Counting Fingers in Real Time Using Computer-Vision Techniques

Stephen C. Crampton, M.A., Margrit Betke, Ph.D., *member, IEEE,* and Stan Sclaroff, Ph.D., *senior member, IEEE*

***Abstract*— A video-based human-computer interface called the *Finger Counter* was developed to allow control of a computer with hand signals. The interface works in real time with an inexpensive, consumer-grade camera, such as a webcam. A background-modeling algorithm compensates for camera noise and minor vibration to segment the hand from a cluttered background. A rule-based protrusion estimator determines the number of fingers a user holds up in front of the camera. The interface was tested in a game designed to teach children to count with their fingers and a program that allows a user to "finger paint" on a computer screen. Tests determined that the hand signal most difficult to detect is "5," with worst-case operational limits on the range of hand's roll, pitch, and yaw angles of 40°, 114° and 21°, respectively, and of distance between hand and camera of 0.67 m. Tests involving 20 subjects showed that the system successfully recognized gestures between 82.5% and 100% of the time. Response times averaging 2.38 s were comparable to those using a keyboard. In tests involving the "finger paint" program, the median error was 19.5 pixel units compared with 14.2 pixel units for a computer mouse.**

***Index Terms*— Image segmentation, Image classification, Image shape analysis, User interfaces**

I. Introduction

The usefulness of hand gestures as a mode of communication has motivated the design of numerous gesture-based computer interfaces [14], [20], [36], [38], [44], [46], [60], [63]. Most of these systems were developed in laboratory conditions, that is, with high-quality cameras on stable tripods under controlled lighting. For a video interface to be useful to general users, it should

- function reliably with an inexpensive camera and mount
- allow a wide array of lighting and background conditions
- be easy and intuitive

The *Finger Counter* is a vision-based interface that aims to meet these goals. Figure 1 shows a typical system configuration. A user places a video camera, such as an inexpensive "webcam" designed to interface with a personal computer, on a table or other surface looking upward. Then, the user makes hand signals above the camera, assisted by a "user feedback" window on the screen. The interface works without special gloves, lighting, cameras, or other equipment. It requires virtually no training on the part of the user because the gestures it recognizes – fingers held up in front of the camera – are simple and intuitive. The system tolerates minor camera movements, including vibration, that shift the hand in the image by not more than a few pixels. The system does not require a database of training images. Experiments show that the response time using the *Finger Counter* is comparable to the time it takes to select a key from a keyboard.

This work was supported by the National Science Foundation (IIS-0093367, IIS-0329009, and EIA-0202067) and the Offi ce of Naval Research (N000014 -01-1-0444).

Fig. 1. The *Finger Counter*. The webcam (on the table) points upward at the hand.

Hand gestures can be divided into three parts [30], [39]: (1) *preparation,* or movement of the hand into position; (2) *stroke,* the gesture itself; and (3) *retraction,* a withdrawal of the hand to a neutral posture. Preparation and retraction are often similar across many different gestures; the stroke contains the communicative content [30]. The stroke may consist of a static hand pose or a hand pose coupled with movement [39]. The literature [31], [39], [40] divides hand gestures into four categories: *Gesticulation* accompanies speech to emphasize or enhance communicative meaning. *Pantomime* connotes a performance of some kind, without speech. An *emblem* is a static gesture that has a particular meaning, such as the "O.K." sign. Finally, a *sign* is a grammatical element in a sign language, such as American Sign Language.

The category of hand gestures recognized by the *Finger Counter* does not fit precisely into a canonical category, though it is most similar to emblems. Like emblems, the gestures recognized by the system are unaccompanied by speech. Also like emblems, they are static in the sense that gestural meaning is unconnected with movement. For example, if a user holds up a single finger in the *Finger Counter*'s "finger paint" application, the meaning is that a single paint brush is chosen. Unlike emblems, however, the entire hand's motion around the camera's field of view can act as another channel of communication. In the "finger paint" application, for example, the sweep of the user's fingertip moves the selected brush to draw on the screen. To distinguish these types of gestures from emblems, this article uses the term *hand signal* to characterize the types of gestures recognized by the system. For purposes

of this article, hand signals comprise a hand posture that sometimes is in motion.

The appearance of a hand posture varies due to perspective effects, the physical characteristics of a particular user's hand, and how a user might form a particular posture [47]. Experiments show that the *Finger Counter*'s rules-based system for determining hand signals can handle a wide variety of hand-posture appearances.

An earlier version of the *Finger Counter* interface [9] used edge-finding techniques to determine the hand contour, whereas the system described here uses a contour-following algorithm. The new system also enhances the background-differencing method of the earlier system by compensating for radial distortion and sensor noise.

II. Related work

To estimate a hand pose, or series of hand poses, from one or more images, most systems go through the following stages: *segmentation* of the hand from the background; *extraction of a feature vector*, containing parameters relevant to a classifier, from the hand image; and *classification* of the hand gesture. This section describes how previous researchers have addressed these challenges.

A. Segmentation

In gesture recognition, hand segmentation is often posed as the separation of pixels in the image into two categories: those corresponding to the hand and those corresponding to the background. Many systems assume that the background is uniform and neutral, so that the hand is easily segmented by thresholding intensity values [1], [10], [15], [59]. A common approach is to have a user hold his or her hand over a black matte surface, such as a black cloth [34] or require the user to wear special gloves [29], [42], [53], wrist bands [37], or other markers [20]. Sometimes, researchers employ special lighting to improve segmentation results [33]. Another common technique is to classify pixels according to their similarity to skin color [4], [13], [21], [32], [37], [48], [51], [64], or proximity to or dissimilarity with other skin-colored pixels [36], [51], though segmentation using skin color can suffer from the fact that skin-colored objects in the background can cause confusion. Another technique for hand segmentation compares successive images in a video stream to detect and track motion [4], [11], [13], [16], [17], [36], [43], [63]. Some systems combine skin-color and motion-detection methods [13], [45], [61]. Still other systems use a thermal infrared camera and threshold the image under the assumption that the hand is the warmest object in the field of view [43]. Other systems use multiple cameras to determine the 3D position of a hand [20], [27], [38] and threshold a composite image by distance, assuming the hand is closer to the camera than the background [27]. Thermal and 3D systems require special cameras and equipment, something the *Finger Counter* aims to avoid.

The *Finger Counter* uses a background alignment method to detect the hand and segment it from the background. A model of a background image, without the hand in it, is dynamically maintained. When the user's hand appears, the algorithm estimates the best translational alignment between the central subimage of the incoming video frame with the current background model. This subimage alignment compensates for the types of camera motion that might be found in a home or office environment, namely, minor perturbations of the camera due to, for example, vibration from an air conditioner. This is in contrast with systems that assume a fixed camera and changing backgrounds (e.g., [36], [52], [55]), for example, to model the changes in outdoor lighting during long-term surveillance of pedestrian and cars.

B. Feature extraction

The second major stage in hand-pose recognition, feature extraction, is the conversion of the color or grayscale pixels of interest into "features," or parameters that may be relevant to a classifier. Sometimes the feature is computed by transforming the image, for example, skin-colored pixels into grayscale [32]. Other systems transform the segmented image into a binary image [29], [33], [37]. Laptev and Lindeberg's system applies linear transformations, extracting features at different scales [35]. Still other systems extract edges, defined as local maxima in the intensity gradient, for use as features [47]; this technique was used in the prior version of the *Finger Counter*, but edges found in the creases of the palm and fingers tended to confuse the recognition process. Freeman et al. report a technique using orientation histograms, which tally the discretized intensity gradient directions at each pixel location in the hand image [16], [17]. Freeman et al.'s method estimates the hand orientation, but does not give specific information as to the hand pose.

The *Finger Counter* system falls into the category of systems that determine the contour, or boundary, between the hand and non-hand pixels and use that contour to derive features. Some such systems process contours found in the image before attempting classification [7], [22]. Such systems determine the number of finger-like protrusions from the hand contour and use that information, combined with other features, to determine hand pose. For example, Athitsos and Sclaroff's system identifies fingertip candidates as points of maximal curvature between inflection points in the hand contour [1]. Finger-like protrusions are labeled as those protrusions that are large enough and whose elongation exceeds a specified threshold. Kumar and Segan analyze the hand contour for points of maximum and minimum curvature to estimate the number of finger-like protrusions and their minimum and maximum extent [34]. Quek et al. use a finite-state machine to determine the position and length of finger-like protrusions [48].

