) of a windowed region. In this example, the pdf (histogram) corresponding to the value coefficient of a windowed region was found. Three such pdfs are needed to fully represent the HSV color windowed region.

HSV color values (hue, sat, val). Therefore, during the first iteration, the $\mathrm{Kernel}(i,j)$ is applied to the raw color image in order to determine a five element mean-vector $\vec{z}$ characterizing the windowed region $S_{\vec{x}}$ centered at $\vec{x} \in \Re^5$, where $\vec{x}$ is a vector contains the center pixel's coordinates and center pixel's color information. The resulting mean-vector $\vec{z}$ contains the region's center pixel coordinates, the mode of the hue m_{hue}, the mode of the saturation m_{sat}, and the mode of the value m_{val} of all the pixels within the windowed color region $S_{\vec{x}}$, where $m_{\mathrm{hue}}, m_{\mathrm{sat}}$, and m_{val} represent the dominant color of the $S_{\vec{x}}$ local region. The kernel window $K(i,j)$ is shifted systematically through out the image, as seen in Fig. 4, creating multiple overlapping regions and extracting the dominant color information of each particular region.

In order to determine the dominant color on a local region $S_{\vec{x}}$ centered at $\vec{x}$, the HSV color representations of all the pixels within the windowed region are histogrammed together forming three distinct histograms corresponding to the Hue histogram, the Saturation histogram, and the Value histogram for the local region centered at $\vec{x}$. These histograms correspond to the probability density functions $p(\cdot)$ that characterize the windowed region, as illustrated in Fig. 5. Then, the mean-vector $\vec{z}$ for the region $S_{\vec{x}}$ is formed by choosing the dominant mode of each of the histograms as $m_{\mathrm{hue}}, m_{\mathrm{sat}}$, and m_{val}. For the example illustrated in Fig. 5 where the Kernel(i, j) has a radius r_{kernel} corresponding to 7 pixels, the dominant mode for the value histogram $(\mathrm{pdf}_{\mathrm{val}})$ corresponds to $m_{\mathrm{val}} = 1.0$, since there are more white pixels (16 total) within the windowed region than any other color pixels.

Next, during the second iteration, the fast mean shift algorithm [13] associates with each pixel in the image the closest local mode in the density distribution $p(\cdot)$ which characterizes the windowed region $S_{\vec{x}}$ of the joint domain. The mean shift vector is defined as the difference between the mean of the probability function on a local area and the center of this region, and it is also proportional to the gradient of the probability density $\nabla p(\vec{x})$, and reciprocal to the probability density $p(\vec{x})$ [13]. Therefore, the mean shift vector associated with a region $S_{\vec{x}}$ centered on $\vec{x}$ can be written mathematically as

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

where c is a constant and $p(\vec{y})$ is the probability density function obtained by computing the histogram of the HSV color representations of all the pixels $\vec{y}\epsilon S_{\vec{x}}$ within the windowed region $S_{\vec{x}}$ in the combined spatial-range domain, as illustrated in Fig. 5. Then, since the mean shift vector $\vec{V}(\vec{x})$ always points towards the direction of the maximum increase in the probability density, it can be used to define a path for the search window to follow, which leads to a local probability density maximum (i.e., a mode of the density or true dominant color). Therefore, instead of shifting the window kernel in a fixed systematic path across the entire image, the mean shift algorithm moves the kernel by following the path of maximum density indicated by the mean shift vector found on (1). As a result the most dominant colors on the entire image are found first, and they are used as the first set of cluster centers instead of using a predetermined number of color clusters chosen ahead of time by a user, as it is the case with other clustering algorithms. Therefore, the mean shift algorithm converges faster than other color segmentation algorithms while generating a more natural looking color segmented image.

B. Skin-Like Regions Segmentation

After segmenting the current frame into homogeneous color regions, we determine whether each region is skin-like by comparing the mode hue and mode saturation values characterizing each dominant color region, with the broad hue and saturation ranges corresponding to skin-like colors. Before a user may extract objects with the robust fingertip tracking interface, the user must initially train the skin segmentation algorithm to properly set the *skin-tone* color range to a narrower skin-tone range centered around the user's skin color under the current lighting conditions. Therefore, during a skin-tone setting session, the user must capture multiple images of the user's hand (finger pointing gestures) within the current environment. Then the skin-tone setting routine uses these images to determine the proper narrow skin-tone range for the Hue, Saturation, and Value ranges to be used with the current user.

Thus, the skin segmentation algorithm erases any dominant color region which has mode-hue and mode-value outside the *preset skin-tone* color range, while keeping any dominant color region with hue and value ranges that fall within the *preset skin-tone* color range. The broad ranges selected to characterize the "Skin" color, account for the skin tones of users of numerous nationalities, as well as for various illumination effects. An example of the results obtained by the color segmentation algorithm (a) and the skin segmentation algorithm (b) are shown in Fig. 6. As it can be seen in Fig. 6, the room lighting and the geometry of the user's hand have created a shadow on the user's hand, which resulted in two different dominant color regions within the user's hand in the color segmented image. However, this lighting effect is corrected by the broad range used by the skin segmentation algorithm, which correctly extracts the entire hand of the user, as it can be seen in the final skin segmented image in Fig. 6.

Therefore, this region-based skin detection procedure is more robust to varying illumination conditions than pixel-based approaches, since we cluster together several pixels of varying skin-toned colors during the color segmentation procedure, which results in several uniformed colored regions. Thus, if

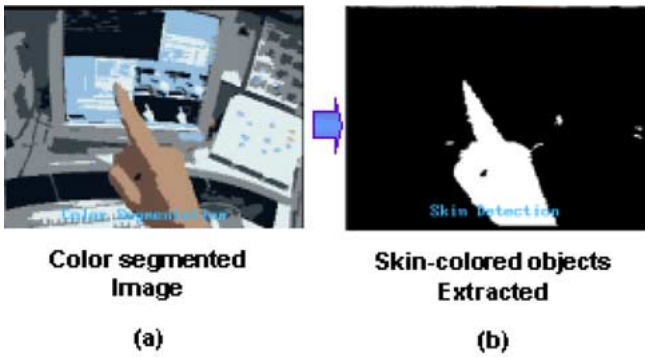

Fig. 6. (a) Example of a color segmented image generated with the fast mean shift algorithm. (b) Example of a user's hand extracted with the skin segmentation algorithm.

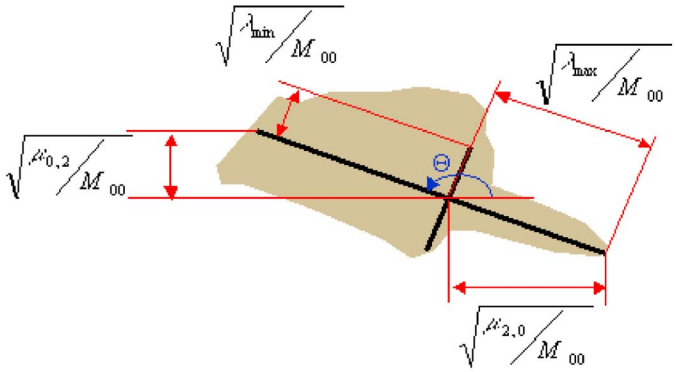

Fig. 7. Principal component analysis features derived from normalized second-order moments.

only a few pixels in the user's hand have color properties outside the skin tone, they are smoothed out by the mean shift color segmentation algorithm, and their pixelized effect is minimized. Furthermore, if several dominant skin regions are found within the user's hand (as shown in Fig. 6), the skin segmentation algorithm groups them together by using a broad skin color range. However, it should be noted that when the user's hand is placed in front of a background colored with a skin-tone color, the user's hand can not be segmented out by our current wearable interface.

III. Extraction of Fingertip Coordinates

After segmenting all the skin-like regions, we eliminate all the skin-colored objects that are outside a reasonable range of the user's hand size. This cleans up the skin segmented image by removing any small skin colored specs or white noise. Furthermore, in the case when the user hand blends with the tan colored background, the shape analysis will reject the large skin-toned blob, and it will force the system to extract new information from future video frames, until a reasonably hand-sized blob is extracted by the skin segmentation algorithm. Once a potential hand-sized blob is detected, geometric properties (e.g., elongatedness, boundary curvature) of the skin-like region are used to determine if the hand-sized blob is indeed the user's hand. Then the user's hand and finger orientation with respect to the x-axis (i.e., pointing direction) is derived using normalized 2nd order moments, and finally the fingertip position is determined as the point of maximum curvature along the contour of the finger.

The moments of order $p+q$ of a region represented by the binary image $b_{i,j}$, where i stands for the x-coordinates of the image and j stands for the y-coordinates of the image, are

$$M_{p,q} = \sum_{i=0}^{I-1}\sum_{j=0}^{J-1} i^p j^q b_{i,j} = \sum_{i,j\epsilon(\text{skin})} i^p j^q \tag{2}$$

where I and J are the spatial dimensions of the skin/nonskin binary image.

