BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Thesis

COUNTING FINGERS IN REAL TIME USING COMPUTER-VISION TECHNIQUES

by

STEPHEN C. CRAMPTON

B.A., Middlebury College, 1986

Submitted in partial fulfillment of the

requirements for the degree of

Master of Arts

2004

Approved by

First Reader

Margrit Betke, Ph.D.
Assistant Professor of Computer Science
Boston University

Second Reader

James Gips, Ph.D.
John R. and Pamela Egan Professor of Computer Science
Boston College

Third Reader

Stan Sclaroff, Ph.D.
Associate Professor of Computer Science
Boston University

ACKNOWLEDGEMENTS

This thesis is based on joint work with Margrit Betke, Ph.D., of Boston University. Thank you, Dr. Betke, for your support and guidance. Without your efforts this document would not have been possible. Thank you to the other readers, Dr. James Gips and Dr. Stan Sclaroff. Thanks also to the other members of the Image and Video Computing Group for their ideas and feedback and for fostering a collegial research environment. A special thanks to Jessie Burger and Christen S. Ng for testing an early version of the *Finger Counter* and for their feedback on the testing protocols. Funding was provided by the National Science Foundation (IIS-0093367, IIS-0329009, and EIA-0202067) and the Office of Naval Research (N000014 -01-1-0444), and is gratefully acknowledged.

COUNTING FINGERS IN REAL TIME USING COMPUTER-VISION TECHNIQUES

STEPHEN C. CRAMPTON

ABSTRACT

A video-based human-computer interface called *Finger Counter* was developed to control a computer with a small set of hand signals. *Finger Counter* works in real time with an inexpensive, consumer-grade camera, such as a webcam. *Finger Counter*'s background-differencing algorithm compensates for camera noise and vibration to segment the hand from a cluttered background in settings with low-image quality or camera motion. *Finger Counter*'s rules-based protrusion estimator determines the number of fingers a user holds up in front of the camera. *Finger Counter* applications include a game designed to teach children to count with their fingers and a program that allows the user to "finger paint" on a computer screen. Experiments show that *Finger Counter* quickly and reliably detected the hand signals of test subjects.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hand gestures facilitate communication in many aspects of life. Hand gestures are a fundamental mode of communication, not only in humans, but in higher mammals such as primates as well [39]. They are such a fundamental form of human communication that babies who have not yet learned to speak can use hand signs to ask for food or otherwise communicate [3]. Moreover, people are accustomed to using hand signals to convey meaning in situations where it is difficult to communicate with speech. At a distance or underwater, where voice communication would be difficult, hand signals are used by scuba divers [4] and sea kayakers [56]. Hand gestures signal commands to crane operators; in this setting, unambiguous communication is necessary to ensure the safety of persons working with the crane payload [1]. The Chicago Mercantile Exchange has a glossary of hand signals for futures traders in its noisy trading pit: for example, to indicate that the last digit of a price is 3, a trader holds up three fingers [17]. The Army uses hand signals to communicate in hostile environments [47]. Figure 1 gives examples of hand gestures in the context of scuba diving, futures trading, crane operation, and military action.

The usefulness of hand gestures as a mode of communication motivates the design of computer interfaces controlled by hand gestures. Gestural interfaces are "high level," that is, they map directly to user intent, as compared with other interfaces

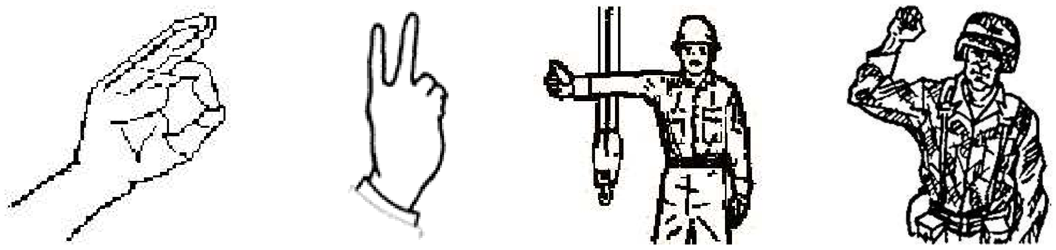


Figure 1.1: Sample hand signals. Shown from left to right are the scuba hand signal for “Are you OK?” [4]; the futures-trading signal for a contract expiring in the month of October [17]; the crane-operations signal for “hoist” [1]; and the Army signal for “freeze” [47].

that require a user to learn and attend to operational details, such as the particular layout of a keyboard [41]. Numerous computer systems have been developed to recognize hand gestures [51, 66, 49]. Virtually all such 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, and
- be easy and intuitive.

The *Finger Counter* is a vision-based interface that aims to meet these goals. Figure 1.2 shows a typical *Finger Counter* 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 *Finger Counter*

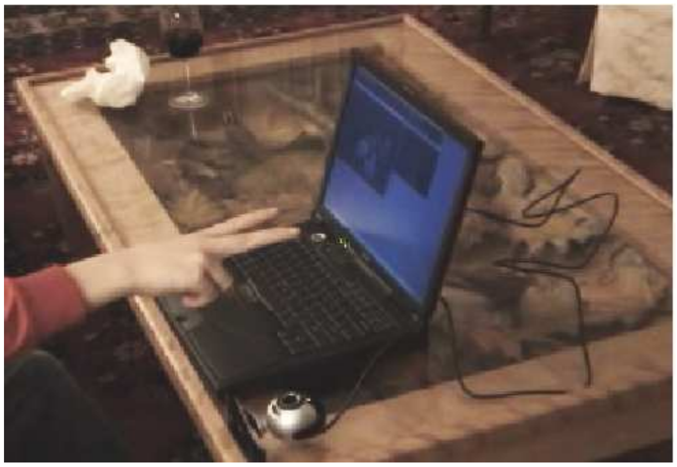


Figure 1.2: The *Finger Counter*. The webcam (on the table) points upward at the hand.

works without users wearing gloves, special lighting, or other equipment. It requires virtually no training on the part of the user, because the gestures it recognizes are simple and intuitive. The system tolerates minor camera motion, including vibration. 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.

Hand gestures can be described to consist of three parts: (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 [32, 42]. Preparation and retraction may be similar across many different gestures; it is the stroke that contains the communicative content [32]. The stroke may consist of a hand posture and a particular movement or it may be a static hand posture [42]. In the literature, hand gestures are divided into four general categories [42, 43, 33]: *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 *Finger Counter* 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 stream of colored pixels should be drawn corresponding to the fingertip location in the camera's image. Unlike emblems, however, the entire hand's motion in the camera's field of view can act as another channel of communication. In the "finger paint" application, for example, the sweep of a user's fingertip guides a virtual brush to draw on the screen. To distinguish these types of gestures from emblems, this thesis uses the term *hand signal* to characterize the types of gestures recognized by the *Finger Counter*. For purposes of this thesis, hand signals comprise a hand posture that sometimes is in motion.

To give an overview of how the *Finger Counter* works, 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. As detailed in Chapter 3, 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 spatial 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.

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 [52]. Thus, the *Finger Counter* requires the user to form a hand silhouette or contour that it can recognize. Forming a particular hand contour is something many people are familiar with, for instance, young children enjoy making silhouette animals with flashlights and finger poses. This familiarity tends to make the *Finger Counter* an intuitive interface for many people.