The *Finger Counter* system converts the contour's Cartesian coordinates to polar coordinates for protrusion analysis. This takes advantage of the fact that extended fingers "radiate" outward from the center of the palm and also makes recognition invariant to a significant range of degrees of rotation parallel to the image plane. That is, the hand need not be held strictly upright for recognition to succeed. The idea of using a polar-coordinate representation of the hand contour has been exploited previously, e.g., [2], [5], [24], [38], [43], [62].

C. Classification

The third aspect of hand-gesture recognition is classification of the feature vector into one of several hand poses. To facilitate comparison, some researchers create models of the hand encoded in the system without using training data. For instance, Laptev and Lindeberg model the palm as a large Gaussian "blob" of image intensity values and each outstretched finger as two smaller, elongated blobs with a difference-of-Gaussians "ridge" for the fingertip [35]. They report no empirical results. Triesch and von der Malsburg model hand postures as graphs; the graph nodes are "jets," or vectors composed of the responses of Gabor wavelets of different sizes and orientations to image intensity values centered at a particular image location [57]. Through a process known as elastic graph matching, the graphs representing hand postures are warped to fit to image features; the best fit results in a classification of that hand posture. In addition to the graphs, which are predetermined, the system uses training images to determine the parameters of the jets at each node. Training images consist of examples of each hand pose against light and dark backgrounds. Triesch and von der Malsburg report that their system correctly recognized gestures in 92.9% of 604 images with a uniform background and in 85.8% of 338 images with a a complex background.

Other systems also must be trained with sample images of the hand in different postures. In Athitsos and Sclaroff's system [1], the number and position of apparent fingertips in an image are compared to a database of over 100,000 images representing 26 hand poses subject to a wide range of 3D rotations representing different possible views of the hand [1]. To supplement the comparisons between the number and position of apparent fingertips in image and training data, Athitsos and Sclaroff use similarity measures such as Chamfer distance, edge-orientation histograms, and moments of least inertia. Lockton and Fitzgibbon describe a system that compares the silhouette image of the hand to previously stored templates of 46 static hand postures [37] using a technique similar to adaptive boosting [18]. Limitations of their approach include the large number of templates required (3000) and the fact that the lighting conditions must be similar for the images used to build templates and the run-time conditions.

Other classification methods that require training include neural nets and hidden Markov models. Kjeldsen and Kender use neural nets to determine hand pose [32]. Given 50 samples of three hand poses for training, they achieved 85-90% recognition rates using 100 test images. Hidden Markov models are suitable for recognition of dynamic hand gestures [7], [49], [51], [58], [59]. Starner et al. extract from the hand silhouette sixteen geometrical measurements, based upon various moments of inertia, and feed them into a hidden Markov model that can recognize five dynamic hand gestures. Recognition rates range from 74.5% to 97.8%, depending upon the test conditions. In other work, the input to the hidden Markov models are the 3D positions of both hands, as determined by a stereoscopic camera setup against a uniform background [38], [59], or projection of low-resolution grayscale images of the hand into an eigenspace using principal components analysis [49].

Like *Finger Counter*, some researchers use rules-based systems to analyze geometrical information. This approach has the advantage that it does not require training. For instance, Cutler and Turk use a rules-based system to determine the intended gesture based upon the geometrical characteristics of an ellipse constraining the imaged hand [11]. Kumar and Segan classify a hand gesture based upon the number of maxima and minima protrusions [34]. Jo et al.'s system recognizes dynamic gestures, such as "grasp," by analyzing a series of frames for differences in the hand area, centroid position, and maximum protrusion length [29]. Quek et al. use a rules-based system with inputs such as the area of a bounding box containing the hand and the geometrical moments [47]. Ji et al. use a row-by-row scanning algorithm to count how many fingers are held up [28]. Such a system depends upon the finger-like protrusions being strictly vertical. The system recognizes zero, one, or two fingers.

Some systems eschew the segmentation and feature-extraction stages, making a classification directly from the video image [10], [16], [17]. For example, Darrell and Pentland use normalized correlation to match view models of particular hand postures with the image [12]. The disadvantage of such a technique is that a new model must be created for each view.

III. The Method

The goal is to estimate a user's hand signal given one or more images. Such an endeavor raises a number of challenges. Images of the same hand pose can look different when made by different people under different lighting and background conditions. Moreover, the same hand pose appears differently when the hand is held in a different position or orientation with respect to the camera.

When the *Finger Counter* interface is started, it captures an image without the user's hand in the field of view, the "background image." Then, as it captures subsequent frames with the user's hand present, it subtracts them from the background image to segment, or identify, the pixels in the image composing the user's hand. The background differencing is done in such a way as to compensate for slight camera motion from side to side or up and down. Once the hand region is identified in the image, the *Finger Counter* determines the hand contour, computes the estimated center of the palm, and then counts the number of fingers protruding from the palm. To exploit the temporal continuity inherent in video sequences, the system reports a particular posture only when it detects the same posture over a series of video frames. This way, the system does not make spurious identifications while the user is in the preparation or retraction phases of gesture-making.

Finger Counter limits itself to a six-signal alphabet $\Sigma = \{\emptyset, $  $, $ $, $ $, $ $, $ $\}$. In addition, to simplify the estimation, *Finger Counter* requires the user to use a single hand, place it in full view of the camera, and keep the palm more or less parallel to the image plane. The background in front of which hand poses are made need not be uniform or planar, however, it must be stationary, and should not contain large regions of colors similar to the color of user's hand.

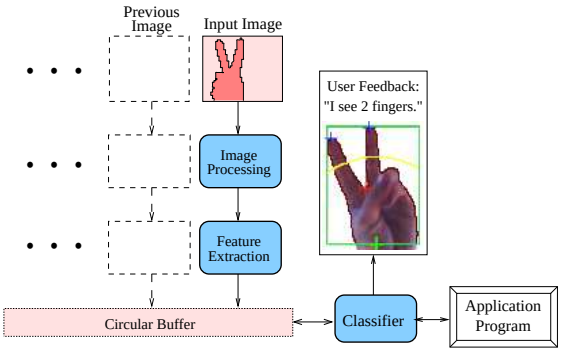

Fig. 2. System overview

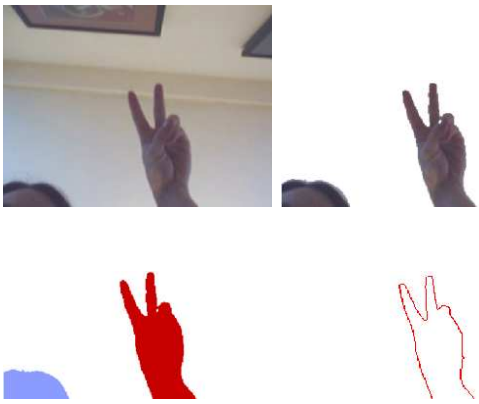

Fig. 3. Example results of image-processing steps in *Finger Counter*'s Image Processing Module. The first image is shown before processing. In the next image, the foreground is segmented from the background. In the third image, connected components are labeled by color. In the last image, the contour of the largest connected component was identified.

Finger Counter estimates a hand signal $\hat{S}$ from a video image I using the three modules shown in Fig. 2: (1) The "Image Processing" module takes a raw video image and processes it to determine the contour of the hand in the image. (2) The "Feature Extraction" module estimates, from a single processed image, how many fingers are held up and the fingertip positions. The feature vector containing the number of fingertips and their estimated positions is placed into a circular buffer. (3) The "Classifier" module examines the buffer to compare the number and location of fingertips over a series of frames and to estimate S. This estimate is considered the classifier's "state." The application program at any time can examine the state of the Classifier and take action accordingly.

Frames are captured continuously as the user moves his or her hand into position to form a hand signal and also as the user retracts the hand. To determine whether an estimated hand posture $\hat{S}$ is the user's intended signal S, as opposed to a transitory movement between signals, the user is required to maintain the hand posture for a short period of time. To assist the user, a "feedback" window is provided, so that the user can see how the system "sees" his or her hand. To make the feedback window more intuitive, the image is mirrored. The system also gives audio cues in the form of a voice that makes statements such as "I see two fingers."

A. Image processing

The image-processing module performs two major tasks: it segments the foreground containing the hand from the background and determines the hand's contour (see Fig. 3). To perform the segmentation, the background-differencing algorithm adapts to changes in camera position. The system then determines the contour between the largest foreground region and the background region in the image.

1) Background differencing: When it starts, the algorithm captures and stores an image $\mathbf{B_0}$ of the background. It assumes that $\mathbf{B_0}$ does not include the user's hand. When the user puts his or her hand in front of the camera, the hand is segmented from the background by subtracting $\mathbf{B_0}$ from the image with the hand. Pixels where the difference is significant, that is, more than the typical range of intensity variation due to camera noise, are considered to be foreground pixels.