Generally, the zero order moment $M_{0,0}$ of a binary image coincides with the area of the binary image. Also, the first order moments $M_{0,1}$ and $M_{1,0}$ are related to the center of gravity or centroid $(\bar{x},\bar{y})$ of the skin-toned blob:

$$\bar{x} = M_{1,0}/M_{0,0} \quad \text{and} \quad \bar{y} = M_{0,1}/M_{0,0}. \tag{3}$$

In order to make the geometric features independent of position, moments can be calculated with respect to the centroid. The results are the so-called central moments, computed as follows [14]:

$$\mu_{0,0} = M_{0,0}, \quad \mu_{0,1} = \mu_{1,0} = 0, \quad \mu_{0,2} = M_{2,0} - \bar{x}M_{1,0}$$
$$\mu_{1,1} = M_{1,1} - \bar{x}M_{0,1}, \quad \mu_{2,0} = M_{0,2} - \bar{y}M_{0,1}.$$

The second order central moments can be used to compute the principal axes of a region by computing the principal moments or corresponding eigenvalues as follows [14]:

$$\lambda_{\max} = 1/2(\mu_{2,0}+\mu_{0,2}) + 1/2\sqrt{\mu_{2,0}^2+\mu_{0,2}^2-2\mu_{2,0}\mu_{0,2}+4\mu_{1,1}^2}$$
$$\lambda_{\min} = 1/2(\mu_{2,0}+\mu_{0,2}) - 1/2\sqrt{\mu_{2,0}^2+\mu_{0,2}^2-2\mu_{2,0}\mu_{0,2}+4\mu_{1,1}^2}.$$

Then, the orientation of the skin-toned blob, Θ, which generally corresponds to the user's finger orientation with respect to the x-axis (i.e., pointing direction), is computed from the second order central moments and the principal moments as follows:

$$\Theta = \tan^{-1}\left(\frac{\lambda_{\max}-\mu_{2,0}}{\mu_{1,1}}\right). \tag{4}$$

Fig. 7 illustrates the principal component analysis features that can be derived from the normalized second order moments. Furthermore, Fig. 8 illustrates some of the steps used by the shape analysis process to determine the user's fingertip position to be tracked by the robust estimation algorithm: Fig. 8(a) shows a reasonably hand-sized blob (determined by its M_{00}) which has been previously cleared of white noise specs; Fig. 8(b) shows the resulting output image obtained after applying the central 2^{nd} order moments analysis to the image blob in Fig. 8(a), in order to determine the user's finger orientation Θ; Fig. 8(c) shows the output image obtained after finding the point of maximum curvature on a local region along the previously found user's finger length $\sqrt{\lambda_{\max}/M_{00}}$ and orientation Θ; Fig. 8(d) shows the final shape analysis output image, which has the point of maximum curvature marked by $*$, the finger orientation marked by the direction from $+$ to $*$, and the fingertip position marked by $\star$; and Fig. 8(e) and (f) show other final shape analysis output images with different orientations.

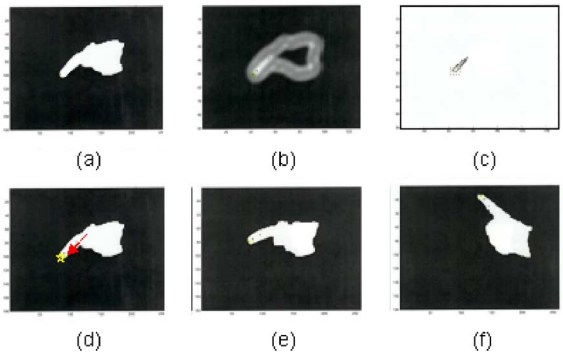

Fig. 8. (a) Reasonably hand-sized blob previously cleared of white noise specs. (b) Output image obtained after applying central 2nd order moment analysis to image blob (a), wherein the whitest regions indicate the orientation of the user's finger. (c) Output image obtained after finding the point of maximum curvature on a local region along the previously found user's finger orientation. (d)–(f) Multiple result images at different orientations obtained from the shape analysis process, wherein the user's fingertip position found is marked with a $\star$.

IV. Robust Finger Tracking Method

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.

A. Camera View Search Size

In the past, the applicability of computer vision algorithms aimed at real-time pattern recognition and object tracking has been hindered by excessive memory requirements and slow computational speeds. Some recent computer vision approaches for tracking applications speed up the computation time by reducing the image search area into a smaller window whereby the window is centered around the last known position of the moving object [1], [2]. 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 forcing the system to reset the image search area to the full view of the camera in order to recover the position of the object. Some tracking solutions have attempted an improvement by gradually varying the search window's size according to the moving object speed [2]. However, the faster the object moves, the larger the search window becomes in order to center the window around the last know position of the object, thus increasing the computation time for the vision algorithm and slowing down the system's response time. More advanced systems [4] 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. However, such systems break down if the camera view of the object abruptly changes introducing modeling uncertainties, as in the case of a head-mounted camera in a wearable system, and the tracking of the user's hand is lost. Therefore, a robust estimation algorithm, such as the one proposed in [8], [9], [15], which models the uncertainties created by the user's random ego motion (i.e., by the head-mounted camera 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.

B. State-Space Modeling for Fingertip Tracking

The presented robust finger tracker [8], [9], [12] is based on the regularized robust filtering method developed in [15]. This robust finger tracker attempts 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. In this section we describe the underlying equations [12], provide motivation and justification, and elaborate on methods for selecting the perturbation models.

Let $\{x_i, y_i\}$ denote the coordinates of the fingertip position in the current frame. Let also $\{\alpha_{x,i}, \alpha_{y,i}\}$ denote the accelerations along the x and y directions (measured in pixels per second^2), and let $\{v_{x,i}, v_{y,i}\}$ denote the speeds along these same directions during the ith frame (measured in pixels/second). Moreover, T denotes the frame capture rate while tracking the user's hand (for the Snap&Tell wearable system, this rate is currently 1/20 s/frame). Then

$$x_{i+1} \approx x_i + v_{x,i}T + \alpha_{x,i}\frac{T^2}{2} \tag{5}$$

$$y_{i+1} \approx y_i + v_{y,i}T + \alpha_{y,i}\frac{T^2}{2} \tag{6}$$

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

$$v_{y,i+1} \approx v_{y,i} + \alpha_{y,i}T. \tag{8}$$

The robust tracker attempts to predict the fingertip coordinate positions $\{x_{i+1}, y_{i+1}\}$ in the next video frame in the presence of both noise and uncertainties.

The above equations motivate the following state-space model with state vector s_i and measurement vector z_i:

$$s_{i+1} = Fs_i + Gu_i \tag{9}$$

$$z_i = Hs_i + w_i \tag{10}$$

where

$$s_i \triangleq [x_i \quad y_i \quad v_{x,i} \quad v_{y,i} \quad \alpha_{x,i} \quad \alpha_{y,i}]^T \tag{11}$$

$$z_i \triangleq [x_i \quad y_i]^T \tag{12}$$

and the model parameters are given by

$$F = \begin{bmatrix} 1 & 0 & T & 0 & 0.5T^2 & 0 \\ 0 & 1 & 0 & T & 0 & 0.5T^2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$G = I_{6\times 6} \tag{14}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{15}$$

The measurement vector z_i consists of the pixel coordinates that are provided by the vision algorithm locating the fingertip

position. These coordinates can be regarded as noisy measurements of the actual pixel coordinates $\{x_i, y_i\}$. The variables $\{u_i, w_i\}$ denote uncorrelated zero-mean white Gaussian process and measurement noises, with corresponding covariance matrices Q and R that satisfy

$$E\left(\begin{bmatrix} s_0 \\ u_i \\ w_i \end{bmatrix}\begin{bmatrix} s_0 \\ u_j \\ w_j \end{bmatrix}^T\right) = \begin{bmatrix} \Pi_0 & 0 & 0 \\ 0 & Q\delta_{ij} & 0 \\ 0 & 0 & R\delta_{ij} \end{bmatrix}. \quad (16)$$

It should be mentioned that we are using a full image frame size of 260×180, which has an x-pixel mean of 130 and a y-pixel mean of 90. Therefore, these mean values are subtracted from the actual coordinate positions before forming the above state-space model. In other words, the variables $\{x_i, y_i\}$ in the above model are zero-mean variables that have been centered by removing their means. This is needed prior to the application of the state-space estimation algorithms.

In addition, the measurement vector z_i consists of the centered pixel coordinates that are provided by the vision algorithm locating the fingertip position. These coordinates can therefore be regarded as noisy measurements of the actual pixel coordinates $\{x_i, y_i\}$. By using the assumed state-space model (11)–(15), we can then proceed to employ a variety of estimation techniques to "clean" z_i from measurement noise and to predict future movements of $\{x_i, y_i\}$.

For a Kalman filter to perform satisfactorily it is required to have knowledge of the process noise covariance matrix Q and the measurement noise covariance matrix R. These two covariances are usually estimated empirically. For the wearable computer system, the particular values for the covariance matrices Q and R were determined by following the method of [16]. A more detailed description of how we determined values for Q and R is given in [12], along with the specific values chosen for Q and R used in the Snap&Tell system.

C. Fingertip Tracker in the Presence of Uncertainties

Let us first describe a Kalman-filter based fingertip tracker. Introduce the following predicted and filtered estimates of the state vector:

$$\hat{s}_i \triangleq \text{l.l.m.s. estimate of } s_i \text{ given } \{z_0, z_1, \ldots, z_{i-1}\}$$
$$\hat{s}_{i|i} \triangleq \text{l.l.m.s. estimate of } s_i \text{ given } \{z_0, z_1, \ldots, z_{i-1}, z_i\}$$

and the corresponding error variances:

$$P_i \triangleq E(s_i - \hat{s}_i)(s_i - \hat{s}_i)^T$$
$$P_{i\,|\,i} \triangleq E(s_i - \hat{s}_{i\,|\,i})(s_i - \hat{s}_{i\,|\,i})^T.$$

In the above, the shorthand notation "l.l.m.s." stands for "linear least-mean squares".

As explained in [15], each step of the prediction form of the Kalman filter admits a deterministic interpretation as the solution to a regularized least-squares problem. Specifically, the prediction form equations of the Kalman filter can be obtained by estimating s_i along with u_i by minimizing the following estimation error cost function:

$$\min_{s_i, u_i} \Big[\|s_i - \hat{s}_{i\,|\,i}\|^2_{P^{-1}_{i\,|\,i}} + \|u_i\|^2_{Q^{-1}} + \|z_{i+1} - Hs_{i+1}\|^2_{R^{-1}} \Big]. \quad (17)$$

However, the central premise in the Kalman filtering formulation is that the underlying model parameters $\{F, G, H, R, Q\}$ are accurately known. When this assumption is violated, the performance of the filter can deteriorate and one is therefore motivated to consider robust variants; robust in the sense that they attempt to limit, in certain ways, the effect of model uncertainties on the overall filter performance. For a wearable computer system, there are several sources of uncertainties that may interfere with the accuracy of the assumed state-space model. The uncertainties can be due to the "head-mounted" camera moving along with the user's head motion, to changes in lighting conditions, to the background and object moving independently from each other, to the user standing still or randomly walking, or to the user's pointing finger abruptly changing directions at variable speeds and accelerations. All these factors changing constantly in time create different conditions of uncertainties.