Applications described in Chapter 4 were developed to evaluate the *Finger Counter* interface: the “voice-interactive counting game” asks the user to hold up between one and five fingers and then tells the user how many fingers it recognizes.

The “finger paint” application allows the user to paint on a blank canvas by moving his or her fingers within the field of view of the camera; the size and number of paint “brushes” are dictated by the user’s hand posture.

In the next chapter, previous work is discussed and the *Finger Counter* is placed in its context. Chapter 3 explains the *Finger Counter*’s methods for detecting hand signals. Chapter 4 covers implementation details, and Chapter 5 describes two applications developed to use the *Finger Counter* interface. Results of tests designed to evaluate the *Finger Counter*’s efficacy as an input device fill Chapter 6. Chapter 7 discusses the test results and the *Finger Counter*’s limitations and potential benefits to users. With Chapter 8, this thesis concludes.

Chapter 2

Related work

Because hand gestures are such a natural and intuitive means of communication, many researchers have developed systems to recognize hand signals as a means of computer input [51, 66, 49]. 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 portion of the image corresponding to the hand from the background; *extraction of a feature vector,* containing parameters relevant to a classifier, from the hand image; and *classification* of the feature vector into a category corresponding to a particular hand gesture. Also, it is common to develop application programs to illustrate or evaluate the efficacy of a gesture-recognition system. This chapter describes how different researchers have solved the segmentation, feature extraction, and classification challenges; some of the applications designed to work with gesture-recognition systems; and how the *Finger Counter* relates to the previous work.

2.1 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. Researchers approach this problem in a number of different ways. Many systems assume that the background is uniform and neutral, so that the hand is easily segmented by thresholding intensity values [65, 5, 12, 19]. A common approach is to have a user hold his or her hand over a black matte surface, such as a black cloth [37]. Sometimes, researchers employ special lighting to improve segmentation results [36]. Another common technique is to classify pixels according to their similarity to skin color [53, 40, 58, 34, 68, 24], or proximity to other skin-colored pixels [58], though segmentation using skin color can suffer from the fact that skin can appear different under different lighting conditions.

Another technique for hand segmentation compares successive images in a video stream to detect motion [13]. Oftentimes, the resulting "motion images," or processed versions of them, are fed directly to a classifier [20, 21]; in this case, segmentation is implicit. Some systems combine skin-color and motion-detection methods [67, 50]. Still other systems require the user to wear special gloves [31, 46, 61] or wrist bands [40] to help the system identify the hand. Some 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 [48]. Other systems use multiple cameras to determine the 3D position of a hand and threshold a composite image by distance, assuming the hand is closer to the camera than the background [29]. Some researchers, focusing on feature detection and classification, use images where the foreground is already segmented, often by hand [54].

The *Finger Counter* falls into the category of systems that use background subtraction to segment the hand from the background: in those systems, a model of the background image, without the hand in it, is stored and then subtracted from subsequent images including the hand. For instance, Stauffer and Grimson describe a method of "adaptive" background subtraction, where the background is modeled, pixel-by-pixel, as a mixture of Gaussians [59]. As the background changes, due to gradual lighting changes or objects, like shadows, moving slowly in or out of frame, the multi-modal Gaussian models are updated. Foreground pixels are classified as those that have low probability under the model. Other pixels, classified as background pixels, are used to update their respective models. The approach of updating the background model based upon pixels deemed part of the background is similar to the *Finger Counter*'s background-differencing method. The difference is that systems like Stauffer and Grimson's assume a stable camera with a slowly changing background while the *Finger Counter* system assumes a stable background with a slightly moving camera.

Like the *Finger Counter,* systems designed to compensate for camera motion tend to make simplifying assumptions related to their problem domain. Challenging domains are aerial surveillance and autonomous vehicles where cameras undergo significant self-motion. For example, Duric and Rosenfeld describe a method for smoothing video taken from a camera mounted on a vehicle [15]. They assume that the vehicle's translational and yaw motion is smooth and the only undesired impulsive movements are the vehicle's pitch and roll. Duric and Rosenfeld also assume that the image is of a static outdoor scene with the horizon visible; in fact, their system uses the horizon as a landmark to estimate pitch and roll.

Medioni et al. report a system to adjust for camera motion in aerial photography using principles used in image mosaicking [44]. Medioni et al.'s system assumes that the ground is a planar surface, which simplifies the estimation. As in image mosaicking, the problem is, given two images where the camera has moved between acquisition of them, determine the homography, or transformation, which best maps between the first and second images. A typical approach in mosaicking is to estimate the homography using a gradient descent approach such as the Levenberg-Marquardt method [60]. Medioni et al.'s refinement to the approach is, instead of pixelwise registration, to divide the image into grids and find within each grid the maximum in the intensity gradient. Using the random-sample consensus algorithm [18] to discard outliers, the remaining features can be registered using much less computation than is required for pixelwise mosaicking.

2.2 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 instance, to produce their features, Kjeldsen and Kender convert the collection of pixels representing skin color to grayscale with histogram equalization [34]. Other systems represent the hand pixels as "one" pixels in a binary image, where background pixels are "zeros;" the binary image is then directly fed to the classifier [40]. For example, Krueger's systems analyze what he terms the "silhouette image" of the hand to determine hand pose and position [36]. Jo et al.'s system recognizes two-hand gestures from hand silhouette images [31]. And, Laptev and Lindeberg's system applies linear transformations, extracting features at different scales by convolving the image with Gaussian kernels of different variances and then searching for local maxima of the normalized squared Laplacian of the result [38].

Other systems extract edges, defined as local maxima in the intensity gradient, for use as features [52]. For example, Quek's system recognizes dynamic gestures using what he terms a "moving edge detector," a module that estimates the first two partial derivatives of image intensity using the Sobel operator, and then thresholds the result. Freeman et al. report a technique using orientation histograms, which tally the discretized intensity gradient directions at each pixel location in the hand image [20, 21].

The *Finger Counter* 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 systems process the contour before attempting classification. For example, Gupta and Ma devise "localized contour sequences" in which each contour pixel is described in terms of its distance from a chord running between pixels before and after it in the sequence of contour pixels [25]. Chen et al. characterize hand contours with a Fourier descriptor, a compact representation of the spatial Fourier series derived from the contour pixels [10]. One advantage of their technique is that the descriptor is scale invariant.

Like *Finger Counter*, some 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 [5]. 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 [37]. Quek et al. use a finite-state machine to determine the position and length of finger-like protrusions [53].

Finger Counter 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 minor rotations 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-coordinates representation of the hand contour was also exploited by Bröckl-Fox et al. [7, 57]. From the polar-coordinate representation of a hand contour, Bröckl-Fox et al. determine a "signature," or one-dimensional functional representation of the boundary, which becomes their feature vector.

2.3 Classification

The third aspect of hand-gesture recognition is classification of the feature vector into one of several categories. In this problem domain, a category represents a particular hand pose. To facilitate comparison, some researchers create hard-wired models of the hand, that is, models 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 [38]. Models with one or more outstretched fingers are compared with blobs and ridges extracted from the image using convolution.