Let $\mathbf{I}_t$ be the image of dimensions $M \times N$ acquired by the camera at time t. This is a color image with three channels; thus, $\mathbf{I}_t(i,j) = \left(I_t^R(i,j), I_t^G(i,j), I_t^B(i,j)\right)^\top$, with I_t^R, I_t^G, and I_t^B denoting the red, green, and blue components respectively. The initial background image is $\mathbf{B}_0 = \mathbf{I}_0$. The *Finger Counter* system models a pixel at image coordinates (i,j) of the scene background as $B_0^{\mathcal{C}}(i,j) + N^{\mathcal{C}}(0,\sigma^2)$, where $\mathcal{C}$ is the color channel and $N^{\mathcal{C}}(0,\sigma^2)$ represents a normal, zero-mean noise term of unknown variance. This is analogous to an experimentally validated model proposed for grayscale cameras [3]. The noise term is typically small compared to the dynamic range of the image for charge-coupled-device cameras. Such cameras have a noise level on the order of $\sigma = 1.3$ out of 256 intensity levels [41].

Starting with time $t = 1$, let $\mathbf{I'}_t$ be the $M' \times N'$ subimage of $\mathbf{I}_t$ aligned at $\mathbf{u}_0 = 1/2(M - M', N - N')$, that is, $\mathbf{I'}_t = \mathbf{I}_t(\mathbf{u}_0 + (i,j))$. To perform background differencing at time $t+1$, the system compares $\mathbf{I'}_t$ with subimage $\mathbf{B'}_{t-1} = \mathbf{B}_0(\mathbf{u}_t + (i,j))$, also of dimensions $M' \times N'$, as shown in Fig. 4. Each pixel in $\mathbf{B'}_t$ is subtracted from the corresponding pixel in $\mathbf{I'}_t$ and the difference compared to two thresholds. The result is a binary mask F_t with "one" pixels for pixels judged to be in the foreground and "zero" pixels elsewhere. With respect to the first threshold,

$$F_t(i,j) = \begin{cases} 1 & \left|\mathbf{I'}_t^{\mathcal{C}}(i,j) - \mathbf{B'}_t^{\mathcal{C}}(i,j)\right| > \tau^{\mathcal{C}} \ \forall \mathcal{C} \in \{R,G,B\} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Threshold $\tau^{\mathcal{C}}$ is computed as $\tau^{\mathcal{C}} = \kappa \max_{(i,j)}\{\mathbf{I'}_t^{\mathcal{C}}(i,j) - \mathbf{B'}_t^{\mathcal{C}}(i,j)\}$, where $0 < \kappa < 1$ and κ was chosen empirically. Because $\tau^{\mathcal{C}}$ is a fraction of the maximum difference value for a particular color channel $\mathcal{C}$, there will always be pixels that exceed the threshold and would incorrectly be judged foreground pixels, even when the only differences in the images are due to sensor noise. Therefore, a second, absolute threshold $\tau'^{\mathcal{C}}$ is used for each color channel to segment foreground pixels. In particular, for all $\mathcal{C} \in \{R,G,B\}$, if $\mathbf{I'}_t^{\mathcal{C}}(i,j) - \mathbf{B'}_t^{\mathcal{C}}(i,j) < \tau'^{\mathcal{C}}$ then $F_t(i,j) = 0$ for all (i,j).

If none of the intensity differences exceed the absolute threshold, the system acquires a new background image $\mathbf{B_0}$. In this way, the system adapts to slow changes in the background,

Fig. 4. Alignment of background image. Note that the fi gure exaggerates the displacement for illustration purposes. At left is background image $\mathbf{B}_0 = \mathbf{I}_0$ at time $t = 0$. Subimage $\mathbf{B}'_0 = \mathbf{I}'_0$ is inside the black rectangle. In the middle is new subimage $\mathbf{I'}_t$, obtained at time t from camera's full frame I_t. At right is background image $\mathbf{B_0}$ with background subimage $\mathbf{B}'_t$ aligned to correspond with $\mathbf{I'}_t$ at time t.

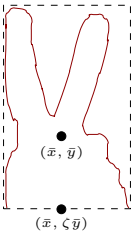

Fig. 5. Region of interest with contour centroid $(\bar{x}, \bar{y})$ and estimated palm center $(\bar{x}, \zeta\bar{y})$.

typically whenever there is not a hand in the scene. To adapt to dramatic changes in the background, the program allows the user to trigger the system's capture of a new background image by pressing a key on the keyboard. Values for all parameters discussed in this chapter, including M, N, M', N', κ, and $\tau'^{\mathcal{C}}$, are given in Section IV.

To update estimated offset $\hat{\mathbf{u}}_{t+1}$, the algorithm minimizes the squared difference between background pixels in $\mathbf{I'}_t$ and the corresponding pixels in $\mathbf{B}_0$ over all possible offsets $\{\mathbf{u}\}$:

$$\hat{\mathbf{u}}_{t+1} = \underset{\mathbf{u}}{\operatorname{argmin}} \sum_{\substack{(i,j) \in \mathcal{B}_t \text{ and} \\ \mathcal{C} \in \{R,G,B\}}} \left(I_t'^{\mathcal{C}}(i,j) - B_0^{\mathcal{C}}\left(\mathbf{u} + (i,j)\right)\right)^2, \tag{2}$$

where $\mathcal{B}_t = \{(i,j) \mid F_t(i,j) = 0\}$ is the set of background pixel coordinates.

The algorithm compensates for combinations of side-to-side and up-and-down camera motion, for example, when a camera is subject to vibration from an air conditioner. Without loss of generality, consider translation t_x parallel to the x-axis of the camera's image plane. Before the camera is moved, a world point at distance Z_0 from the camera is imaged at x_1 and afterward at x_2. Under perspective projection, the disparity δ_x is expressed as follows:

$$\delta_x = x_2 - x_1 = \frac{t_x f}{Z_0}, \tag{3}$$

where f is the camera's focal length. Disparity δ_x is the ground truth for the x-component of *Finger Counter*'s estimate $\hat{\mathbf{u}} = (\hat{u}_x, \hat{u}_y)$ (Equation 2) if the background consists of a planar surface at Z_0, in our experiments typically the ceiling, about 2 meters overhead. For a relatively large camera motion of $t_x = 1$ cm, the disparity for $Z_0 = 2$ m, with focal length $f = 2.24$ mm and pixel size 5.6 μm, is only 2 pixel units. If the background is not strictly planar, but contains an object at distance $X_0 - \Delta$, the disparity between its image before and after camera motion is only 2.5 pixel units. As the disparities are small, the difference between them is even smaller, in this case, half a pixel unit, which shows that the *Finger Counter*'s background-estimation method would be expected to work under environments where the background is not strictly planar. As another example, for $Z_0 = 1$ m and $\Delta =$ 0.1 m, the difference between disparities is less than half a pixel unit.

2) Finding the hand contour: ¿From the foreground image F, the system finds the contour of the largest connected component in the image. To do this, first, regions of foreground pixels are collected into connected components [25]. Connected components are identified as regions of eight-connected pixels. Once connected components are identified, all but the largest are discarded, and an iterative boundary-following algorithm [26] finds its contour C.

The centroid of C is $(\bar{x}, \bar{y}) = \frac{1}{A}\sum_{(i,j)\in C}(i,j)$, where A is the number of pixels in C. ¿From the centroid, *Finger Counter* estimates the palm center $(\bar{x}, \zeta\bar{y})$; the parameter $\zeta = 0.67$ was chosen in developing the system and was appropriate for all five hand signals. Parameter ζ scales the vertical location of the centroid of hand contour pixels downward in the image to estimate the palm center, based on the assumption that the hand is held approximately upright. The system also defines a rectangular region of interest containing all contour pixels level with or above the estimated palm center $(\bar{x}, \zeta\bar{y})$, as shown in Fig. 5.

To estimate the position of the center of the palm, the centroid's y coordinate was multiplied by $\zeta = 0.67$.