One way to model uncertainties in a wearable computer system is to treat the given parameters $\{F, G\}$ as nominal values, and to assume that the actual values lie within a certain set around them. Thus consider an uncertain model of the form:

$$s_{i+1} = (F + \delta F_i)s_i + (G + \delta G_i)u_i \quad (18)$$

where the perturbations in $\{F, G\}$ are modeled as

$$[\delta F_i \quad \delta G_i] = M\Delta_i[E_f \quad E_g] \quad (19)$$

for some matrices $\{M, E_f, E_g\}$ and for an arbitrary contraction $\Delta_i, \|\Delta_i\| \leq 1$. The model (19) allows the designer to restrict the sources of uncertainties to a certain column space (defined by the matrix M), and to assign different levels of distortion by selecting the entries of $\{E_f, E_g\}$ appropriately [15]. The case of uncertainties in F only can be handled by setting $E_g = 0$. Likewise, the case of uncertainties in G only can be handled by setting $E_f = 0$. Finally, the case of accurate models is obtained by setting $M = 0, E_f = 0$, and $E_g = 0$.

Using the uncertainty model (18)–(19), we may estimate s_i along with u_i by minimizing the estimation error of the tracker over the worst-case uncertainties δF_i and δG_i by solving

$$\min_{s_i, u_i} \max_{\delta F_i, \delta G_i} \Big[\|s_i - \hat{s}_{i\,|\,i}\|^2_{P^{-1}_{i|i}} + \|u_i\|^2_{Q^{-1}} + \|z_{i+1} - Hs_{i+1}\|^2_{R^{-1}} \Big] \quad (20)$$

where $\Pi_0 > 0, R > 0, Q > 0$ are given weighting matrices. In this way, the robust tracker attempts to minimize at iteration i the estimation error at the worst possible case created by the bounded uncertainties δF_i and δG_i. This minimization–maximization procedure results in the following robust estimation

algorithm, which has a similar form to the prediction form equations of the Kalman filter [15]:

Initial conditions: $\hat{s}_0 = 0, P_0 = \Pi_0$, and $\widehat{R} = R$, where $\Pi_0 = Q$ for our wearable interface.

Step 1a: Using $\{\widehat{R}, H, P_i\}$ compute $P_{i|i}$:

$$\begin{aligned} P_{i|i} &= (P_i^{-1} + H^T \widehat{R}^{-1} H)^{-1} \\ &= P_i - P_i H^T (\widehat{R} + H P_i H^T)^{-1} H P_i \end{aligned}$$

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

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

where $\|\cdot\|$ denotes the maximum singular value of its argument.

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: Now update $\{\hat{s}_i, P_i\}$ to $\{\hat{s}_{i+1}, P_{i+1}\}$ as follows:

$$\begin{aligned} \hat{s}_{i+1} &= \widehat{F}_i \hat{x}_i + \widehat{F}_i P_i H^T R_{e,i}^{-1} e_i \\ e_i &= z_i - H \hat{s}_i \\ P_{i+1} &= F P_i F^T - \overline{K}_i \overline{R}_{e,i}^{-1} \overline{K}_i^T + \widehat{G}_i \widehat{Q}_i \widehat{G}_i^T \\ \overline{K}_i &= F P_i \overline{H}_i^T \\ \overline{R}_{e,i} &= I + \overline{H}_i P_i \overline{H}_i^T \\ \overline{H}_i^T &= \begin{bmatrix} H^T \widehat{R}_i^{-T/2} & \sqrt{\hat{\lambda}_i} E_f^T \end{bmatrix}. \end{aligned}$$

The estimates $\{\hat{s}_{i+1}, \hat{s}_{i+1\,|\,i+1}\}$, and thus the predicted fingertip coordinates for the next frame $\{\hat{x}_{i+1}, \hat{y}_{i+1}\}$, can be obtained by recursively iterating between steps 2 and 3. For the case when $\hat{\lambda}_i = 0$, steps 2 and 3 get reduced to the standard prediction form equations of the Kalman filter.

D. Selection of Perturbation Models

Experimental and empirical evidence in [8], [12] has suggested that possible choices for the robust tracker's perturbation matrices $\{M, E_f, E_g\}$, in the context of fingertip tracking with constant speed model (i.e., user standing still), are

$$E_f = [0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1] \tag{21}$$

$$E_g = [1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0] \tag{22}$$

$$M = [0.1 \quad 0.2 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T. \tag{23}$$

Using these values, the robust fingertip tracker interface has shown an average improvement in mean square error (MSE) of 10% over the prediction performance of a plain Kalman filter for tracking the fingertip trajectory of a user encircling an object of interest while standing still. In order to find the average MSE improvement of the robust filter results over the Kalman filter results, the average MSE for each filter was computed by first finding the individual MSE (pixels^2) of the estimates for the x and y pixel coordinates and the Δx and Δy pixel displacements, over the number of video frames captured from the moment the fingertip "starts" being tracked to the moment the user halts the tracking using the "stop" audio command. Then the individual MSE of the estimates for a particular filter are averaged together to obtain the average MSE for the filter, as follows

$$\begin{aligned} \text{MSE}_x &= \sum_{l=1}^{L} \frac{(x_l - \hat{x}_l)(x_l - \hat{x}_l)}{L} \\ \text{MSE}_y &= \sum_{l=1}^{L} \frac{(y_l - \hat{y}_l)(y_l - \hat{y}_l)}{L} \\ \text{MSE}_{\Delta x} &= \sum_{l=1}^{L} \frac{(\Delta x_l - \hat{\Delta x}_l)(\Delta x_l - \hat{\Delta x}_l)}{L} \\ \text{MSE}_{\Delta y} &= \sum_{l=1}^{L} \frac{(\Delta y_l - \hat{\Delta y}_l)(\Delta y_l - \hat{\Delta y}_l)}{L} \\ \text{MSE}_{\text{ave}} &= \frac{(\text{MSE}_x + \text{MSE}_y + \text{MSE}_{\Delta x} + \text{MSE}_{\Delta y})}{4} \end{aligned}$$

where l is a variable index denoting the frame numbers, L denotes the total number of frames in the tracking sequence, and the Δx and Δy pixel displacements are defined as follows:

$$\begin{aligned} \Delta x_i &= v_{x,i} T \quad (\text{pixels}) \\ \Delta y_i &= v_{y,i} T \quad (\text{pixels}). \end{aligned}$$

E. A Genetic Algorithm Procedure

Following the initial investigations in [8], [12], a more systematic method to generate the perturbation models was pursued. Specifically, a genetic algorithm was integrated into the robust wearable interface in order to automatically generate the perturbation models for the various wearable interface uncertainty scenarios.

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics [19]. They combine survival of the fittest among string structures, or vectors such as the perturbation models $\{M, E_f, E_g\}$, with a structured yet randomized information exchange to form a search algorithm. In every generation, a new offspring (set of artificial strings, vectors, or children) is created using bits and pieces of the fittest parents, and occasionally a new part is added to the mix (random mutation). Only the new offspring or children (new vectors $\{M, E_f, E_g\}$) which generate an improved performance (smaller prediction MSE) will be allowed to survive.

The first step towards using a genetic algorithm to search for the fittest perturbation model is to create a random population

of L parent vectors (chromosomes) or L sets of $\{M_i, E_{fi}, E_{gi}\}$ for $i = 1 \cdots L$, with each set of three vectors varying somewhat from the other sets. The population perturbation models $\{M, E_f, E_g\}$ for the wearable interface were chosen to be (1×6) row vectors, that are bounded to individual element values between 0 and 5, which vary by increments of 0.1. These particular uncertainty boundaries were previously determined through experimentation.

Next, the fitness or accuracy of each set of three vectors $\{M_i, E_{fi}, E_{gi}\}$ within the random population must be determined. The fitness function can be thought of as a measure of profit or goodness that we want to maximize. Selecting vectors according to their fitness values means that strings with a higher fitness value have a higher probability of contributing one or more children in the next generation. The measurement of fitness f_i (or cost function) used to judge the set of three model vectors $\{M_i, E_{fi}, E_{gi}\}$ was chosen to be a comparison between the average MSE results for x and y (fingertip coordinates) of the Kalman filter and the robust tracker using the perturbation models $\{M_i, E_{fi}, E_{gi}\}$.

Then, all the sets of three vectors are ranked according to their score. The vectors with the highest score are the vectors that produced the smallest MSE. Then the vectors with the best scores are kept and some of the lowest scoring vectors are discarded, i.e., survival of the fittest.

Once the strongest vector candidates have been selected, they are altered stochastically to produce a next generation of vectors or children. Some of the child vector sets will have higher scores than their parent vectors in the previous generation, some will have lower scores. The overall process is then repeated for subsequent generations. During each generation or iteration, only the perturbation models $\{M_i, E_{fi}, E_{gi}\}$ with the best scores are retained, and randomly altered to give yet another generation, and so on. The process is halted when the single best set of perturbation models $\{M_i, E_{fi}, E_{gi}\}$ in a generation has a score that exceeds a desired MSE criterion value or when the MSE stops improving and the fitness measure saturates.

There are three primary genetic operators that govern reproduction, and that are used to stochastically alter the parent vectors in a genetic algorithm. These primary genetic operators are replication, crossover, and mutation. A graphical representation of the three primary genetic operators used for reproduction is shown in Fig. 9, where the number of elements in a parent vector corresponds to six for the case of the perturbation models $\{M_i, E_{fi}, E_{gi}\}$, and the fitness "*fit*" corresponds to the average MSE improvement over the Kalman filter results.