Sometimes the hard-wired models are supplemented with training. For instance, 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 [62, 63]. 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 92.9% of the time against a simple background and 85.8% of the time against a complex background. They ran their system on 604 test images against a uniform light or dark background and 338 images against complex backgrounds.

Flórez et al. use a graph-based technique employing a learning model called "growing neural gas" [19]. Nodes of the graph are defined by their (x, y) image coordinates. The iterative algorithm results in nodes being moved toward pixel locations that are surrounded by a large number of other pixels determined to be part of the hand image in the segmentation stage of the algorithm. Further constraints on the graph topology result in a graph with equally spaced nodes filling the hand region of the image. Against a black background, Flórez et al. [19] report gesture-detection success rates ranging from 90-100%, depending upon the particular gesture. Flórez et al. had a single user train the system with 5 samples each of 12 gestures. The test set consisted of the same user making 20 samples of each gesture.

Other systems also must be trained with sample images of the hand in different postures. In Athitsos and Sclaroff's system [5], 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 [5]. 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 [40]. Direct comparisons between binary pixels in silhouettes and those in templates are computationally expensive, so Lockton and Fitzgibbon first cluster the templates and then use a series of weak classifiers to quickly make the matches using deterministic boosting. Using a technique similar to adaptive boosting [22], the system finds pixels that are good discriminants, that is, knowing their value (either “zero” or “one”) allows the system to remove a substantial portion of the training templates from consideration. By successive selections of such pixels, the best match can be obtained in time logarithmic to the number of templates. 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. Lockton and Fitzgibbon report a 99.87% success rate, defined as one minus the percentage of false positives. This was out of 3,000 gesture images collected from a video stream of a single test subject using the system for 10 minutes. Lockton and Fitzgibbon did not report a false-negative rate. Bröckl-Fox et al. use correlation to compare their feature vectors, “signatures” derived from a polar-coordinate representation of the hand contour, with stored “signatures” derived from training data of hand postures to be recognized by the system [7, 57].

Other classification methods that require training include neural nets and hidden Markov models. Kjeldsen and Kender convert the collection of pixels representing skin color to grayscale with histogram equalization [34]. The grayscale image is scaled to fixed resolution and then used as input to a neural net. The neural net outputs a determination of hand pose. Kjeldsen and Kender's experiment for pose recognition used 50 samples of three hand poses for training. They achieved correct pose-recognition rates ranging from 85-90% of the time, using 100 test images. The recognition rate varied depending upon the particular images selected for training or testing.

Hidden Markov models, used extensively in speech recognition systems, model sequences of different states over time. As such, they are suitable for recognition of dynamic hand gestures [58, 64, 65, 10]. Starner et al. extract from the hand silhouette sixteen geometrical measurements, based upon various moments of inertia [58]. Their system uses them as inputs to a hidden Markov model; the model is tuned to recognize a limited vocabulary of dynamic hand gestures. Starner et al. tested their system on a series of sentences, consisting of five hand poses, with the same pose for the first and fifth pose. They had two experimental setups. For the first setup, with a camera in front of and slightly higher than the user, they used 384 test sentences for training and 94 for testing. Restricting their grammar to "pronoun—verb—noun—adjective—(same) pronoun," and restricting the location of the hand pose to approximately the same position in front of the camera for testing and training, Starner et al.'s system correctly recognized words in sentences defined by their grammar 91.9% of the time. With an unrestricted grammar, and allowing the hand pose to be made in different places in the field of view, the system achieved a 74.5% correct recognition rate. With the second experimental setup, a camera placed on the user's hat looking down at the hand, corresponding accuracy rates were 97.8% for restricted-grammar sentences and 96.8% for unrestricted-grammar sentences. They used 400 sentences for training and 100 for testing. The authors did not indicate whether or not the poses were made in the same location in the field of view for the second experiment.

Wilson and Bobick also use hidden Markov models for dynamic gesture recognition [65]. In their case, the input to the models are the 3D positions of both hands, as determined by a stereoscopic camera setup against a uniform background. Their system recognized two pointing gestures without error; the goal of their system was to minimize the error of the estimated pointing direction. Shamaie and Sutherland report a system to recognize dynamic hand gestures using hidden Markov models [54]. The system projects a low-resolution grayscale image of the hand in each frame into an eigenspace using principal components analysis. The projections at each frame are compared to a training set using a hidden Markov model for recognition. Shamaie and Sutherland report an 89.6% recognition rate of 100 dynamic gestures. They used 500 gesture samples for training and 500 for testing.

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 [13]. Kumar and Segan classify a hand gesture based upon the number of maxima and minima protrusions [37]. Jo et al.'s system determines the hand area, centroid and maximum distance between the centroid and any other point in the hand, that is, the length of the maximum protrusion [31]. The system recognizes dynamic gestures, such as "grasp," by analyzing a series of frames for differences in the area, centroid position, and maximum protrusion length. 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 [52]. Ji et al. use a row-by-row scanning algorithm to count how many fingers are held up [30]. Such a system depends upon the finger-like protrusions being strictly vertical. The system recognizes zero, one, or two fingers.

One of several systems reported by Freeman et al. uses geometrical information without classifying it into discrete categories [20, 21]. The system computes the first and second moments of least inertia of the hand image and uses that information to control the speed and direction of a mobile robot.

Some systems eschew the segmentation and feature-extraction stages, making a classification directly from the video image. For example, Darrell and Pentland use normalized correlation to match view models of particular hand postures with the image [14]. The disadvantage of such a technique is that a new model must be created for each view. Along the same lines, another system reported by Freeman et al. matches a template of the hand with all five fingers outstretched with the image, again using normalized correlation [20, 21]. Crowley et al.'s system likewise uses normalized correlation to track a pointer device [12]

2.4 Applications

Researchers have developed or proposed a wide variety of applications for hand-gesture recognition interfaces. Krueger describes hand-gesture input devices for such applications as finger drawing, painting, and a "videodesk," which uses hand gestures to control word-processing functions, such as selecting text [36]. Kjeldsen and Kender's system uses hand poses and motion features, such as the direction of motion or the lack of motion, to form a Haptic grammar, which allows the user to communicate commands to a user interface, such as, "move object" or "iconize window" [34]. Kohler developed a hand-gesture recognition system, for which he proposes the following applications: computer-aided design, visualization, control of a video projector, sterile switches in medical operations, selection of display monitors, steering animated cartoons, and control of devices for the elderly [35].

Freeman et al. developed a number of different video-based systems for human-computer interaction [20, 21]. One system controls the direction in which a small robot moves from how a user moves his or her hand. Another system plays the rock/scissors/paper game against the user. Yet another system allows the user to control a television set with hand gestures. Kumar and Segen's applications include virtual flight control and a 3D graphical editor [37]. Crowley et al. developed a "FingerPaint" application, similar to *Finger Counter*'s "finger paint" application, that allows the user to draw with a single pointer device [12].

2.5 The *Finger Counter* in context

The *Finger Counter* identifies hand pixels in the image through background subtraction. What differentiates the *Finger Counter* from previous approaches is that the *Finger Counter* is designed to compensate 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 distinguished from systems in the literature that assume either a static background or else attempt to smooth a video sequence obtained from an aircraft or motor vehicle. The *Finger Counter*'s segmentation system allows the interface to work in real time against a cluttered background.