B. Feature extraction

Human fingers when extended radiate outward from the palm, a trait *Finger Counter* exploits. Pixels in the contour image C, referenced by Cartesian coordinates (i,j), are converted to polar coordinates (θ, r) with the origin at the estimated palm center $(\bar{x}, \zeta\bar{y})$. Angle θ is quantized by using $\theta_p = p^\circ$, for $p \in \{0, 1, 2, \cdots, 180\}$. For each θ_p, a single edge pixel $\mathbf{e}_p \in C$ with radius r_p is selected as follows: Let $\mathcal{A}_p$ be the set of edge pixels $\{\mathbf{e}_k\} = \{(x_{\mathbf{e}_k}, y_{\mathbf{e}_k})\}$ for which the closest ray extending from the polar-coordinate origin is at angle θ_p, that is, $\mathcal{A}_p$ is the set of edge pixels that fall between rays extending from the polar-coordinate origin at angles $\theta_p - 0.5^\circ$ and $\theta_p + 0.5^\circ$ (see Fig. 6). Then, $\mathcal{A}_p$ is defined by the following equation:

$$\mathcal{A}_p = \left\{ \mathbf{e}_k \ : \ \left| \arctan\left(\frac{y_{\mathbf{e}_k} - \zeta\bar{y}}{x_{\mathbf{e}_k} - \bar{x}} \right) \right| < \theta_p + 0.5^\circ \right\}. \tag{4}$$

¿From each $\mathcal{A}_p$, the system chooses the farthest edge pixel from the polar-coordinates origin, that is, $\mathbf{e}_p = \operatorname{argmax}_{\mathbf{e}_k \in \mathcal{A}_p} |\mathbf{e}_k - (\bar{x}, \zeta\bar{y})|$. The radius r_p corresponding to angle θ_p becomes simply $r_p = |\mathbf{e}_p - (\bar{x}, \zeta\bar{y})|$. Figure 7 illustrates the result of this conversion on a contour image.

To count fingers the system first sets threshold τ_r to be a fraction of the maximum r in the region of interest, that is, $\tau_r = c \max r$ (see Fig. 7). Let t be the frame number and k range from 1 through the total number of protrusions

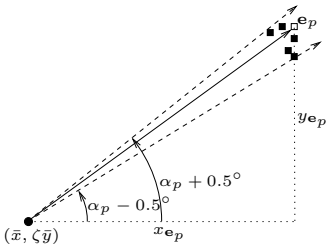

Fig. 6. Illustration of how a contour pixel $\mathbf{e}_p$ is selected for a discrete polar-coordinate angle θ_p. The squares represent the pixels between rays extending from the palm center $(\bar{x}, \zeta\bar{y})$ at angles $\theta_p - 0.5°$ and $\theta_p + 0.5°$. The hollow square is $\mathbf{e}_p$, the farthest pixel.

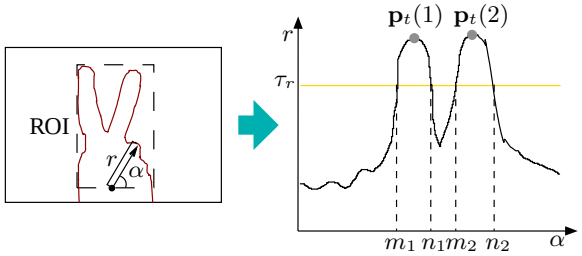

Fig. 7. Illustration of transformation from Cartesian (x, y) to polar coordinates (r, θ). Cartesian coordinate origin is at $(\bar{x}, \zeta\bar{y})$.

found. The system identifies the location $\mathbf{p}_t(k)$ of the tip of finger-like protrusion number k in frame number t as a local maximum in a range of farthest-edge pixels for which the ℓ_2-norm exceeds τ_r. If m_k defines the lower end of a range and n_k the upper end, then, for all p such that $m_k \leq p \leq n_k$, all pixels are found with $|\mathbf{e}_p| > \tau_r$. Then, $\mathbf{p}_t(k) = \text{argmax}_{\mathbf{e}_p}|\mathbf{e}_p|$. The ranges must be disjoint, that is, $0 \leq m_1 \leq n_1 < m_2 \leq n_2 < \cdots < m_k \leq n_k \leq 180$, to identify separate fingers.

Let ν_t be the number of protrusions found in frame t. The system stores ν_t and $\mathbf{p}_t = \{\mathbf{p}_t(1), \mathbf{p}_t(2), \cdots \mathbf{p}_t(\nu_t)\}$ in its circular buffer.

C. Classification

The number of fingertips identified in an image, provided it is between 1 and 5, categorizes the hand pose. However, as a user moves his or her hand in preparation for making a hand pose or retracts the hand, the system may capture images and report a different number of fingers from what the user intended. To determine whether an estimated hand posture $\hat{S}$ is the intended hand signal, rather than a gestural preparation or retraction, requires the user to maintain the hand posture for a short period of time so the system can verify his or her intent. In this regard, *Finger Counter*'s Classifier module, at time t considers the last ρ records from the circular buffer for two conditions:

1) The ρ records report the same number of finger-like protrusions.
2) The squared distance in pixels between the same protrusion in successive frames is less than a threshold τ_ν.

The second condition requires that the user's hand be relatively stationary before a determination is made. If these conditions are met, then the classifier reports a finger count at time t.

IV. Implementation Details

The interface was implemented under Linux 2.4 on a laptop computer with a Pentium IV 1.4 GHz processor and 256 MB of RAM. Attached to the computer via a universal serial bus was a Logitech Quickcam 4000 Pro or a Creative Labs Webcam III, running (with compression) at 30 frames per second. The *Finger Counter* interface processes about 10 frames per second. Most of the delay comes from updating the background subimage alignment $\mathbf{u}_t$ (Eq. 2). Note that Eq. 2 could be computed in parallel using the SIMD (single instruction, multiple data) microprocessor instruction set to speed up the frame rate of the interface.

The webcam captured images of dimension $M \times N = 320 \times 240$. The images were cropped to dimensions $M' \times N' = 300 \times 226$, which maintained the aspect ratio of the image while allowing the background-differencing algorithm to correct for minor camera motion, i.e., motion at most 10 pixels to the left or right and at most 8 pixels up or down. In developing the interface, $\kappa = 0.2$ was found to appropriately separate pixel differences due to camera noise from those due to the presence of a foreground object. Similarly, the absolute threshold for differencing was set to $\tau'^{\mathcal{C}} = 40$; if the maximum value for all color channels of all pixels did not exceed that value, then all differences between the background and current image were considered to be due to noise. The percentage of maximum protrusion deemed to be a finger-like protrusion was determined to be $c = 0.75$. Higher values for c made it difficult to recognize short fingers while lower values caused the algorithm to begin recognizing knuckles as fingers. To suppress spurious recognitions, the circular buffer size was set to $\rho = 5$. Higher values than 5 tended to make the interface delay noticeable. Finally, $\tau_\nu = 800$ pixel units squared constrains the squared distance between estimated fingertip positions from frame to frame enough to ensure that the intended hand signal is recognized.

Radial distortion occurs when a camera lens causes straight lines near the edge of the field of view to appear curved. For the camera used with the interface, radial distortion [23] was found to be as much as 4.6 pixel units, which would have the potential to confound the background-differencing algorithm. Accordingly, the *Finger Counter* interface undistorts each image as it is received from the camera, including the initial background image $\mathbf{B_0}$.

To minimize the effect of sensor noise, incoming images are convolved with a 3×3 mask approximating an isotropic Gaussian function. Noise from the camera was found to be sufficiently suppressed by using a Gaussian function with standard deviation $\sigma = 1.0$.

The system uses several threads of execution [50] to improve the system's efficiency. The threads timeshare the central processing unit and access common global variables. The system uses one thread to interface with Video4Linux, a Linux module that facilitates communication with cameras attached to the system. Another thread processes incoming frames as described above and implements one of the application programs described below. A final thread handles audio output. Multithreading prevents the system from stalling while waiting

Fig. 8. Template to teach a user how to hold his or her hand in order to be recognized by the system

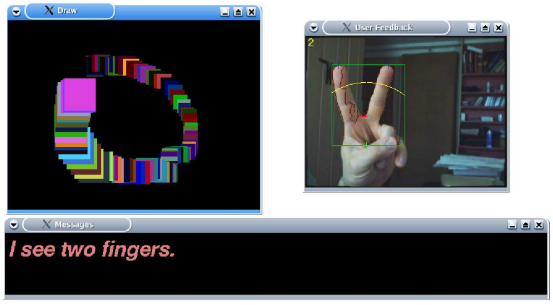

Fig. 9. "Finger paint" game: the bottom window gives text messages. The right window is the "user-feedback window," and the left window is the painting, created by a sequence of detected hand signals. In the example shown, the player moved her two fingers in a clock-wise circular motion, spreading them apart to increase the brush size.

for input or output and thus maintains seamless interaction with the user.