During replication the most fit parent vector set of perturbation models $\{M_i, E_{fi}, E_{gi}\}$ is merely reproduced unchanged (survival of the fittest), as it can be seen in Fig. 9.

During crossover two parent vectors are mixed together to yield two new child vectors, as illustrated in Fig. 9. A split point is chosen randomly along the length of either parent, and the first part of vector "parent-1" is spliced to the last part of vector "parent-2", and vice versa, thereby yielding two new child vectors. The probability that a given pair of parent vectors will undergo crossover is given by P_{co}. It should be noted that only parent vectors of the same kind (i.e., M_i, M_j or E_{fi}, E_{fj} or E_{gi}, E_{gj}) are allowed to crossover with each other.

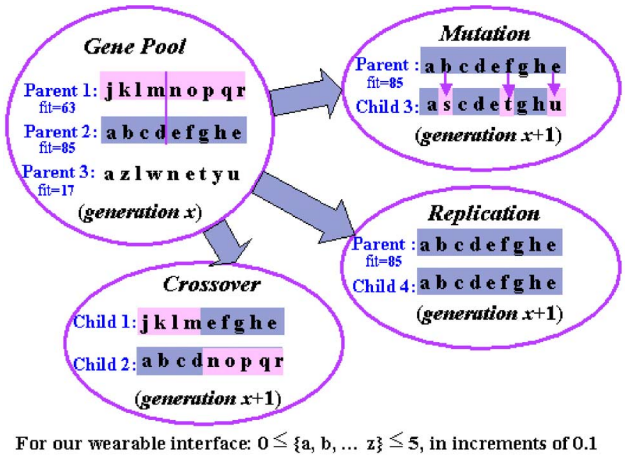

Fig. 9. Three primary genetic operators used to stochastically alter the most fit parent vectors in a genetic algorithm.

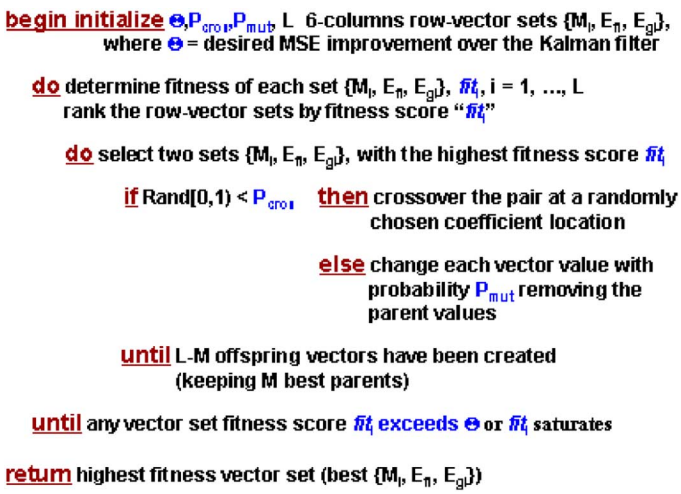

Fig. 10. Outline of the genetic algorithm used to find a suitable perturbation model $\{M_i, E_{fi}, E_{gi}\}$ for the robust wearable computer interface.

During mutation, each individual element inside a parent perturbation vector is given a small chance, P_{mut}, of being changed from its current value to a new value within its boundary range $\{0, 0.1, 0.2, \ldots, 5\}$.

An outline showing the steps performed by the genetic algorithm used to find the perturbation models $\{M_i, E_{fi}, E_{gi}\}$ for the robust wearable computer interface is given in Fig. 10.

The genetic algorithm generated the following choices for the robust tracker's perturbation matrices $\{M, E_f, E_g\}$, in the context of fingertip tracking with constant speed model (user standing still):

$$E_f = [0 \quad 0 \quad 0 \quad 0 \quad 2 \quad 4] \tag{24}$$

$$E_g = [0.1 \quad 0.1 \quad 0 \quad 0 \quad 0 \quad 0] \tag{25}$$

$$M = [0.1 \quad 0.2 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T. \tag{26}$$

The fitness measure generated by these perturbation matrices was 24% improvement over the averaged MSE (pixels^2) of the Kalman filter.

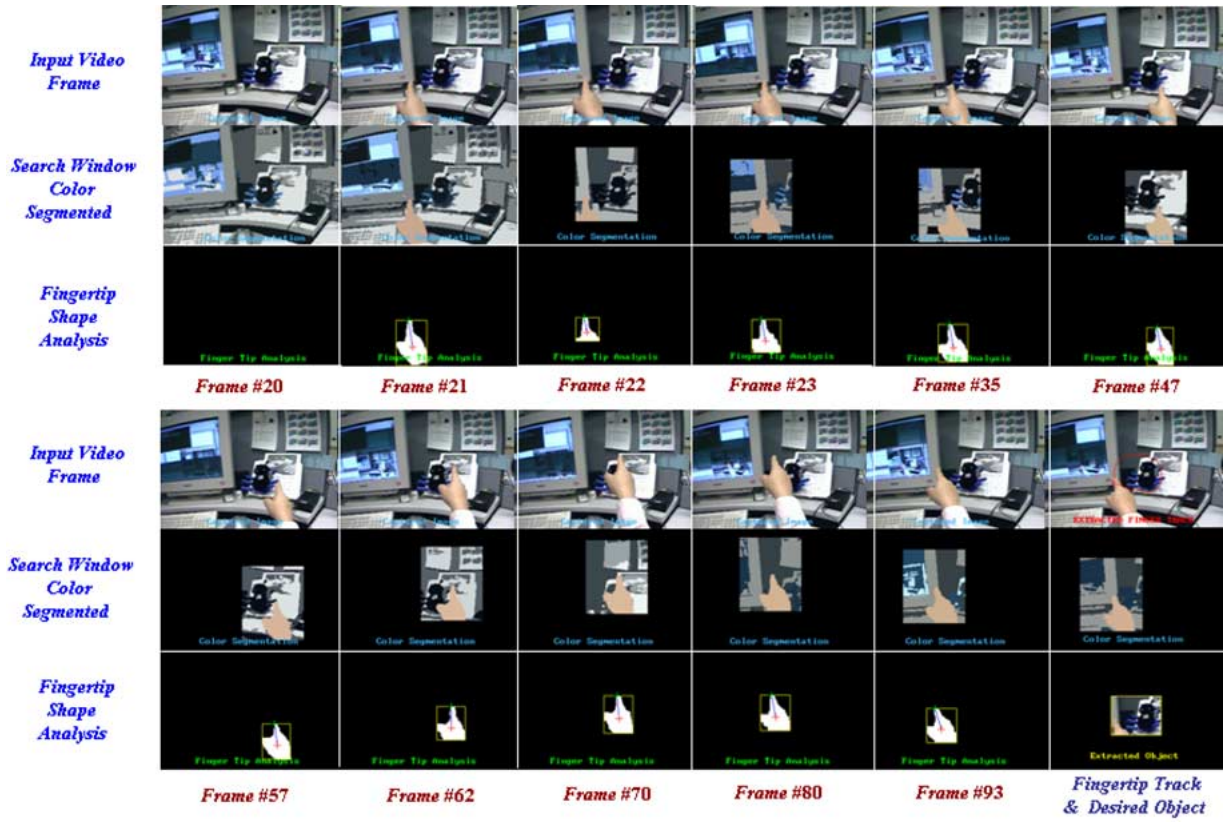

Fig. 11. Sample frames from a real-time fingertip tracking sequence generated with our robust tracker. This tracking sequence coarsely segmented a stuffed toy (blue bug) from a complex background (top of a crowded office desk).

V. Integration Results

The robust multimodal wearable computer interface using state-space estimation with uncertainty models was integrated into the Snap&Tell system in order to test the robustness of the finger tracking algorithm within a wearable environment. The fingertip tracking results obtained using the robust interface and the 1.2 GHz laptop computer of the Snap&Tell system over a sequence of frames are illustrated in Fig. 11 (see also [9] and [12]). The last frame shown in Fig. 11 illustrates 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.

In the proposed interface, the user is constantly provided with a visual-overlay displayed on the user's head-mounted display, wherein the visual-overlay consist of the continuous video frames captured by the head-mounted camera. The user, utilizes the feedback from these overlays to visually align the object of interest and the user's hand within the field of view of the head-mounted camera, in order to solve the calibration and synchronization problems generated by the user's eye, the user's hand, the head-mounted camera, and the object of interest, having different coordinate systems from one another. Therefore, this interface treats the 3-D world image captured by the head-mounted camera as a 2-D "flat" image containing the user's hand-fingertip, the object of interest, and a complex background. Then, the interface performs 2-D tracking of the user's pointing fingertip in order to separate (coarsely segment out) the object of interest from the complex background.

A. Overall Performance of Multimodal Wearable Interface

As seen in Fig. 11, the robust tracker reduces the search area into a small search window centered at the predicted 2-D fingertip coordinates found by the robust tracker, and thus the robust tracker speeds up the processing time of the vision algorithms. Note that once the robust tracker locks onto the user's fingertip position (from frame 23 until the end of the sequence in Fig. 11), the center of the reduced search window, which is centered at the previously predicted fingertip position, very closely overlaps with the actual user's fingertip position. In this simulation, the response time of our overall system was found to be 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).

A small database of ten common objects found on or near an office desk including a stapler, mouse, keyboard, phone, trash can, poster, picture, pen, and two stuffed toys (a Furby and a blue bug), were coarsely segmented by a user wearing the Snap&Tell system containing the robust finger tracking multimodal interface. Fig. 12 shows a small sample of tracking sequences captured by the wearable user with corresponding example frames illustrating each sequence's color/skin segmentation results and fingertip location. Fig. 12 also illustrates the user's fingertip trajectory extracted by the interface (marked in red color), and the final coarsely segmented object of interest ready to be sent to a wearable computer system to be recognized.