The *Finger Counter*, like some of the systems described above, uses the contour of the hand image to determine hand pose. What distinguishes the *Finger Counter* from previous work is the *Finger Counter*'s feature-extraction technique for identifying finger-like protrusions. As explained in detail in the next chapter, the *Finger*

Counter system converts the contour pixels from a Cartesian to a polar-coordinates representation, with the polar coordinates origin at the estimated position of the palm center. The system then searches for local maximum distances in sets of contour pixels corresponding to potential protrusions. This technique results in reliable protrusion estimation, as confirmed by experimental results.

An earlier version of the *Finger Counter* interface was reported by Crampton and Betke [11]. The earlier system 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. Also, the new system shows the background as well as the hand image in the "user feedback" window, which makes the interface easier to use. The Crampton and Betke article emphasized the human-computer interaction aspects of the system, while this thesis describes the computer-vision contributions of the system in detail.

Chapter 3

The method

The goal of any video-based computer system for interpreting hand signals is to detect 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 different when the hand is held in a different position with respect to the camera.

Finger Counter limits itself to a five-signal alphabet $\Sigma = \{\emptyset,$ ☝, ✌, 🤟, 🖖, 🖐$\}$. In addition, to simplify the detection, *Finger Counter* requires the user to keep his or her 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.

Finger Counter detects a hand signal $\hat{S}$ from a video image I using the three modules shown in Figure 3.1: (1) The "Image Processing" module takes a raw video image and processes it to determine the contour of the hand in the image. (2) From a single processed image, the "Feature Extraction" module detects how many fingers are held up and estimates 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

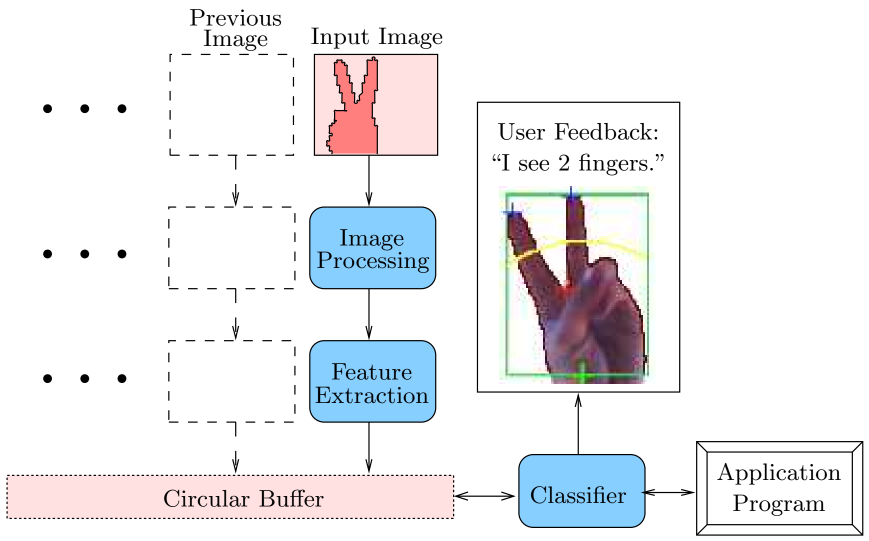


Figure 3.1: System overview

and location of fingertips over a series of frames and reach a final determination of the hand signal. The detected hand signal $\hat{S}$ 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 a detected 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 so the system can verify his or her intent. 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."

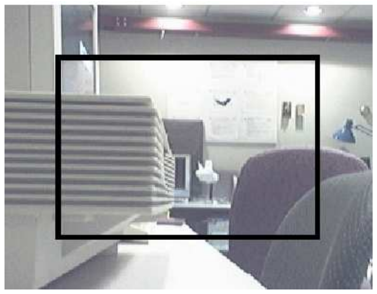
3.1 Image processing

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

3.1.1 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, $\mathbf{I}(i,j) = \left(I_t^R(i,j), I_t^G(i,j), I_t^B(i,j)\right)^\top$, with $I_t^R(i,j)$, $I_t^G(i,j)$, and $I_t^B(i,j)$ 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 $B_0^{\mathcal{C}}(i,j) + N^{\mathcal{C}}(0,\sigma)$, where $\mathcal{C}$ is the color channel and $N^{\mathcal{C}}(0,\sigma)$ represents a normal, zero-mean noise term of unknown standard deviation. This is analogous to an experimentally


(a) Background image $\mathbf{B}_0 = \mathbf{I}_0$ at time $t = 0$. Subimage $\mathbf{B}'_0 = \mathbf{I}'_0$ is inside the black rectangle.

(b) New subimage $\mathbf{I'}_t$ obtained at time t from camera's full frame I_t.

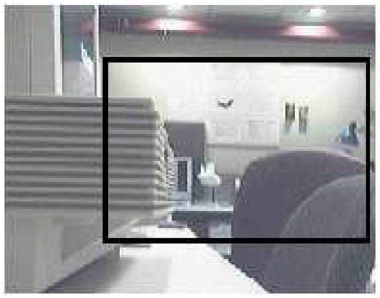

(c) Background image $\mathbf{B_0}$ with background subimage $\mathbf{B}'_t$ aligned to correspond with $\mathbf{I'}_t$ at time t.

Figure 3.2: Alignment of background image. Note that the figure exaggerates the displacement for illustration purposes.

validated model proposed for grayscale cameras [6]. 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 [45].

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, $\mathbf{I'}_t = \mathbf{I}_t\left(\mathbf{u}_0 + (i,j)\right)$. To perform background differencing at time t, the system compares $\mathbf{I'}_t$ with subimage $\mathbf{B'}_t = \mathbf{B}_0\left(\mathbf{u}_{t-1} + (i,j)\right)$, also of dimensions $M' \times N'$, as shown in Figure 3.2.

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,

$$F_t(i,j) = \begin{cases} 1 & \text{if, } \forall \mathcal{C} \in \{R,G,B\},\ \left|\mathbf{I'}^{\mathcal{C}}_t(i,j) - \mathbf{B'}^{\mathcal{C}}_t(i,j)\right| > \tau^{\mathcal{C}} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Threshold $\tau^{\mathcal{C}}$ is computed as $\tau^{\mathcal{C}} = \kappa \max_{(i,j)}\{\mathbf{I'}^{\mathcal{C}}_t(i,j) - \mathbf{B'}^{\mathcal{C}}_t(i,j)\}$, 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 $\mathbf{I'}^{\mathcal{C}}_t(i,j) - \mathbf{B'}^{\mathcal{C}}_t(i,j) > \tau'^{\mathcal{C}}$ 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, 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 Chapter 4.