To acclimate a user to the interface, *Finger Counter* briefly displays the template shown in Fig. 8 and asks the user to fit his or her hand to it. This step teaches the user how to hold his or her hand in order to be recognized by the system.

V. Applications

Two applications were developed to demonstrate the hand-recognition capabilities of the *Finger Counter* interface. The first is a voice-interactive game, a program that audibly prompts the player to hold up a certain number of fingers and then counts them. The output of the system is an audio and text message regarding the number of fingers recognized. The second application allows the user to paint on the screen using her fingertips, controlling the brush size or quantity of brushes by modifying her hand pose. If a player holds up one finger, painting is done with one brush. If two fingers are held up, a single brush does the painting, and the player can vary the brush size dynamically by spreading or contracting the two fingers. Holding up three or four fingers allows the player to paint simultaneously with three or four brushes. Finally, holding up five fingers erases the entire image. A screenshot of the "finger paint" game is shown in Fig. 9.

VI. Experiments: Methodology and Results

Four evaluation methods were used to test the *Finger Counter* interface. The first was a series of tests designed to test to what degree a hand signal could be rotated with respect to, or translated toward or away from, the camera and still be recognized by the system. The next two methods were quantitative performance analyses, in which test subjects played modified versions of the games described in Section V: The second method assessed the accuracy and response time of the system in responding to hand signals in the voice-interactive game. The third method measured how well the interface estimated fingertip positions in the "finger paint" game. The fourth and final evaluation method, a questionnaire, was designed as a qualitative measure of *Finger Counter*'s usability as an interface.

A. Experiments to determine operating limits of the interface

For the first evaluation tool, the *Finger Counter* interface was run on a computer in a laboratory under fluorescent and incandescent lighting. The camera was placed on a tripod facing the ceiling, hand signals were formed above the camera, and then the position of the hand with respect to the camera was altered in one of the following ways:

1) The hand was moved closer to the camera.
2) The hand was moved away from the camera.
3) The wrist or forearm was rotated in the direction of one of the Euler angles around axes modeled to go through the center of the palm:
   - "Pitch": The wrist was rotated so the fingertips were closer to the camera than the palm or vice versa (wrist in flexion or extension).
   - "Roll": The forearm was rotated so that the side of the hand including the base of the little finger was closer than the side including the base of the thumb, or vice versa (forearm in pronation or supination).
   - "Yaw": The wrist was rotated in a plane parallel to the image plane, so that, from the interface camera's perspective, the fingertips moved to one side while the palm remained fixed (ulnar or radial deviation).

An additional digital camera was set up on a tripod next to the webcam to capture still images of hand positions for "out of plane" rotations, that is, pitch and roll. The user placed his hand over the camera so that the hand was oriented parallel to the camera lens, perpendicular to the bottom of the image frame, and the system properly recognized the hand signal. The hand was then moved in each manner listed above until recognition failed. The user then moved his hand back to the last point where the hand was consistently recognized.

A tape measure was used to determine how close and how far the hand can be from the camera. Figure 10 shows screenshots from the *Finger Counter* program taken when the hand was at the nearest and farthest distances respectively. The nearest and farthest distances depend upon the focal length of the camera. For the camera used in these tests, the focal length was 2.24 mm, which appears to be typical for webcams.

To measure pitch and roll, angles were measured from the images taken by the additional digital camera using an image-manipulation program. Angles were measured between the horizontal and a line from the center of the palm best representing the attitude of the hand. Yaw angles were measured from the vertical using screenshots from the *Finger Counter* program. Figure 11 shows sample measurements.

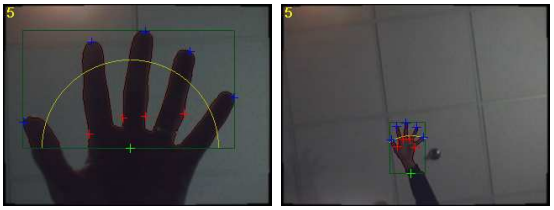

Fig. 10. Operating limits of the *Finger Counter*: Near and far distance range with correct recognition of signal .

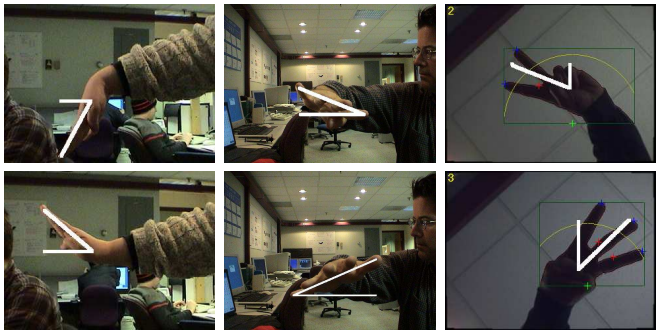

Fig. 11. Operating limits of the *Finger Counter*: Examples of the range of recognized orientations of the hand with respect to the camera. The white lines superimposed on the images show the x or y axis and the line used to measure the angle. In all cases, the *Finger Counter* camera faced upward. A second camera captured the images in the left and center columns; the second camera was leveled and placed either directly to the left or directly in front of the user. The left two images show pitch measurements, with wrist in flexion and then in extension. The middle two images show roll angles, with the forearm in pronation and in supination. The right two images show yaw measurements, taken by the *Finger Counter* camera, with the wrist in ulnar and then radial deviation.

The palm centers in the images on the right in Figures 10 and 11 are not correctly identified; however, for the purpose of hand signal classification, the estimated locations serve reliably as the origins of the polar coordinate transformation.

Tests were conducted on three subjects. Table I shows how subjects' hand sizes compare with a study of Air Force personnel [19] and a smaller study including the general public [6]. Test Subject 1 had a slightly larger than average hand, but the proportion of length to width was comparable to those in both anthropomorphic studies. Test Subject 2 had an average-sized hand, also with comparable proportion to those in the studies. Test Subject 3 had an average-sized hand, but the width of her hand was disproportionately larger than the length compared to an average female hand.

Table II shows the narrowest ranges for all test subjects for the various types of hand positions with respect to the camera. The signal was recognized in the narrowest range of 0.67 m, for instance, from 0.29 m to 0.96 m from the camera. The symbol was also least robust to rolls, recognized for all test subjects in a range of 40°. For pitch, the symbol was recognized in the narrowest range of 106°, for example, from 56° below the horizontal to 50° above. Finally, for yaw, the system was least resistant to rotations of the symbol, with a minimum range of 21°.

In tests of an earlier version of the *Finger Counter* interface [8], [9], the minimum ranges were as follows: distance, 0.4 m; pitch, 84°; roll, 14°; yaw, 13°. The earlier experiments were conducted with the hands of three volunteers: the slightly larger-than-average male hand described above, a smaller- and wider-than-average female hand, and an average-sized female hand of average proportions.

TABLE I
AVERAGE HAND (METACARPALE) MEASUREMENTS
N = SAMPLE SIZE, SD = STANDARD DEVIATION

	N	Length (± SD)	Width (± SD)	Ratio of Length to Width
Air Force male	148	19.7 cm (± 0.9 cm)	9.0 cm (± 0.4 cm)	2.2
General public male	15	18.7 cm (± 1.0 cm)	8.7 cm (± 0.5 cm)	2.2
Test Subject 1		20.4 cm	9.5 cm	2.2
Test Subject 2		19.1 cm	8.9 cm	2.2
Air Force female	211	17.9 cm (± 0.9 cm)	7.7 cm (± 0.4 cm)	2.3
General public female	15	16.7 cm (± 0.5 cm)	7.5 cm (± 0.3 cm)	2.2
Test Subject 3		17.8 cm	10.2 cm	1.8

TABLE II
NARROWEST RANGES OF HAND POSITION AND ORIENTATION WITH THREE TEST SUBJECTS USING FINGER COUNTER INTERFACE

Hand Signal	Distance	Roll	Pitch	Yaw
	1.99 m	321°	106°	150°
	1.87 m	126°	107°	123°
	1.87 m	100°	108°	100°
	1.42 m	78°	112°	57°
	0.67 m	40°	114°	21°

B. Experiments with voice-interactive game

Twenty volunteers played a version of the voice-interactive game that, for each frame, logged (1) the time since the last request was made, (2) the number of fingers requested, (3) the number of fingers detected for that frame, and (4) *Finger Counter*'s estimate of the number of fingers held up. The latter two numbers may differ, because, as described in Section III, the system uses a number of successive frames to reach a conclusion as to how many fingers are held up.