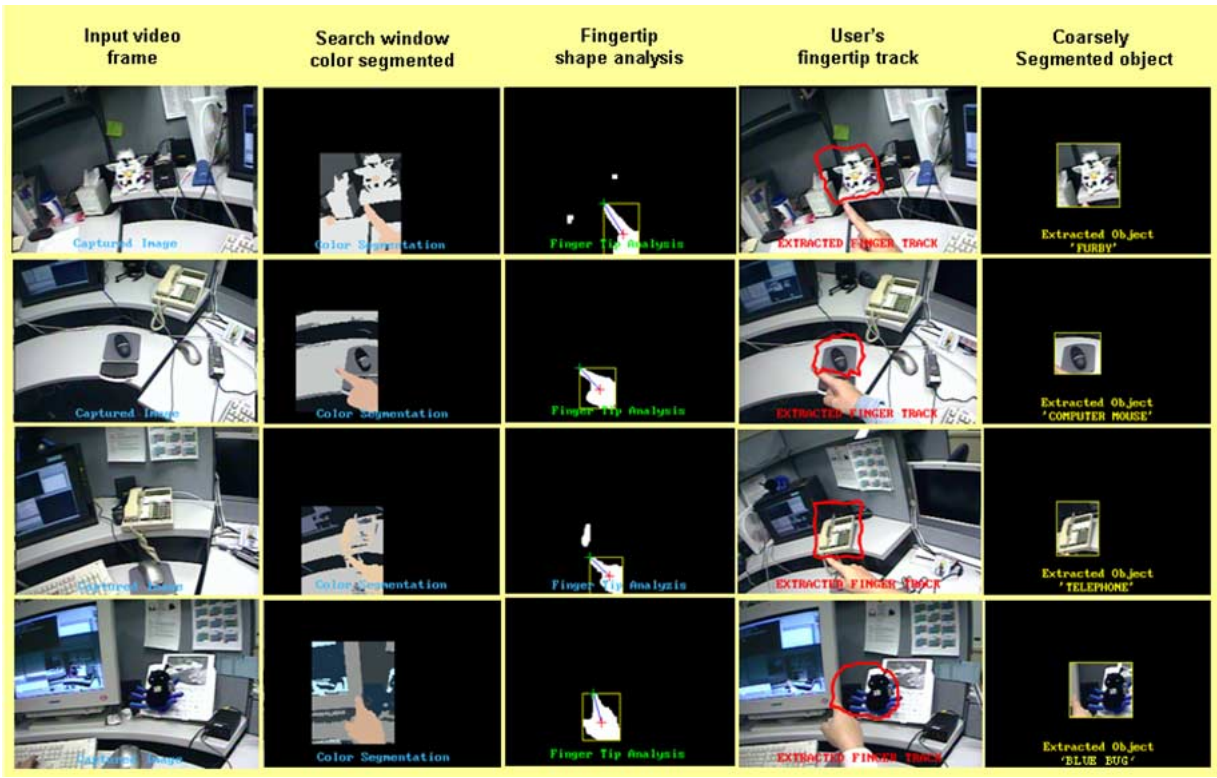

Fig. 12. Sample of four typical tracking sequences captured in "real-time" with the Snap&Tell system containing the robust finger tracking interface.

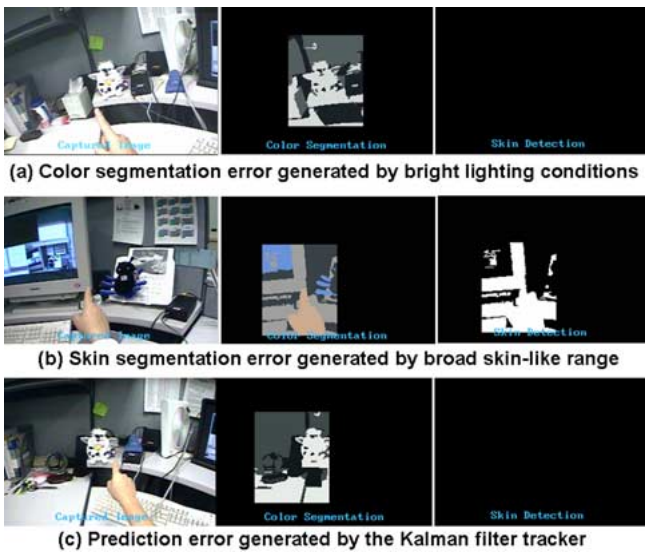

Fig. 13. Example frames illustrating a typical (a) color segmentation error, (b) skin segmentation error, and (c) prediction error, generated with the finger tracking multimodal interface.

Sample sequences	Skin - Color segmentation performance	Kalman prediction performance	Robust prediction performance	Overall robust interface performance
Normal lighting conditions for skin detection	9 errors ¦ 48 errors 99.05% correct	9 errors 99.85% correct	2 errors 99.97% correct	59 errors 99.02% correct
Dimmed lighting conditions	33 errors ¦ 52 errors 98.58% correct	12 errors 99.80% correct	3 errors 99.95% correct	88 errors 98.53% correct
Bright lighting conditions	21 errors ¦ 137 errors 97.37 % correct	11 errors 99.82% correct	2 errors 99.97% correct	160 errors 97.33% correct

Average performance of finger tracking interface over 6000 frames under three different lighting conditions

Fig. 14. Summary of results from multiple experiments performed in real-time under three different lighting conditions obtained with the robust fingertip tracking multimodal interface.

In the third sample sequence in Fig. 12 where the user encircles a beige telephone, the color segmentation routine assigns the same color to the user's hand and to parts of the telephone. In this situation, the fingertip shape analysis routine correctly extracts the users fingertip coordinates. However, when the user's hand is in contact with, or overlaps, skin-toned objects, the color segmentation routine blends (links) the skin-tone object with the user's hand, causing the fingertip tracking interface to fail. Blending of skin-tone objects with the user's hand may also occur under poor lighting conditions, such as when the current lighting changes to a dark environment or a bright environment, as shown in Fig. 13(a). Fig. 13(a) illustrates the case when the current lighting conditions on an office were changed using a second source of bright light shined on the user. Here, the color segmentation algorithm assigned the same color to the pale colored hand, the light gray desk, and the white toy, making it impossible for the skin segmentation algorithm to find the user's hand.

Extreme changes from the initial light settings also may lead to skin segmentation errors by modifying the current skin-tone of the user significantly enough, to make the user's skin-tone

fall outside the preset skin-tone range initially set by the user, or by making nonskin-tone objects fall within the user's preset skin-tone range. Another example of a skin segmentation error occurs when the preset skin-tone range of the user is too broad, as illustrated in Fig. 13(b), wherein two distinctly color segmented objects (gray computer monitor and pink user's hand) fell within the broad preset skin-tone range initially set by the user.

In the case of color segmentation or skin segmentation errors, the interface attempts to recover automatically and find the user's fingertip on the following frames without loosing the previously found fingertip track. However, in the case of poor lighting conditions or skin-tone backgrounds, the interface may not be able to locate the user's hands, forcing the user to find a better view of the object of interest away from the skin-tone background or/and poor lighting conditions. If this is not possible, the user may retrain the skin segmentation algorithm on the fly, to set the skin segmentation routine to a narrower skin-tone color range centered around the user's skin tone under the current lighting conditions by using the skin-tone setting routine.

A typical prediction tracking error generated by the Kalman filter is illustrated Fig. 13(c), where the Kalman filter predicted that the position of the user's fingertip would be to the bottom left of the object (Furby), i.e., center of the reduced search window. In this particular sequence, the user moved abruptly introducing uncertainties into the model, which caused the Kalman filter to fail, while the robust tracker, through the use of its perturbation models, tolerated the user's movement and was able to predict the user's fingertip position within the reduced search window.

In the case when the user's head or body move in small movements and his/her sight do not leave the object of interest, the robust fingertip tracker easily tolerates the uncertainties, and continues to track the user's fingertip. On the other hand, if the user moves with large or erratic movements causing the robust tracker to loose the position of the user's fingertip, then the robust interface automatically attempts to recuperate from the tracking error (lost of fingertip location) by expanding the search area to a full camera view and searching for the user's fingertip on the following frames without loosing the fingertip track information previously found. In most cases, the robust interface automatically recuperates from the majority of the tracking errors (prediction errors, color/skin segmentation errors) without the user's intervention, unless the user has moved away from the object of interest, in which case the user "clears" the track (fingertip path) previously found using audio commands, and re-issues the "start" command (once the object is back within the camera's field of view) in order to enable the system to once again begin tracking the user's pointing fingertip.

Once the user finishes encircling the object of interest, an overlay of the extracted fingertip track is superimposed on top of the current camera view of the object. At this point, the user has the choice to accept the fingertip track and object as they appear on the head-mounted display by using the audio command "snap"; or before snapping the encircled object, the user may first reposition and steady his/her head in order to ensure that the final image captured (snapped) of the object is stable, clear (not blurry), and fully encircled within the limits of the extracted fingertip track overlay; or the user may choose to reject the entire fingertip track and the coarsely segmented object by using the "clear" or "reset" audio commands.

Fig. 14 contains three image frames exemplifying three different lighting conditions (normal, dimmed, and bright) and a table of results summarizing the overall average performance of the fingertip tracking interface. This table contains the results from multiple experiments performed in real-time under three different lighting conditions using the presented fingertip tracking multimodal interface with respect to color segmentation, skin segmentation, and fingertip prediction performed by the robust tracker and by a Kalman filter tracker.