To update offset $\mathbf{u}_t$, the algorithm minimizes the squared difference between background pixels in $\mathbf{I'}_t$ and the corresponding pixels in $\mathbf{B}_0$. Let $\mathcal{B}_t = \{(i,j) \mid F_t(i,j) = 0\}$ be the set of background pixel coordinates. To find $\mathbf{u}_{t+1}$, the algorithm computes the squared difference for each possible offset $\mathbf{u}$:

$$\hat{\mathbf{u}}_{t+1} = \underset{\mathbf{u}}{\operatorname{argmin}} \sum_{\substack{(i,j) \in \mathcal{B}_t \text{ and} \\ \mathcal{C} \in \{R,G,B\}}} \left(I_t'^{\mathcal{C}}(i,j) - B_0^{\mathcal{C}}\left(\mathbf{u} + (i,j)\right) \right)^2 . \tag{3.2}$$

The algorithm compensates for combinations of side-to-side and up-and-down camera motion, of the type sometimes encountered when a camera is perturbed or subject to vibration. Without loss of generality, consider translation parallel to the x-axis of the camera's image plane. In Figure 3.3, the world coordinate system is aligned with the camera so that the camera's optical axis is parallel to the world coordinate system's z axis and the image plane's x and y axes are parallel to the world coordinate system's x and y axes. In the figure, capital letters, for instance, (X, Y, Z_0), denote the coordinates of points in the camera's field of view, while lowercase letters indicate distances along the x or z axis. The distance between the camera's optical axis before and after the camera motion is b units. The camera's focal length is f units. Note that, if the origin of the camera's image plane coincides with the camera's principal point, where the optical axis intersects the image plane, then the distances x_1 and x_2 as shown in Figure 3.3 are also the x-coordinates of the respective image points in the image-plane coordinate system.

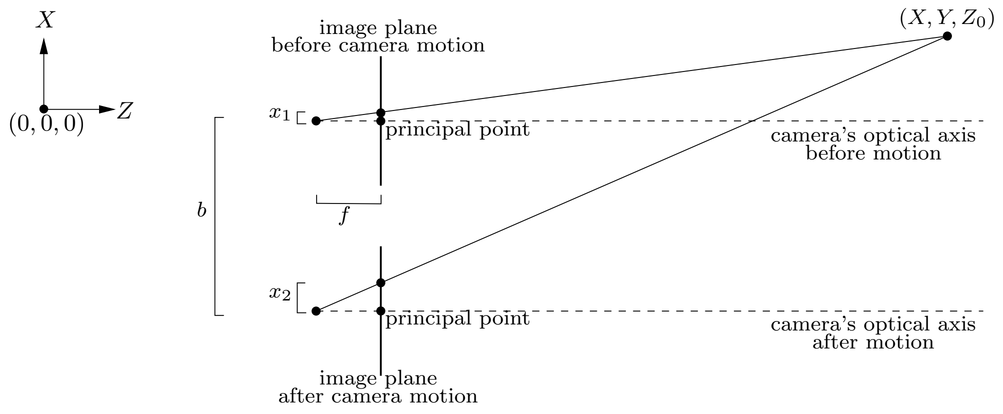


Figure 3.3: Illustration showing the effect of camera translation parallel to x-axis. The world coordinate origin is shown in the upper left part of the figure. The y-axis, which points upward from the page, is not shown. Capital letters, for example, (X, Y, Z_0), indicate coordinates of points in the camera's field of view while small letters, for instance, x_1 and b, denote scalar distances on the x-z plane. Before the camera is moved, world point (X, Y, Z_0) projects to the image plane at a point with distance x_1 from the camera's optical axis. After the camera is translated by b units parallel to the x-axis, the image of world point (X, Y, Z_0) is x_2 units from the camera's new optical axis. The difference $\delta_x = x_2 - x_1$ is called the "disparity."

Before the camera is moved, world point (X, Y, Z_0) is imaged at x_1 and afterward at x_2. The disparity δ_x in the x direction is expressed as follows:

$$\delta_x = x_2 - x_1 = \frac{bf}{Z_0} \tag{3.3}$$

Disparity δ_x is the ground truth for the x-component of *Finger Counter*'s estimate of $\hat{u}_x$ (Equation 3.2) 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 $b = 1$ 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:

$$\begin{aligned}
\delta_x &= \frac{bf}{Z_0} \\
&= \frac{(10 \times 10^{-3}\text{ m})(2.24 \times 10^{-3}\text{ m})}{2\text{ m}} \\
&= 11.2 \times 10^{-6}\text{ m} \\
&= 2\text{ pixel units}
\end{aligned}$$

If the background is not strictly planar, but contains an object at distance $X_0 - \Delta$, as illustrated in Figure 3.4, the disparity between its image before and after camera motion is also small:

$$\begin{aligned}
\delta' &= \frac{bf}{Z_0 - \Delta} \\
&= \frac{(10 \times 10^{-3}\text{ m})(2.24 \times 10^{-3}\text{ m})}{1.6\text{ m}} \\
&= 14 \times 10^{-6}\text{ m} \\
&= 2.5\text{ pixel units}
\end{aligned}$$

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

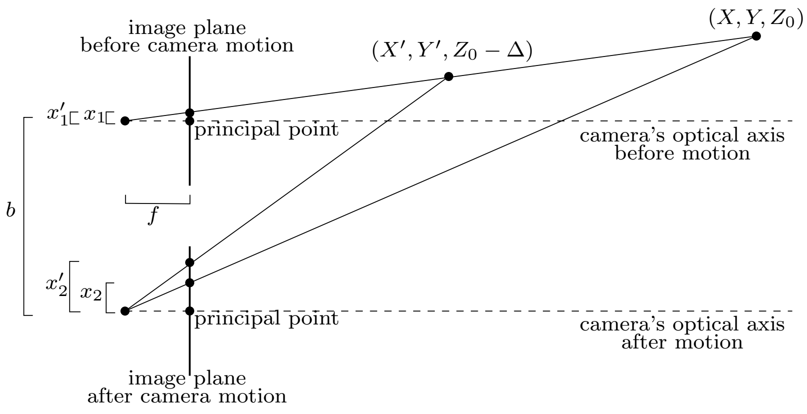


Figure 3.4: Illustration showing the effect of camera translation parallel to x-axis on two points with different z-coordinates. This figure depicts the same camera motion as in Figure 3.3. In this case, before the camera is moved, world points (X, Y, Z_0) and $(X', Y', Z_0 - \Delta)$ project to the image plane at the same point, with distance $x_1 = x'_1$ from the camera's optical axis. After the camera is translated by b units parallel to the x-axis, the image of world point (X, Y, Z_0) is x_2 units from the camera's new optical axis, while the image of $(X', Y', Z_0 - \Delta)$ is x'_2 units from the optical axis.

between disparities is less than half a pixel unit:

$$\begin{aligned}
\delta'_x - \delta_x &= \frac{bf}{Z_0 - \Delta} - \frac{bf}{Z_0} \\
&= \frac{(10 \times 10^{-3}\ \mathrm{m})(2.24 \times 10^{-3}\ \mathrm{m})}{0.9\ \mathrm{m}} - \frac{(10 \times 10^{-3}\ \mathrm{m})(2.24 \times 10^{-3}\ \mathrm{m})}{1\ \mathrm{m}} \\
&= 24.8 \times 10^{-6}\ \mathrm{m} - 22.4 \times 10^{-6}\ \mathrm{m} \\
&= 4.4 - 4.0 \text{ pixel units} \\
&= 0.4 \text{ pixel units}
\end{aligned}$$

3.1.2 Finding the hand contour

From the foreground binary image F_t, 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 using an iterative algorithm similar to the one described in Jain et al. [28]. Connected components are identified as regions of eight-connected pixels, all of which have at least one color channel with a value greater than zero. Once connected components are identified, all but the largest are discarded; the result is a binary image with "one" pixels where the connected region is located and "zero" pixels elsewhere. Finally, an iterative contour-following algorithm [28] finds the contour C of the region, again represented by "one" pixels in a binary image. Note that the t subscript for time is dropped from C_t and subsequent variable designations to simplify notation.