The test subjects included students, doctors, teachers, a college professor, administrative assistants, and an economist. Test subjects were in their twenties or thirties. There was a wide range in the length of computer experience among test subjects. One test subject had less than one year of computer experience, while two reported 22 years of computer experience. There was also a wide range of weekly computer usage by test subjects, from less than one hour a day to 10 hours per day.

Five tests were conducted in a room with ambient natural light and incandescent interior lighting. One test was conducted in a laboratory with incandescent and fluorescent lighting. Sixteen tests were conducted in offices at a university, where fluorescent lighting supplemented the ambient light coming through windows. In all locations, the camera pointed at a ceiling about two meters overhead.

TABLE III

CONFUSION MATRIX. EMPTY ENTRIES INDICATE 0%. TWENTY TEST SUBJECTS EACH MADE TWO VERSIONS OF EACH HAND SIGNAL.

Ground Truth	*Finger Counter*'s Estimate				
	100.0%				
		100.0%			
		2.5%	97.5%		
			15.0%	82.5%	2.5%
				15.0%	85.0%

Before the test began, the subjects were given a brief introduction to the *Finger Counter*: how it works, its intended purpose, the types of tests they would complete, and a demonstration on how to use it. They were then given the opportunity to familiarize themselves with the program by playing with the painting program or else trying a few signals with the voice-interactive game before the formal testing began. During this trial period, the only advice given to the participants was to keep their hand in the field of view of the camera, at a suitable distance so that all fingers could be seen clearly. Users were also asked to remove their hand from the field of view between tests. The demonstration and explanation took no more than two minutes.

The "voice-interactive test" is a version of the voice-interactive game described in Section V. The test administrator initiated each request by pressing a key on the keyboard. Each test subject was asked to make ten hand signals, two of each hand signal; the sequence of requests was generated ahead of time at random and the same sequence was used for all test subjects. Following a request, a user was given up to ten seconds to make the hand signal or type the key on the keyboard. When the subjects used the keyboard, they always selected the correct number. When a hand signal was made and the system failed to identify it within ten seconds, it was assumed that the system was unlikely to identify it given more time. In fact, in the tests the longest response time was 6.77 s.

Within the ten-second window, the system logged frames until the requested signal was detected; on a few occasions, an incorrect signal was detected first and then the correct signal. Table 6.4 gives a confusion matrix, showing requests made ("ground truth") and the system's initial estimate of the hand signal. For hand signals and , there was no confusion, that is, the system correctly identified the hand signal made on the first try 100% of the time. *Finger Counter* misidentified as on one occasion out of 40, or 2.5% of the time. For signals and , there were higher confusion rates. Once *Finger Counter* misidentified as , and *Finger Counter* misidentified as five times, or 15% of the time, in both cases out of a total of 40 requests to make hand signal . Finally, five times out of 40, *Finger Counter* misidentified as .

Table IV shows minimum, median, and maximum response times broken down by signals requested and recognized. Over all signals, the median time between when a user was asked to form a particular signal and when the *Finger Counter* system recognized it is 2.33 s. The mean response time is 2.38 s and the 95%-confidence interval, computed using Student's t distribution, was [2.04 s, 2.73 s].

TABLE IV

RESPONSE TIMES DURING VOICE-INTERACTIVE TEST

Hand Signal	Minimum	Median	Maximum
	1.40 s	2.21 s	3.34 s
	1.37 s	2.33 s	5.34 s
	1.57 s	2.44 s	3.72 s
	1.50 s	2.34 s	4.21 s
	1.49 s	2.37 s	6.77 s

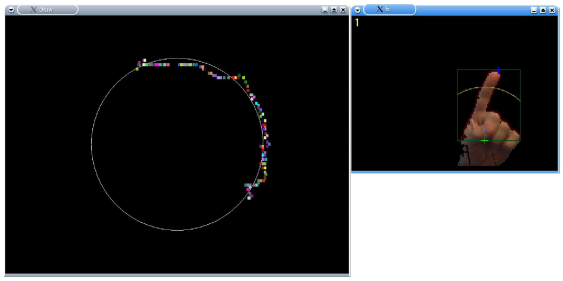

Fig. 12. *Finger Counter* "fi nger paint" test: the left screen shows the drawing; the right screen shows the "user feedback" window.

These numbers are an improvement on test results on an earlier version of the *Finger Counter* [8], [9]. In those tests, involving 37 test subjects, correct recognition rates ranged from 68% to 97%, depending upon the hand signal. The mean response time was 2.96 s with 95%-confidence interval [2.69 s, 3.24 s]. By comparison, the time it took the same test subjects to respond to a request to press a particular key on the keyboard, starting with the hand away from the keyboard, was also measured—the mean of 2.24 s, 95%-confidence interval [2.04 s, 2.43 s], was 0.72 s shorter than the mean response time using the earlier version of the system. Note that the difference between the keyboard response times and the current *Finger Counter* interface response times is statistically insignificant: the keyboard's mean response time 2.24 s is within the *Finger Counter*'s 95%-confidence interval [2.04 s, 2.73 s] and vice versa (2.38 s is in [2.04 s, 2.43 s]).

C. Experiments with "finger paint" application

As described in Section V, the "finger paint" application allows a user to "paint" on the screen by moving one or more fingertips in front of the camera. For the third evaluation tool, instead of beginning with a blank canvas, the program presented users with a circle on the screen as a template to draw on. The "finger paint" test was conducted immediately after the voice-interactive test with five of the volunteers. After the program was started, users were asked to move their finger to position the brush on the template. Then, the test administrator began the test by pressing a key on the keyboard. Once the user made one full circumnavigation of the template, the test administrator pressed another key to stop the test. Each position of the brush was logged by the program. For comparison, the test was repeated with the test subject using a computer mouse, instead of the *Finger Counter* interface, with the same test protocol.

To evaluate the accuracy of a subject's trace of the circular template, the average distance D =

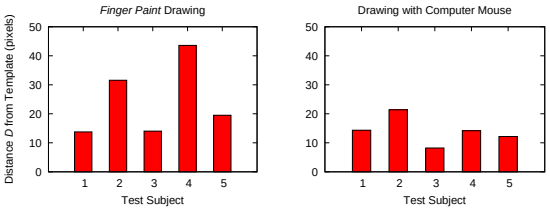

Fig. 13. Average distances between test subjects' drawings and the circular template

$\frac{1}{N}\sum_{i=1}^{N}|(||\mathbf{p}_i - \mathbf{O}|| - r)|$ was computed from each point $\mathbf{p}_i$ in a given drawing to the closest point in the template, where $\mathbf{O}$ is the center of the template, r is the template radius, and N is the number of points drawn. Figure 13 shows results of the "finger paint" test. For drawings of the five subjects with the *Finger Counter* interface, D ranged from 13.8 pixel units to 43.6 pixel units on an 800×600-pixel draw window containing a circle of radius 200 pixel units. The median of D was 19.5 pixel units. By comparison, for drawings with a computer mouse, D ranged from 8.2 to 21.4 pixel units, with an median of 14.2 pixel units.

D. Qualitative evaluation of usability

The fourth evaluation tool, the questionnaire, asked test subjects to rate on a scale from 1 to 10 the (1) ease of use and (2) "naturalness" of the *Finger Counter* interface versus an ordinary computer mouse. For instance, a 10 on "ease of use" means that the respondent thought the interface was "super easy" to use; a 1, "very hard." A 10 on the "naturalness" test means the respondent thought the pointing device was "completely intuitive;" a 1 means it was "completely unnatural." Twelve test subjects responded to the questionnaire. Figure 14 shows histograms of the results. The questionnaire responses show similar ranges for "ease of use." For "naturalness," responses for the computer mouse are noticeably higher than for the *Finger Counter*. Some respondents commented that "naturalness" was application-dependent, and thus it was difficult to rate.

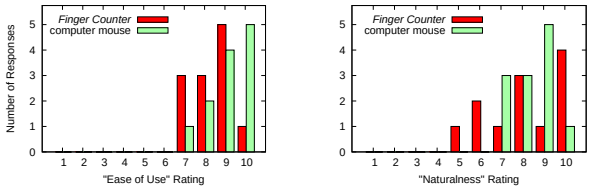

Fig. 14. Histograms showing ratings of 12 respondents for "ease of use" and "naturalness" of a computer mouse versus *Finger Counter*

VII. Discussion

The *Finger Counter* extends prior work to produce a simple and reliable hand-signal recognizer. There are two main algorithmic contributions. Firstly, a background modeling/alignment algorithm is proposed that can compensate for small motions of the camera. While the algorithm is based on subtracting subsequent frames, a popular method in computer vision, it does more than that: it captures a "foreground-free subimage" prior to and automatically during interface use to model the background dynamically. Once the foreground, in our application the user's hand, appears, the algorithm estimates the best subimage alignment for each frame, employing pixel-wise comparisons in the three color channels with an automatically computed threshold. Secondly, a rules-based protrusion estimator is used for the reliable detection of fingers given the radial representation of the hand boundary. Our contribution here is a fresh approach to combine similar techniques into a system that works under less restrictive conditions than some of the previously published approaches. In particular, high-resolution visible-light or thermal cameras, stable tripods, finger markers, uniform backgrounds etc. are not required.