For each of the lighting conditions, ten user finger pointing sequences were captured for each of the ten objects in our data base, with an average of 60 frames/sequence. Thus, each lighting condition has: 10 sequences $\times$ 10 objects $\times$ 60 frames $\approx$ 6000 frames per lighting condition. The normal lighting conditions corresponded to an office environment with the lights turned on and with the preset skin-tone range initially trained by the user. The dimmed lighting conditions were achieved by using a dimmer on the office's light switch, wherein the lights were dimmed to approximately 70% of the intensity of the normal lighting conditions. Bright lighting conditions were achieved by adding to the normal lighting conditions a second light source shined from above directly at the user and the object of interest. Note that during the dimmed and bright lighting conditions the preset skin-tone range was kept the same as during the normal lighting conditions in order to simulate a change on lighting conditions surrounding a user in a wearable environment.

The majority of the errors in the fingertip tracking interface, under any lighting conditions, were generated by the color segmentation algorithm when the complex scene contained skin-tone objects. The number of color segmentation errors significantly increased during bright lighting conditions, as seen in Fig. 14, since the majority of the users hands became very pale and they blended with the light colored backgrounds. In addition, an error in prediction performance in Fig. 14 corresponds to the case when a prediction algorithm (i.e., Kalman or robust) failed to center the reduced search window in the vicinity of the user's hand, and the reduced search window did not contain the user's fingertip. The fingertip prediction performance of both, robust tracker and Kalman filter, exceeded 99.80% correct under all lighting conditions. Fig. 14 shows that the prediction algorithms were not affected by the change in lighting conditions, they were only affected by the user's head and body movements. Overall, the fingertip tracking interface using the robust tracking algorithm achieved a fingertip tracking performance of over 97% correct under all the lighting conditions, as seen in Fig. 14.

Of the 300 sequences tested (100 sequences per lighting condition), 73% of the tracking sequences (220 out of the 300 tested sequences) were tracked without any errors. Fifteen percent of the sequences with errors (44 out of 300) automatically recuperated from the various errors (color/skin segment and tracking) without the user's intervention, by resetting the interface to the full camera view or by the user looking back at the object of

interest after momentarily loosing sight of the object. Five percent of the sequences with errors (14 out of 300) required the user's intervention to clear the track and re-start the tracking sequence, since the users moved their head or body excessively during the captured sequence. Seven percent of the sequences with errors (22 out of 300) required the user's intervention to clear the track and retrain the skin segmentation algorithm to set a narrower skin-tone color range centered around the user's skin tone under the new current lighting conditions. Therefore, the robust interface was able to coarsely segment out the objects of interest on 88% of the sequences (264 out of 300) without the user having to clear or reset the interface.

Once the scene object or landmark of interest is isolated (coarsely segmented) using the robust finger tracking multimodal interface (as seen in Fig. 12), the coarsely segmented object may be sent to a wearable computer system to be recognized. At this point, any object recognition method may be used to classify the object and provide the user with relevant information pertaining the recognized object.

In order to test the recognizability of the coarsely segmented objects extracted with the robust finger tracking method, the robust multimodal wearable computer interface sent the coarsely segmented objects to the Snap&Tell system, which attempted to recognize the segmented object by matching local appearance descriptors extracted at salient points [12]. The Snap&Tell object recognition approach assumes that there are only a small number of significant objects at each location where the wearable system is being used. Therefore, the Snap&Tell object recognition approach was tested on our small database of ten common objects found on or near an office desk. Each of these objects were coarsely segmented during a training session by a user wearing the Snap&Tell system containing the robust finger tracking multimodal interface, where the user built the database by capturing ten snapshots of an object at varying viewpoints, and at two different scales. The object shots also exhibited slight variations in intensity due to the lighting present during the capture of each perspective view. The snapshot database of objects was then used to train a plurality of MFA classification models. The Snap&Tell object recognition approach achieved 96% correct object recognition on a test set of five views per object captured at arbitrary viewpoints and scales, wherein the test set objects corresponded to some of the objects contained in the training session but with the test set objects independently captured (coarsely segmented) at different scales and viewpoints from the objects in the training session. Comparing the object's test image to its best matching image in the training set, we find the approach to be robust to small changes in illumination present due to the perspective, pose differences of less than 40° along any of the coordinate axes of the viewing perspective, and scale differences of less than 60% deviation between the size of the object's test image and the size of its best training matching image. In practical applications, the user must capture enough training views to model the expected variation in appearance, which may be more or less than the ten views utilized in our test experiments. When the number of objects is quite large, the categories or classes of objects (e.g., faces, stuffed toys) are modeled as an "object class" rather than modeling each individual object separately, as demonstrated in Keaton *et al.* [24]. Therefore, as the number of objects increases, a two-stage recognition process may be employed to first recognize the object class associated with the input image followed by the recognition of the specific object within the class of objects. A detailed description of the object recognition algorithms and the object recognition results obtained with the Snap&Tell system can be found in [11] and [12].

B. Robust Finger Tracking Results

The presented robust tracking algorithm has been previously compared in extensive detail with many other robust filters in publication [15], which contains extensive simulations and comparisons of the robust tracking algorithm with other filters and trackers such as Kalman filters, H-infinity filters, set-value estimation methods, and guaranteed-cost designs, in addition to the theoretical motivation and derivation of the presented robust filter. In particular, this robust tracking filter does not require existence conditions, is applicable to time-variant as well as time-invariant models, and has been shown to yield a smaller error variance when compared with other robust filters. Furthermore, the proposed robust finger tracking algorithm is computationally simple to implement, it performs in "real-time" while at the same time, it limits the effect of model uncertainties created by the wearable computer user standing still or randomly walking while wearing a head mounted camera during various lighting conditions.

One critical constraint for a wearable computer interface is its ability to perform in "real-time," in a seemingly transparent manner. This particular requirement imposes limitations on the complexity of the computer algorithms, i.e., fingertip tracker, that can be pursued to create the gesture tracking interface. Statistical modeling filters, such as particle filtering, require an enormous amount of data to model the probability density function (pdf) of the object of interest, and to train the filters in order to be able to perform tracking properly. For a wearable computer system, where the user requests information on the fly by encircling objects with his fingertip, there are very few parameters or sample data available to the tracking algorithm. On the average, a user takes about 2 to 3 s to encircle an object, which corresponds to approximately 60 frames of data (usually between 50 to 75 frames per sequence). This corresponds to a system with a reduced model with limited or no training data.

The proposed robust finger tracking algorithm does not attempt to find the pdf of the data, it only models the effect of the uncertainties present on the wearable environment through the perturbation models, which contain only 18 perturbation coefficients. In addition, the use of genetic algorithms provide one convenient method to learn the perturbation models since this is a fundamentally nonlinear problem. In most instances, the learning is performed offline under different conditions, and visual cues, such as frame differencing, are used to determine the transition between models. That is, when the system detects abrupt background changes through frame differencing, the system switches the perturbation models from the "user standing still" to "the user walking" model.

Tracking Results Using Empirical Perturbation Models: The square root of the various mean-square-error (MSE) results (given in pixels) and the maximum estimation error $(\sqrt{\text{MSE}})$

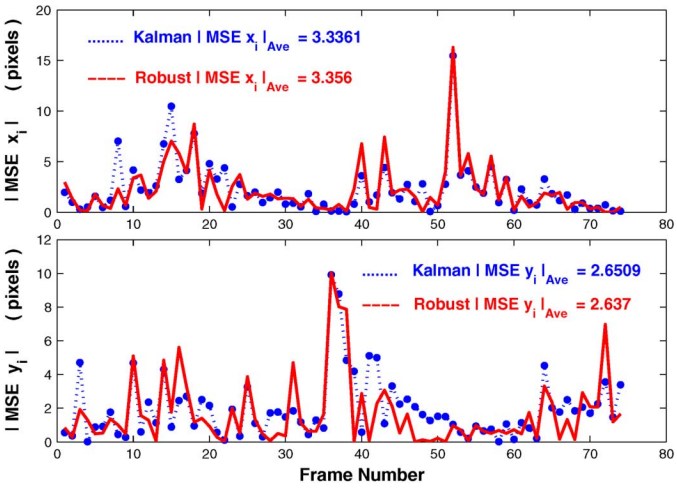

Fig. 15. Comparison of the fingertip coordinate estimation errors between the Kalman filter and the robust tracker using the empirical perturbation models defined by (21), (22), and (23) with $\alpha = 0.5$.

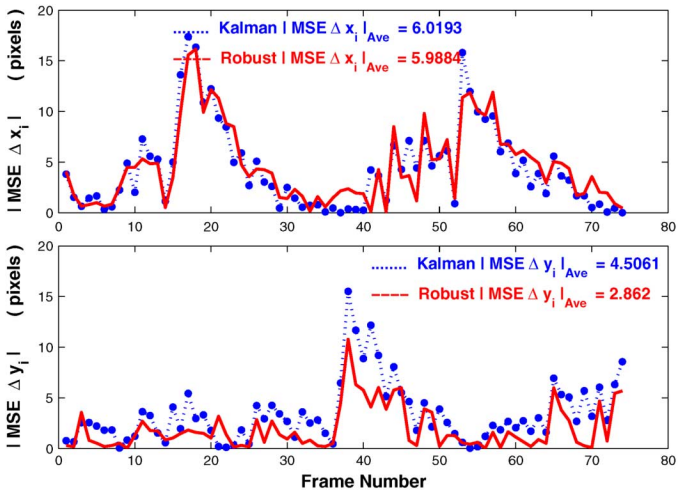

Fig. 16. Comparison of the fingertip displacement estimation errors between the Kalman filter and the robust tracker using the empirical perturbation models defined by (21), (22), and (23) with $\alpha = 0.5$.