Let A be the number of "one" pixels in C. The centroid of the "one" pixels composing the largest contour in the image is thus

$$(\bar{x}, \bar{y}) = \frac{1}{A} \left(\sum_{(i,j)} jC(i,j), \sum_{(i,j)} iC(i,j) \right). \tag{3.4}$$

From this, *Finger Counter* estimates the palm center as $(\bar{x}, \zeta\bar{y})$; parameter ζ was chosen in developing the system. For the *Finger Counter* system, the origin $(0, 0)$ of the image plane was defined to be in the top left corner of the image. Thus, parameter $\zeta > 1.0$ scales the palm center downward in the field of view, based on the

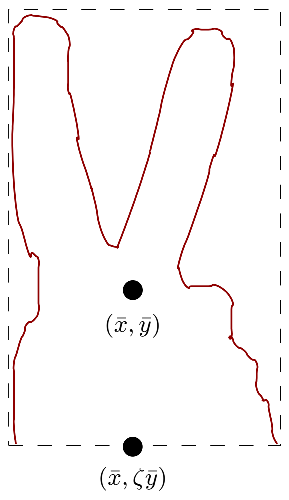


Figure 3.5: Region of interest

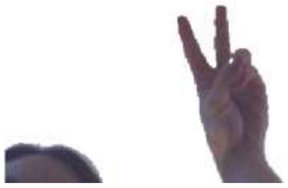
assumption that the hand is held approximately upright. The system also defines a rectangular region of interest containing all contour pixels level with or above the palm center $(\bar{x}, \zeta\bar{y})$, as shown in Figure 3.5.

Figure 3.6 shows the results of each image-processing step. Figure 3.6(a) shows the image prior to processing by the *Finger Counter*'s image-processing module. Figure 3.6(b) shows the foreground image, segmented from the background using *Finger Counter*'s method. Figure 3.6(c) shows the connected components of foreground regions. Finally, Figure 3.6(d) shows the contour of the largest connected component.

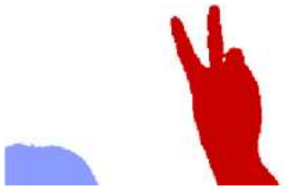
3.2 Feature extraction

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),

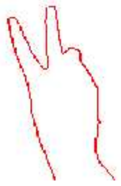
(a) Image prior to processing



(b) Foreground segmented from background



(c) Connected components labeled by color



(d) Contour of largest connected component identified

Figure 3.6: Example results of image-processing steps in *Finger Counter*'s Image Processing Module

are converted to polar coordinates (α, r) with the origin at the estimated palm center $(\bar{x}, \zeta\bar{y})$. The angle component α of the polar coordinates is quantized by using discrete angles from 0 to 180 degrees. Let $\alpha_p = p^\circ$, for $p \in \{0, 1, 2, \cdots, 180\}$. For each α_p, a single edge pixel $\mathbf{e}_p \in C$ with radius r_p is selected as follows: Let S_p be the set of edge pixels $\{\mathbf{e}_k\}$ that fall between rays extending from the polar-coordinate origin at angles $\alpha_p - 0.5^\circ$ and $\alpha_p + 0.5^\circ$. Denote the x and y components of edge pixel $\mathbf{e}_k$ as $x_{\mathbf{e}_k}$ and $y_{\mathbf{e}_k}$. Then, S_p is formally defined by the following equation:

$$S_p = \left\{ \mathbf{e}_k \;\middle|\; \alpha_p - 0.5^\circ \le \arctan\left(\frac{y_{\mathbf{e}_k} - \zeta\bar{y}}{x_{\mathbf{e}_k} - \bar{x}}\right) < \alpha_p + 0.5^\circ \right\}. \tag{3.5}$$

From each S_p, the system chooses the farthest pixel from the polar-coordinates origin, that is, $\mathbf{e}_p = \operatorname{argmax}_{\mathbf{e}_k \in S_p} ||\mathbf{e}_k - (\bar{x}, \zeta\bar{y})||$. The radius r_p corresponding to angle α_p becomes simply $r_p = ||\mathbf{e}_p - (\bar{x}, \zeta\bar{y})||$. Figure 3.7 illustrates this process. Figure 3.8 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 Figure 3.9). Let t be the frame number and k range from 1 through the total number of protrusions 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 r polar coordinate, or distance from estimated palm center $(\bar{x}, \zeta\bar{y})$, exceeds τ_r. If m_k defines the lower end of a range and n_k, the upper end, then, for all p such

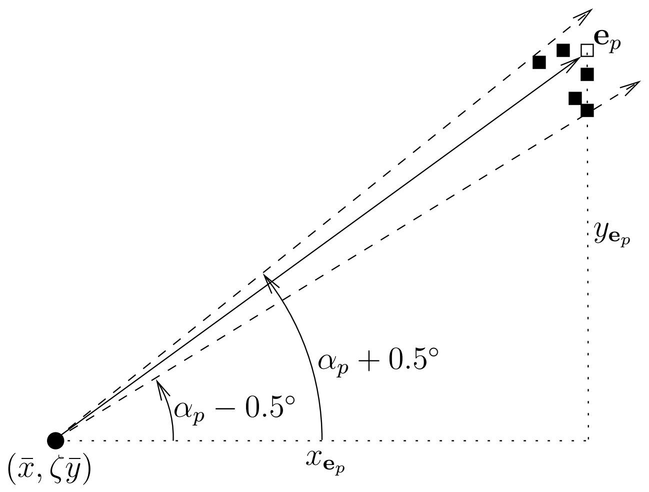


Figure 3.7: Illustration of how a contour pixel $\mathbf{e}_p$ is selected for a discrete polar-coordinate angle α_p. The squares represent the pixels between rays extending from the palm center $(\bar{x}, \zeta\bar{y})$ at angles $\alpha_p - 0.5°$ and $\alpha_p + 0.5°$. The hollow square is $\mathbf{e}_p$, the farthest pixel.

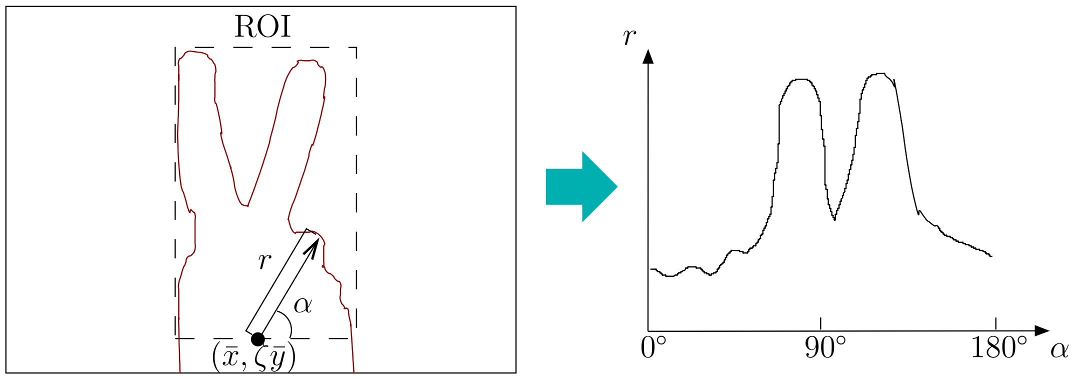


Figure 3.8: Illustration of transformation from Cartesian (x, y) to polar coordinates (r, α). Cartesian coordinate origin is at $(\bar{x}, \zeta\bar{y})$.