Our experiments show that the interface reliably and quickly recognizes hand signals as input and can be used in application programs, such as the voice-interactive game and "finger paint" application. The experiments were conducted in a variety of real-world situations in offices, laboratories, people's homes, coffee shops – in natural light and in incandescent and fluorescent lighting. The system was tested by people with various levels for computer experience – children, students, and professionals.

One of the *Finger Counter*'s strengths is that it works for any user without training. This is why the system uses a rules-based approach, assuming that most human hands are capable of forming the contours recognized by the system. Indeed, anthropomorphic studies show relatively little variation in hand length or width [6], [19]. Untrained gesture-recognition systems [10]–[12], [16], [17], [28], [29], [34], [35], [47] in the literature did not include quantitative experimental results. The *Finger Counter*'s successful recognition rates were comparable with hand-gesture recognition systems [32], [37], [49], [51], [56], [57] that required training. Future studies may show whether other shape [62] or color-space representations [54] can lead to more successful hand signal detection.

Experiments detailed in Section VI show that the *Finger Counter* is robust to variations in hand position with respect to the camera. Although the interface is designed to recognize hand postures from hands held parallel to the image plane, in fact the rules-based protrusion estimator allows for substantial variation in how a user can hold his or her hand. Using the most conservative criteria, that is, the narrowest of the ranges of motion of the test subjects, the system still tolerates a wide range of motion. Note that the *Finger Counter* users are willing to cooperate with the system requirements, and do not purposely try to make the system fail. This cooperation also ensures that the *Finger Counter* system can initialize and maintain dynamically the background image without the user's hand in the field of view.

Tests show the *Finger Counter* interface to be reliable and efficient. The interface is only a little slower than a keyboard for making a selection from five items. This was consistent with the observation that, in most cases, response times reflected how long it took the user to process and respond to the request; once the hand signal was formed,

the system recognized it quickly. Because response times are comparable to a keyboard, users might be expected to find the *Finger Counter* an adequate substitute for tasks where input consists of the selection of a small number of items, such as from a menu.

Tests using the "finger paint" application showed that the *Finger Counter* is not as accurate as other pointing devices that require direct contact with the hand. In particular, the interface is adequate for rough drawing tasks, but, for fine pointer control, the computer mouse was shown to be superior. Note also that the test subjects were very familiar using a mouse, but were exposed to the *Finger Counter* for only a few minutes. With more practice, users may find the *Finger Counter* as natural or easy to use as a computer mouse, especially if they support their hand by a table.

The earlier system [8], [9] was less tolerant of changes in hand position than the current system. Moreover, correct recognition ranged from 68% to 97% in the earlier system compared with 83% to 100% in the current system. Finally, response times were significantly better. Apparently, correction for radial distortion and sensor noise, and substitution of a contour-following algorithm for an edge-detection algorithm in the Feature Extraction module, resulted in improved performance.

VIII. Conclusion

The *Finger Counter* is a viable human-computer interface using inexpensive computer and video equipment. It

- functions reliably with a "webcam" and tolerates minor camera motion,
- has minimal lighting and background requirements, and
- is easy to use.

The philosophy underlying the *Finger Counter* is that simple demands on the user can result in a viable human-computer interface. By requiring users to make a small number of hand poses and hold their hands approximately parallel to the image plane, the system is able to make reliable recognitions in real time. By compensating for small camera movements that shift the user's hand by only a few pixels in the image, the system is flexible enough to handle non-ideal camera setups.

In experiments, test subjects with minimal training achieved response times with *Finger Counter* comparable to a keyboard. This suggests that the system is a useful interface for selecting among a small number of menu items. As a pointing device, experience shows that the *Finger Counter* is less accurate than a computer mouse, but may be suitable for selecting relatively large on-screen objects and performing gross manipulations on them.

In some contexts, control by simple, intuitive hand signals may be a preferable input modality. A user may not want to use a keyboard or mouse, for example, because the user's hands are dirty (for example, an auto mechanic's) or sterile (for example, a surgeon's) or because one hand is occupied with another task. For these situations and others, the *Finger Counter* may prove a useful interface.

References

[1] Vassilis Athitsos and Stan Sclaroff. An appearance-based framework for 3D hand shape classification and camera viewpoint estimation. Technical Report BU-CS-TR-2001-022, Boston University, 2001. Shorter version published in *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 45–50, Washington, DC, May 2002.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[3] M. Betke and N. C. Makris. Recognition, resolution and complexity of objects subject to affine transformation. *Int J Comput Vis*, 44(1):5–40, August 2001.

[4] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *Proceedings of 5th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 63–74, Washington, D.C., USA, May 2002.

[5] U. Bröckl-Fox, L. Kettner, A. Klingert, and L. Kobbelt. Hand gesture recognition as a 3-D input technique. In N. Magnenat-Thalmann and D. Thalmann, editors, *Artificial Life and Virtual Reality*, pages 173–187, New York, NY, 1994. Wiley.

[6] B. Buchholz and T. J. Armstrong. An ellipsoidal representation of human hand anthropometry. *Hum Factors*, 33(4):429–441, 1991.

[7] F.-S. Chen, C.M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image Vision Comput*, 21(8):745–758, 2003.

[8] S. C. Crampton. Counting fingers in real time using computer-vision techniques. Master's thesis, Boston University Department of Computer Science, September 2004.

[9] S. C. Crampton and M. Betke. Counting fingers in real time: A webcam-based human-computer interface with game applications. In *Proceedings of Universal Access in Human-Computer Interaction Conference (UA-HCI)*, pages 1357–1361, Crete, Greece, June 2003.

[10] J. Crowley, F. Berard, and J. Coutaz. Finger tracking as an input device for augmented reality. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 195–200, Killington, VT, 1996.

[11] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 416–421, Nara, Japan, 1998.

[12] T. Darrell and A. Pentland. Space-time gestures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 335–340, New York, NY, 1993.

[13] S. M. Dominguez, T. Keaton, and A. H. Sayed. Robust finger tracking for wearable computer interfacing. In *Workshop on Perceptive User Interfaces (PUI)*, Orlando, FL, November 2001. ACM Digital Library. 8 pp.

[14] J. Fails and D. Olsen. A design tool for camera-based interaction. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 03)*, pages 449–456. ACM Press, 2003.

[15] F. Flórez, J. M. García, J. García, and A. Hernández. Hand gesture recognition following the dynamics of a topology-preserving network. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 318–323, Washington, D.C., USA, May 2002.

[16] W. T. Freeman, D. B. Anderson, P. A. Beardsley, C. N. Dodge, M. Roth, C. D. Weissman, W. S. Yerazunis, H. Kage, K. Kyuma, Y. Miyake, and K. Tanaka. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, 18(3):42–53, May/June 1998.

[17] W. T. Freeman, P. A. Beardsley, H. Kage, K. Tanaka, C. Kyuman, and C. Weissman. Computer vision for computer interaction. *SIGGRAPH Computer Graphics*, 33(4):65–68, November 2000.

[18] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the European Conference on Computational Learning Theory*, pages 23–37, 1995.

[19] J. W. Garrett. The adult human hand: Some anthropometric and biomechanical considerations. *Hum Factors*, 13(2):117–131, 1971.

[20] T. Grossman, D. Wigdor, and R. Balakrishnan. Multi-finger gestural interaction with 3D volumetric displays. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST 04)*, pages 61–70. ACM Press, 2004.

[21] D. Guo, Y. Yan, and M. Xie. Vision-guided human-vehicle interaction through hand sign understanding. In *Proceedings of the Fifth International Conference on Control, Automation, and Robotics and Vision*, volume 1, pages 151–155, Singapore, December 1998.

[22] L. Gupta and S. Ma. Gesture-based interaction and communication: Automatic classification of hand gesture contours. *IEEE Trans Syst Man Cyb*, 31(1):114–120, February 2001.

[23] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000. Page 191.

[24] E.-J. Holden and R. Owens. Representing the finger-only topology for hand shape recognition. *Machine Graphics and Vision International Journal*, 12(2):187–202, 2003.