results (given in pixels) for tracking a typical fingertip trajectory of a user encircling an object of interest while standing still by using a plain Kalman filter and the robust interface with the perturbation models (21)–(23) found empirically are shown in Fig. 15 for the estimation error of the x and y pixel coordinates, and in Fig. 16 for the estimation error in the Δx and Δy displacements. The square root of the various MSEs (given in pixels) of the Kalman filter were found to be

$$|\mathrm{MSE}_x|_K = \sqrt{\sum_{l=1}^{75} \frac{(x_l - \hat{x}_l)(x_l - \hat{x}_l)}{75}} = 3.3361$$

$$|\mathrm{MSE}_y|_K = \sqrt{\sum_{l=1}^{75} \frac{(y_l - \hat{y}_l)(y_l - \hat{y}_l)}{75}} = 2.6509$$

$$|\mathrm{MSE}_{\Delta x}|_K = \sqrt{\sum_{l=1}^{75} \frac{(\Delta x_l - \widehat{\Delta x}_l)(\Delta x_l - \widehat{\Delta x}_l)}{75}} = 6.0193$$

$$|\mathrm{MSE}_{\Delta y}|_K = \sqrt{\sum_{l=1}^{75} \frac{(\Delta y_l - \widehat{\Delta y}_l)(\Delta y_l - \widehat{\Delta y}_l)}{75}} = 4.5061$$

where l is a variable index denoting the frame number, and where there were 75 frames in this particular training tracking sequence (illustrated in Figs. 15 and 16) from the moment the "start" audio command is given until the moment the "stop" audio command is uttered by the user. Likewise, the square root of the various MSEs (given in pixels) of the robust filter using the empiric perturbation models were found to be

$$|\mathrm{MSE}_x|_R = \sqrt{\sum_{l=1}^{75} \frac{(x_l - \hat{x}_l)(x_l - \hat{x}_l)}{75}} = 3.3560$$

$$|\mathrm{MSE}_y|_R = \sqrt{\sum_{l=1}^{75} \frac{(y_l - \hat{y}_l)(y_l - \hat{y}_l)}{75}} = 2.6370$$

$$|\mathrm{MSE}_{\Delta x}|_R = \sqrt{\sum_{l=1}^{75} \frac{(\Delta x_l - \widehat{\Delta x}_l)(\Delta x_l - \widehat{\Delta x}_l)}{75}} = 5.9884$$

$$|\mathrm{MSE}_{\Delta y}|_R = \sqrt{\sum_{l=1}^{75} \frac{(\Delta y_l - \widehat{\Delta y}_l)(\Delta y_l - \widehat{\Delta y}_l)}{75}} = 2.8620$$

The average number of predicted pixels off from the user's fingertip position are approximated by the square root of the average $\mathrm{MSE}_{\mathrm{ave}}(\sqrt{\mathrm{MSE}_{\mathrm{ave}}})$ obtained from the performances of the Kalman filter and the robust tracker as follows:

$$\begin{aligned} &\text{average predicted pixels off} \propto \sqrt{\mathrm{MSE}_{\mathrm{ave}}} \\ &\quad \triangleq |\mathrm{MSE}_{\mathrm{ave}}| \\ &|\mathrm{MSE}_{\mathrm{ave}}| \\ &\quad = \frac{|\mathrm{MSE}_x| + |\mathrm{MSE}_y| + |\mathrm{MSE}_{\Delta x}| + |\mathrm{MSE}_{\Delta y}|}{4} \end{aligned}$$

Therefore, the average number of pixels off predicted by the Kalman filter was found to be

$$\begin{aligned} |\mathrm{MSE}_{\mathrm{ave}}|_K &= \frac{3.3361 + 2.6509 + 6.0193 + 4.5061}{4} \\ |\mathrm{MSE}_{\mathrm{ave}}|_K &= 4.1281 \quad \text{average pixels off} \end{aligned}$$

and the average number of pixels off from the user's fingertip position predicted by the robust filter using the empirical perturbation models was found to be

$$\begin{aligned} |\mathrm{MSE}_{\mathrm{ave}}|_R &= \frac{3.3560 + 2.6370 + 5.9884 + 2.8620}{4} \\ |\mathrm{MSE}_{\mathrm{ave}}|_R &= 3.71085 \quad \text{average pixels off} \end{aligned}$$

Then, the performance improvement of average number of pixels off from the user's fingertip position predicted by the ro-

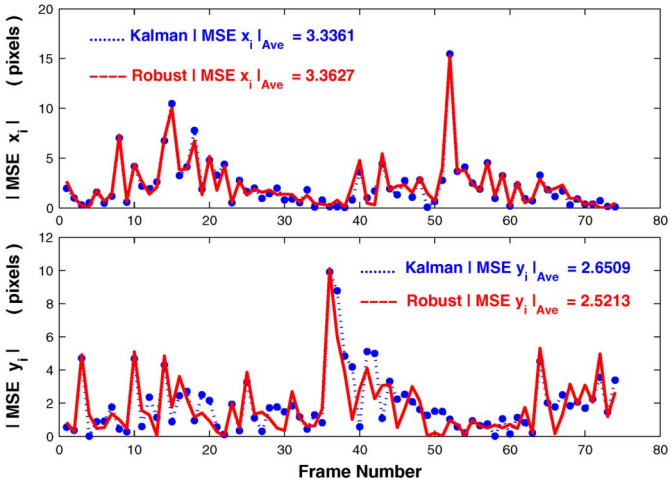

Fig. 17. Comparison of the fingertip coordinate estimation errors (pixels off) between the Kalman filter and the robust tracker using the genetic perturbation model defined by (24)–(26) with $\alpha = 0.5$.

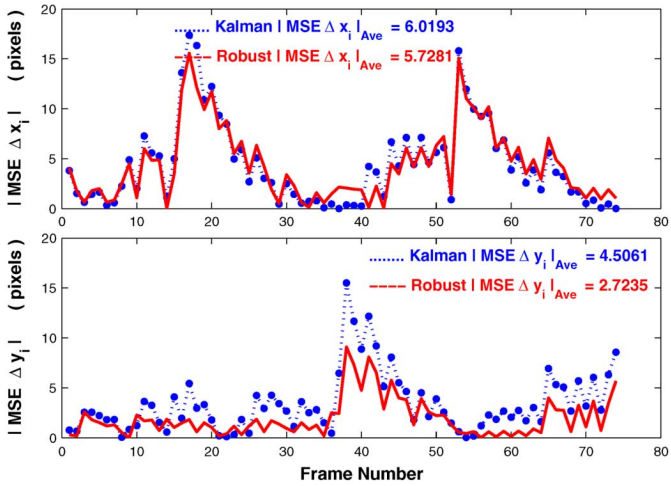

Fig. 18. Comparison of the fingertip displacement estimation errors (pixels off) between the Kalman filter and the robust tracker using the genetic perturbation model defined by (24)–(26) with $\alpha = 0.5$.

bust fingertip tracker compared with the Kalman filter was found to be

$$\frac{4.1281(K) - 3.71085(R)}{4.1281(K)} * 100\% \approx 10\%$$

while the performance improvement of the robust fingertip tracker over the averaged MSE (pixels^2) of the Kalman filter was found to be

$$\frac{17.04121(K) - 13.77041(R)}{17.04121(K)} * 100\% \approx 19\%$$

Therefore, the performance of the robust fingertip tracker interface using the empirical perturbation models (21)–(23) has shown an improvement of 10% on the average number of predicted pixels off from the actual user's fingertip position, and an improvement in average mean square error (MSE) of 19%, over the prediction performance of a plain Kalman filter tracking a typical fingertip trajectory of a user encircling an object of interest while standing still. The maximum estimation errors for the Kalman filter and the robust fingertip tracker can be found in Figs. 15 and 16. The maximum estimation errors (given in pixels) for the Kalman filter are

$$\tilde{x}_{\max} = 15.4581 \quad \text{(pixels)}, \quad \tilde{y}_{\max} = 9.9089 \quad \text{(pixels)}$$
$$\tilde{\Delta x}_{\max} = 17.3755 \quad \text{(pixels)}, \quad \tilde{\Delta y}_{\max} = 15.5054 \quad \text{(pixels)}$$

and the maximum estimation errors (given in pixels) for the robust filter using the empiric perturbation models are

$$\tilde{x}_{\max} = 16.2897 \quad \text{(pixels)}, \quad \tilde{y}_{\max} = 10.0308 \quad \text{(pixels)}$$
$$\tilde{\Delta x}_{\max} = 16.1440 \quad \text{(pixels)}, \quad \tilde{\Delta y}_{\max} = 10.7803 \quad \text{(pixels)}$$

Tracking Results Using Genetic Perturbation Models: The square root of the various MSE results (given in pixels) and the maximum estimation error results (given in pixels) for the Kalman filter and the robust finger tracker with the perturbation model defined by (24)–(26) found using a genetic algorithm, are shown in Fig. 17 for the estimation error of the x and y pixel coordinates, and in Fig. 18 for the estimation error in the Δx and Δy pixel displacements. The square root of the various MSE results (given in pixels) for the Kalman filter were found to be

$$|\text{MSE}_x| = 3.3361 \text{ (pixels)}, \quad |\text{MSE}_y| = 2.6509 \text{ (pixels)}$$
$$|\text{MSE}_{\Delta x}| = 6.0193 \text{ (pixels)}, \quad |\text{MSE}_{\Delta y}| = 4.5061 \text{ (pixels)}$$

where there were 75 frames in this typical training tracking sequence. Likewise, the square root of the various MSEs of the robust filter using the genetically generated perturbation models were found to be

$$|\text{MSE}_x| = 3.3627 \text{ (pixels)}, \quad |\text{MSE}_y| = 2.5213 \text{ (pixels)}$$
$$|\text{MSE}_{\Delta x}| = 5.7281 \text{ (pixels)}, \quad |\text{MSE}_{\Delta y}| = 2.7235 \text{ (pixels)}$$

where the average number of pixels off predicted by the Kalman filter was found to be

$$|\text{MSE}_{\text{ave}}|_K = \frac{3.3361 + 2.6509 + 6.0193 + 4.5061}{4}$$
$$|\text{MSE}_{\text{ave}}|_K = 4.1281 \quad \text{average pixels off}$$

and the average number of pixels off from the user's fingertip position predicted by the robust filter using the genetic perturbation models was found to be

$$|\text{MSE}_{\text{ave}}|_R = \frac{3.3627 + 2.5213 + 5.7281 + 2.7235}{4}$$
$$|\text{MSE}_{\text{ave}}|_R = 3.5839 \quad \text{average pixels off.}$$