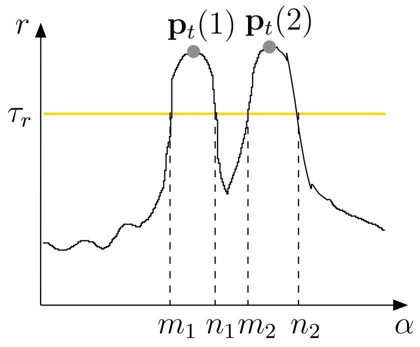


Figure 3.9: Identifying protrusions as local maxima of distance to palm center

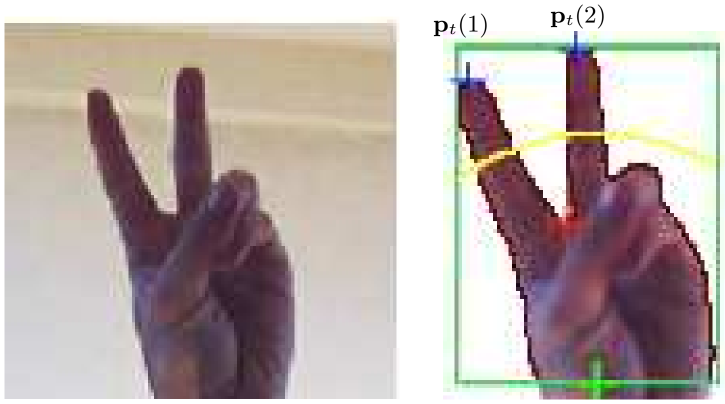


Figure 3.10: From unprocessed image to features. At left is an unprocessed image and, at right, the same image with two detected fingers. The box at right is the region of interest. The arc has radius τ_r. The cross at the base of the region of interest is the estimated palm center $(\bar{x}, \zeta\bar{y})$. The crosses at the top are estimated fingertip positions $\mathbf{p}_t(1)$ and $\mathbf{p}_t(2)$.

that $m_k \leq p \leq n_k$, all pixels are found with $||\mathbf{e}_p - (\bar{x}, \zeta\bar{y})|| > \tau_r$. Then, $\mathbf{p}_t(k) = \text{argmax}_{\mathbf{e}_p}||\mathbf{e}_p - (\bar{x}, \zeta\bar{y})||$. 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. Figure 3.10 shows the contour, estimated palm center, threshold τ_r, and locations of two protrusions found from an image.

3.3 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 a detected 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, that is, $\nu_{t-\rho} = \cdots = \nu_{t-1} = \nu_t$.
2. The squared distance in pixels between the same protrusion in successive frames is less than a threshold τ_ν, that is, $||\mathbf{p}_k(i) - \mathbf{p}_{k-1}(i)||^2 < \tau_\nu$, for all k and i such that $t - \rho < k \leq t$ and $1 \leq i \leq \nu_t$.

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 concludes that there are ν_t fingers held up in frame t.

Chapter 4

Implementation of *Finger Counter* interface

Finger Counter was implemented under Linux kernel 2.4 on a laptop computer with a Pentium IV 1.4 GHz processor and 256 MB of random-access memory. 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 $\mathbf{u}_t$.

Chapter 3 referred to parameters of the *Finger Counter* algorithm. 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 *Finger Counter*'s background-differencing algorithm to correct for minor camera motion. As detailed in Section 3.1.1, for each color channel $\mathcal{C}$, the threshold for foreground pixels was determined as a fraction of the maximum difference between pixels in subimage $\mathbf{I'}_t$ and those in background subimage $\mathbf{B'}_t$, that is, $\tau^{\mathcal{C}} = \kappa \max_{(i,j)} \left\{ \mathbf{I'}^{\mathcal{C}}_t(i,j) - \mathbf{B'}^{\mathcal{C}}_t(i,j) \right\}$. 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. To estimate the position of the center of the palm, the centroid's y coordinate was multiplied by $\zeta = 0.67$. 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 [26], as illustrated in Figure 4.1. It is an artifact of lens production most noticeable in lower priced cameras, such as the webcam used by the *Finger Counter*. For the camera used with the *Finger Counter* interface, radial distortion was found to be as much as 4.6 pixel units, which would have the potential to confound *Finger Counter*'s background-differencing algorithm. Accordingly, the *Finger Counter* interface undistorts each image as it is received from the camera, including the initial background image $\mathbf{B_0}$, using functions provided in Intel's OpenCV library [27].

To minimize the effect of sensor noise, incoming images are convolved with a 3×3 mask approximating convolution with an isotropic Gaussian function. Noise

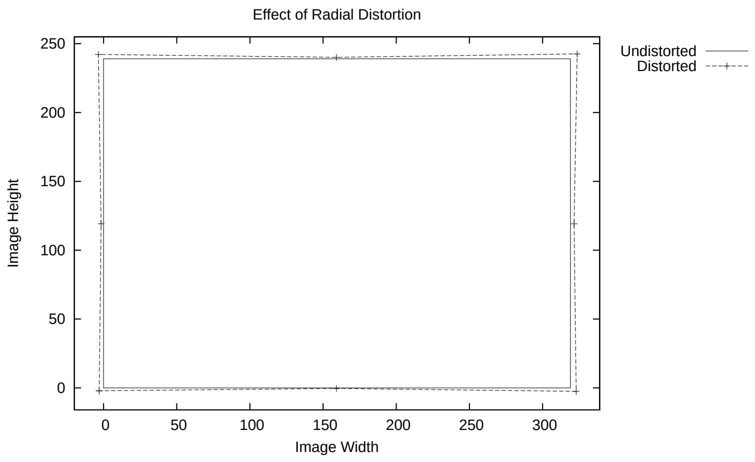


Figure 4.1: Effect of radial distortion on pixels at the edge of the image. Note how the corners are stretched outward.

0.0751	0.1238	0.0751
0.1238	0.2042	0.1238
0.0751	0.1238	0.0751

Figure 4.2: Mask approximating an isotropic Gaussian function with $\sigma = 1.0$

from the camera was found to be sufficiently suppressed by using a Gaussian function with standard deviation $\sigma = 1.0$. The mask is shown in Figure 4.2

Finger Counter uses several threads of execution [55, 9] to improve the system's efficiency. A single process (or running program) can have multiple threads, sometimes called "lightweight processes." The threads timeshare the central processing unit just as processes do. Threads in a single process, however, require less administrative overhead than multiple processes, and they share the same address space, meaning they can access common global variables. Thread libraries, such as pthreads [16], the library used by *Finger Counter*, provide means for controlling access to variables and sending signals between threads.