[25] R. Jain, R. Kasturi, and B. Schunk. *Machine Vision*. McGraw Hill, 1995. Pages 46 and 47.

[26] R. Jain, R. Kasturi, and B. Schunk. *Machine Vision*. McGraw Hill, 1995. Page 50.

[27] C. Jennings. Robust finger tracking with multiple cameras. In *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 152–160, Corfu, Greece, September 1999.

[28] E.-M. Ji, H.-S. Yoon, and Y. Bae. Touring into the picture using hand shape recognition. In *Proceedings of the Eighth ACM International Conference on Multimedia*, pages 388–390, Los Angeles, CA, USA, November 2000.

[29] K.-H. Jo, Y. Kuno, and Y. Shirai. Manipulative hand gesture recognition using task knowledge for human computer interaction. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 468–473, Nara, Japan, 1998.

[30] A. Kendon. Current issues in the study of gesture. In J. L. Nespoulous, P. Peron, and A. R. Lecours, editors, *The Biological Foundations of Gestures: Motor and Semiotic Aspects*, pages 23–47. Lawrence Erlbaum Assoc., 1986.

[31] A. Kendon. An agenda for gesture studies. *The Semiotic Review of Books*, 7(3):8–12, 1996.

[32] R. Kjeldsen and J. Kender. Toward the use of gesture in traditional user interfaces. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 151–156, Killington, VT, 1996.

[33] M. W. Krueger. *Artificial Reality*. Addison-Wesley, 1991.

[34] S. Kumar and J. Segen. Gesture based 3D man-mach interaction using a single camera. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 630–635, June 1999.

[35] I. Laptev and T. Lindeberg. Tracking of multi-state hand models using particle filtering and hierarchy of multi-scale image features. In M. Kerckhove, editor, *Lecture Notes in Computer Science. Vol. 2106: Proceedings of the IEEE Workshop on Scale-Space and Morphology*, pages 63–74. Springer-Verlag, July 2001.

[36] J. Letessier and F. Bérard. Visual tracking of bare fingers for interactive surfaces. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST 04)*, pages 119–122. ACM Press, 2004.

[37] R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *Proceedings, British Machine Vision Conference*, pages 817–826, Cardiff, UK, 2002.

[38] S. Malik and J. Laszlo. Visual touchpad: a two-handed gestural input device. In *Proceedings of the 6th International Conference on Multimodal Interfaces (ICMI 04)*, pages 289–296. ACM Press, 2004.

[39] D. McNeill. *Hand and mind*. University of Chicago Press, 1992.

[40] D. McNeill. Introduction. In D. McNeill, editor, *Language and gesture*, pages 1–7. Cambridge University Press, 2000.

[41] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 374–380, Fort Collins, CO, USA, 1999.

[42] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 237–242, New Orleans, LA, 1991.

[43] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, November/December 2002.

[44] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans Pattern Anal Machine Intelligence*, 19(7):677–695, 1997.

[45] I. Poddar, Y. Sethi, E. Ozyildiz, and R. Sharma. Toward natural gestures/speech HCI: A case study of weather narration. In *Proceedings of the Workshop on Perceptual User Interfaces (PUI98)*, pages 1–6, San Fransisco, CA, November 1998.

[46] M. Porta. Vision-based interfaces: methods and applications. *Int J Hum-Comput St*, 57:27–73, 2002.

[47] F. Quek. Unencumbered gestural interaction. *IEEE Multimedia*, 3(4):36–47, 1997.

[48] F. Quek, T. Mysliwiec, and M. Zhao. Fingermouse: A freehand pointing interface. In *Proceedings of the First IEEE International Conference on Automatic Face and Gesture Recognition*, pages 372–377, Zurich, Switzerland, 1995.

[49] A. Shamaie and A. Sutherland. Accurate recognition of large number of hand gestures. In *Proceedings of the Second Iranian Conference on Machine Vision and Image Processing*, pages 308–317, Tehran, Iran, February 2003.

[50] A. Silberschatz and P. Galvin. *Operating system concepts*. Wiley, 1997.

[51] T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Trans Pattern Anal Machine Intelligence*, 20(12):1371–1375, 1998.

[52] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 246–252, Fort Collins, CO, USA, 1999.

[53] T. Takahashi and F. Kishino. A hand gesture recognition method and its applications. *Syst Comput Jpn*, 23(3):38–48, 1991.

[54] J.-C. Terrillon and S. Akamatsu. Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 54–61, Grenoble, France, March 2000.

[55] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proceedings of the Seventh International Conference on Computer Vision (ICCV)*, pages 255–261, Kerkyra, Greece, September 1999. IEEE Computer Society.

[56] J. Triesch and C. von der Malsburg. Robust hand posture classification against complex backgrounds. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 170–175, Killington, VT, USA, 1996.

[57] J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Trans Pattern Anal Machine Intelligence*, 23(12):1449–1453, 2001.

[58] C. Vogler and D. Metaxas. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 363–369, Mumbai, India, January 1998. IEEE Computer Society.

[59] A. D. Wilson and A. F. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Trans Pattern Anal Machine Intelligence*, 21(9):884–900, 1999.

[60] Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. In *Lecture Notes in Artificial Intelligence 1739, Gesture-Based Communication in Human-Computer Interaction*, volume 1739, pages 93–104. Springer, 1999.

[61] M.-H. Yang and N. Ahuja. Extracting gestural motion trajectories. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 10–15, Nara, Japan, 1998.

[62] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37:1–19, 2004.

[63] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer. Visual Panel: Virtual mouse, keyboard and 3D controller with an ordinary piece of paper. In *Workshop on Perceptive User Interfaces (PUI)*, Orlando, FL, November 2001. ACM Digital Library. 5 pp.

[64] H. Zhou and T. S. Huang. A Bayesian framework for real-time 3D hand tracking in high clutter background. In *Proceedings of the Universal Access in Human-Computer Interaction Conference*, pages 1303–1307, Crete, Greece, June 2003.

Stephen C. Crampton is a computer-science teacher at the Dwight-Englewood School in Englewood, NJ. Under the Dwight-Englewood School's Mathematics, Science, and Technology Program, computer science is part of the core curriculum. Mr. Crampton received his Master's degree in 2004 from Boston University, where he was a member of the Image and Video Computing Research Group. In 2003, he won the Chancellor's Award at the Boston University Science and Technology Day for "Counting Fingers in Real Time: A Webcam-based Human-Computer Interface with Game Applications." Mr. Crampton is a 1986 graduate of Middlebury College and a 1989 graduate of the George Washington University National Law Center, where he was a member of the *George Washington Law Review* and winner of the Van Vleck Constitutional Law Moot Court Competition. Mr. Crampton has published in the area of human-computer interaction.

Margrit Betke received her Ph.D. and S.M. degrees in Computer Science and Electrical Engineering from the Massachusetts Institute of Technology in 1995 and 1992, respectively. She is an Associate Professor in the Computer Science Department at Boston University, where she co-leads the Image and Video Computing Research Group, and a Research Scientist at the Massachusetts General Hospital and Harvard Medical School. From 1995 to 1997 she was a postdoctoral researcher at the Institute for Advanced Computer Studies at the University of Maryland and was also affiliated with the Computer Vision Laboratory of the university's Center for Automation Research. ¿From 1997 to 2000, she was an Assistant Professor at Boston College. Dr. Betke received the National Science Foundation Faculty Early Career Development Award for "Video-based computer interfaces for people with severe disabilities." She is an honoree of the Mass High Tech's 2005 "Women to Watch Award." She has published in the areas of human-computer interfaces, medical image analysis, animal sensing, intelligent vehicles, machine learning, and robot navigation.

Stan Sclaroff received the S.M. degree and the Ph.D. degree from the Massachusetts Institute of Technology in 1991 and 1995, respectively. He is an Associate Professor in the Computer Science Department at Boston University, where he founded the Image and Video Computing Research Group in 1995. In 1996, he received a Young Investigator Award from the US Office of Naval Research and a Faculty Early Career Development Award from the US National Science Foundation. From 1989 to 1994, he was a research assistant in the Vision and Modeling Group at the MIT Media Laboratory. Prior to that, he worked as a senior software engineer in the solids modeling and computer graphics groups at Schlumberger Technologies, CAD/CAM Division. Dr. Sclaroff has coauthored more than 80 publications in the areas of deformable shape modeling, image and video database retrieval, image segmentation, human tracking, and computer animation. He served as an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* from 2000 to 2004. He is a member of the IEEE.