Then, the performance improvement of average number of pixels off from the user's fingertip position predicted by the ro-

bust fingertip tracker using genetic perturbation models compared with the Kalman filter was found to be

$$\frac{4.1281(K) - 3.5839(R)}{4.1281(K)} * 100\% \approx 13\%$$

while the performance improvement of the robust fingertip tracker over the averaged MSE (pixels^2) of the Kalman filter was found to be

$$\frac{17.04121(K) - 12.84434(R)}{17.04121(K)} * 100\% \approx 24\%$$

Therefore, the performance of the robust fingertip tracker interface using the genetic perturbation models (24)–(26) has shown an improvement of 13% on the average number of predicted pixels off from the actual user's fingertip position, and an improvement in average MSE of 24%, over the prediction performance of a plain Kalman filter tracking a typical fingertip trajectory of a user encircling an object of interest while standing still. The maximum estimation errors (given in pixels) for the Kalman filter and the robust fingertip tracker can be found in Figs. 17 and 18. The maximum estimation errors for the Kalman filter are

$$\tilde{x}_{\max} = 15.4581\ (\text{pixels}), \quad \tilde{y}_{\max} = 9.9089\ (\text{pixels})$$
$$\tilde{\Delta x}_{\max} = 17.3755\ (\text{pixels}), \quad \tilde{\Delta y}_{\max} = 15.5054\ (\text{pixels})$$

and the maximum estimation errors for the robust filter with genetic perturbation models were found to be

$$\tilde{x}_{\max} = 15.2897\ (\text{pixels}), \quad \tilde{y}_{\max} = 10.1077\ (\text{pixels})$$
$$\tilde{\Delta x}_{\max} = 15.5645\ (\text{pixels}), \quad \tilde{\Delta y}_{\max} = 9.0860\ (\text{pixels}).$$

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 ($\Delta W_x \geq 2\tilde{x}_{\max}, \Delta W_y \geq 2\tilde{y}_{\max}$), 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. For these simulations, where the maximum estimation errors for the x and y coordinates for both filters were very close in magnitude to each other, there was not a clear advantage on the reduced search window size between one filter to the other filter. However, the maximum estimation error for the Δx and Δy displacements (i.e., for the speeds in the x and y directions, respectively) are significantly larger for the Kalman filter in comparison to the robust filter when the user speeds up his hand movements. Therefore, the robust finger tracker leads to a more stable response than the Kalman filter under the condition when the user points at an object with fast gestures, as illustrated in Figs. 16 and Fig. 18.

REFERENCES

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

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

[3] A. Wu, M. Shah, and N. Da Vitoria Lobo, "A virtual 3D blackboard: 3D finger tracking using a single camera," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 2000, pp. 536–543.

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

[5] K. Imagawa, L. Shan, and S. Igi, "Color-based hands tracking system for sign language recognition," in *Proc. Conf. Automatic Face and Gesture Recognition*, Nara, Japan, 1998, pp. 462–467.

[6] F. K. H. Quek, T. Mysliwiec, and M. Zhao, "Finger mouse: A freehand pointing interface," in *Proc. Int. Workshop on Automatic Face and Gesture Recognition*, Zürich, Switzerland, June 1995, pp. 372–377.

[7] X. Zhu, J. Yang, and A. Waibel, "Segmenting hands of arbitrary color," in *Proc. Conf. Automatic Face and Gesture Recognition*, Grenoble, France, 2000, pp. 446–453.

[8] S. M. Dominguez, T. Keaton, and A. H. Sayed, "Comparison of robust estimation and Kalman filtering applied to fingertip tracking in human-machine interfaces," in *Proc. Asilomar Conf. Signal, Systems & Computers*, Pacific Grove, CA, Nov. 2001, pp. 342–346.

[9] S. M. Dominguez, T. Keaton, and A. H. Sayed, "Robust finger tracking for wearable computer interfacing," in *Proc. Perceptive User Interfaces*, Orlando, FL, Nov. 2001 [Online]. Available: http://www.cs.ucsb.edu/conferences/PUI/PUIWorkshop/

[10] T. Keaton, S. M. Dominguez, and A. H. Sayed, "Snap&Tell™: A Vision-Based Wearable System to Support Web-On-The-World Applications," in *Proc. Digital Image Computing - Techniques and Applications (DICTA) Conference*, Melbourne, Australia, Jan. 2002, pp. 92–97.

[11] T. Keaton, S. M. Dominguez, and A. H. Sayed, "Snap&Tell™: A multimodal wearable computer interface for browsing the environment," in *Proc. Int. Symp. Wearable Computers*, Seatle, WA, Oct. 2002, pp. 75–82.

[12] T. Keaton, S. M. Dominguez, and A. H. Sayed, "Browsing the environment with the Snap&Tell™ wearable computer system," *Pers. Ubiquitous Comput. J.*, vol. 9, no. 6, pp. 343–355, Dec 2005.

[13] D. Comaniciu and P. Meer, "Robust analysis of feature space: Color image segmentation," in *Proc. Conf. Computer Vision and Pattern Recognition*, San Juan, PR, 1997, pp. 750–755.

[14] F. Van der Heijden, *Image Based Measurement Systems*. New York: Wiley, 1994.

[15] A. H. Sayed, "A framework for state-space estimation with uncertain models," *IEEE Trans. Automat. Contr.*, vol. 46, no. 7, pp. 998–1013, Jul. 2001.

[16] R. K. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Trans. Automat. Contr.*, vol. AC-15, pp. 175–183, 1970.

[17] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Upper Saddle River, NJ: Prentice-Hall, 2000.

[18] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ: Wiley, 2003.

[19] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley-Interscience, 2001.

[20] V. Colin de Verdiere and J. L. Crowley, "Visual recognition using local appearance," in *Proc. Eur. Conf. Computer Vision*, Frieburg, Germany, 1998.

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

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Proc. Roy. Statist. Soc.*, vol. B39, pp. 1–38, 1977.

[23] Z. Ghahramani and G. E. Hinton, The EM algorithm for mixtures of factor analyzers Univ. Toronto, Toronto, ON, Canada, Tech. Rep. CRG-TR-96-1, 1996.

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

[25] C. Tomasi and T. Kanade, Detection and tracking of point features Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, Apr. 1991.

Sylvia M. Dominguez received the B.S. degree in electrical engineering (magna cum laude) from the University of Texas at El Paso, the M.S. degree in electrical engineering (with highest honors) from New Mexico State University, Las Cruces, and the Engineering degree (with high honors) from the University of California, Los Angeles (UCLA), where she is currently pursuing the Ph.D. degree in electrical engineering.

Her research experience include computer vision, signal processing, robust tracking systems, adaptive filtering, human computer interfacing, graphical modeling, integrated VLSI design, bio- controls, and hearing aid design. She has two patents, two scientific journals, and six conference publications based on her research work. She has been working for the past four years as a Senior Patent Engineer for the Intellectual Property Law Firm of Tope-McKay & Associates. Previously, she worked for two years as a Research Intern at HRL Laboratories in Malibu CA. In addition, she was a Lecturer for the UCLA Center for Excellence in Engineering and Diversity for seven years.

Trish Keaton (M'04) is currently pursuing the Ph.D. degree in electrical engineering at the California Institute of Technology.

She is a Program Manager for Northrop Grumman Corporation. Prior to joining Northrop Grumman, she was a Senior Scientist at Rockwell Scientific Company, and a Research Scientist at HRL Laboratories (formerly Hughes Research Laboratories), where she was the Principal Investigator of projects focused on 3-D human tracking and activity recognition with applications involving surveillance, ubiquitous and wearable computing. She has two patents awarded, five patents pending, and multiple journal and conference publications based on her research. Her research interests include computer vision, robust tracking systems, human computer interfacing, software product lines, graphical modeling, level set methods, and multimedia indexing and retrieval.

Ali H. Sayed (F'01) is Professor and Chairman of electrical engineering at the University of California, Los Angeles (UCLA). He is also the Principal Investigator of the UCLA Adaptive Systems Laboratory (www.ee.ucla.edu/asl). He has over 250 journal and conference publications, is the author of the textbook *Fundamentals of Adaptive Filtering* (Wiley, 2003), and is coauthor of the research monograph *Indefinite Quadratic Estimation and Control* (SIAM, 1999) and of the graduate-level textbook *Linear Estimation* (Prentice-Hall, 2000). He is also co-editor of the volume *Fast Reliable Algorithms for Matrices with Structure* (SIAM, 1999). He has contributed several articles to engineering and mathematical encyclopedias and handbooks and has served on the program committees of several international meetings. His research interests span several areas, including adaptive and statistical signal processing, filtering and estimation theories, signal processing for communications, interplays between signal processing and control methodologies, system theory, and fast algorithms for large-scale problems.

Dr. Sayed received the 1996 IEEE D. G. Fink Prize, a 2002 Best Paper Award from the IEEE Signal Processing Society, the 2003 Kuwait Prize, the 2005 Terman Award, a 2005 Young Author Best Paper Award from the IEEE Signal Processing Society, and two Best Student Paper Awards at international meetings (1999, 2001). He served as a Distinguished Lecturer of the IEEE Signal Processing Society during 2005. He was also a member of the Publications (2003–2005) and Award (2005) Boards of the IEEE Signal Processing Society. He is a member of the technical committees on Signal Processing Theory and Methods and Signal Processing for Communications, both of the IEEE Signal Processing Society. He currently serves as General Chairman of ICASSP 2008. He has served as Editor-in-Chief of the IEEE TRANSACTIONS ON SIGNAL PROCESSING (2003-2005) and is now serving as Editor-in-Chief of the EURASIP *Journal on Applied Signal Processing*.