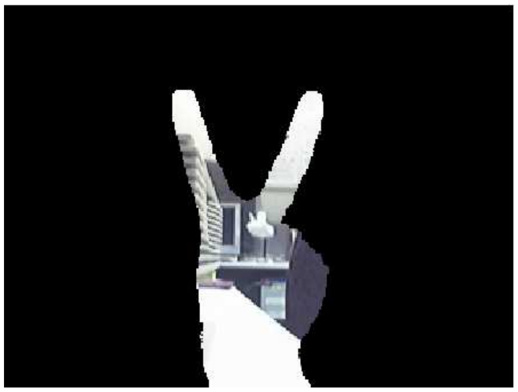


Figure 4.3: Template to teach a user how to hold his or her hand in order to be recognized by the system

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

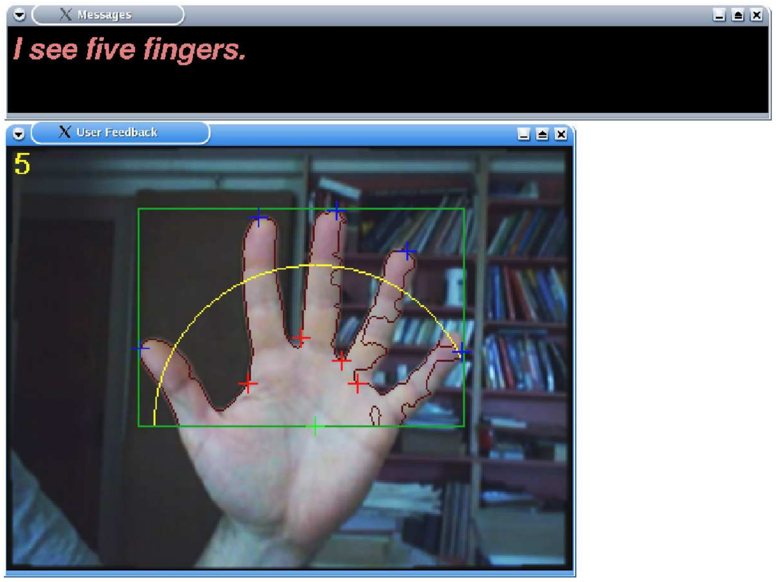
To acclimate a user to the interface, *Finger Counter* briefly displays the template shown in Figure 4.3 and asks the user to fit his or her hand to it. This step teaches the user how to hold his or her hand in order to be recognized by the system. As described in Chapter 3, the interface provides a user-feedback window to give the user an idea of how the computer "sees" his or her hand posture.

Chapter 5

Applications using the *Finger Counter* interface

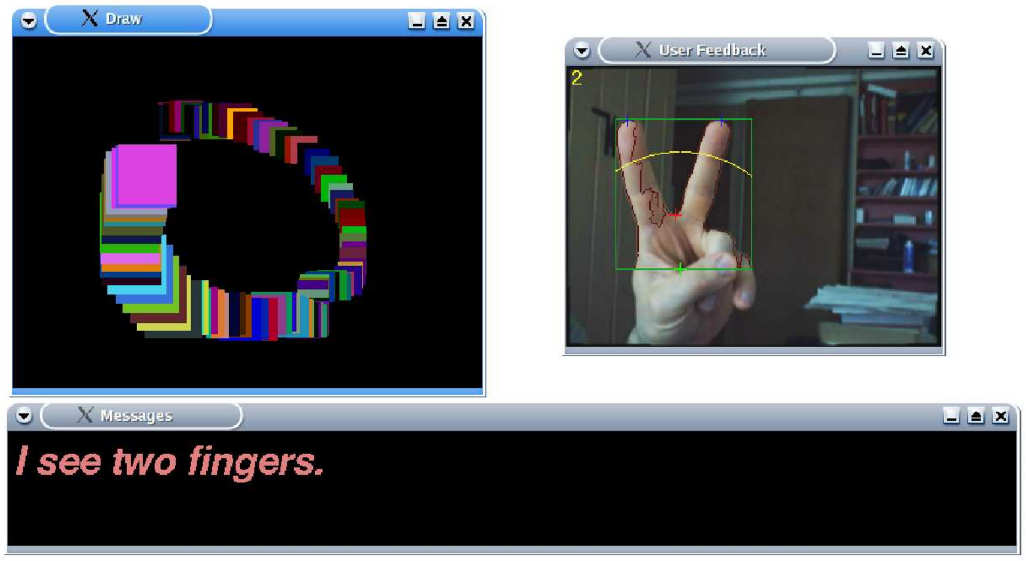
Two applications were developed to demonstrate the hand-recognition capabilities of the *Finger Counter*. The first is a voice-interactive game, a program that audibly prompts the player to hold up a certain number of fingers and then counts them. The output of the system is an audio and text message regarding the number of fingers recognized. Figures 5.1 shows a screenshot of the game.

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 Figure 5.2.



I see five fingers.

Figure 5.1: Voice-interactive game: the top window is for text messages; the bottom window is the "user-feedback window."



I see two fingers.

Figure 5.2: "Finger paint" game: the bottom window gives text messages. The right window is the "user-feedback window," and the left window is the painting.

Chapter 6

Experimental 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 or else translated toward or away from the camera and still be recognized by the system. The second and third methods were quantitative analyses of *Finger Counter* performance: The second method employed a version of the voice-interactive game described in Chapter 5, which was modified to log its performance. The third was the "finger paint" application, also described in Chapter 5, which was modified to log painting activities. The final tool, a questionnaire, was designed to obtain a qualitative evaluation of *Finger Counter*'s usability as an interface.

The earlier version of *Finger Counter*, described at the end of Chapter 2, was tested on 37 subjects. Experience in testing the interface resulted in the most-recent test protocols, which, compared to the earlier test protocols, are streamlined and better designed. In this thesis, testing on the most-recent version of the interface is reported, and earlier test results are used for comparison purposes where relevant.

6.1 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 florescent 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 toward 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.
    - "Roll": The forearm was rotated so that the side of the hand including the base of the little finger was closer to the camera than the side including the base of the thumb, or vice versa.
    - "Yaw": The wrist was rotated in a plane parallel to the image plane, so that, from the *Finger Counter* camera's perspective, the fingertips moved to one side while the palm remained fixed.

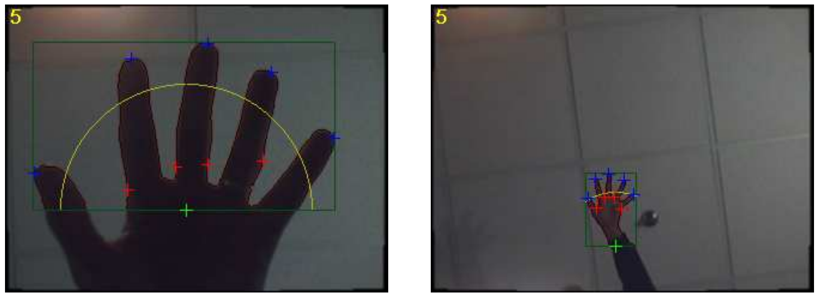
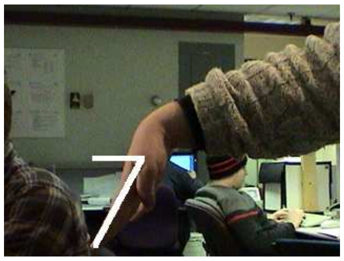


Figure 6.1: Operating limits of the *Finger Counter*: Near and far distance range with correct recognition of signal ✋ .

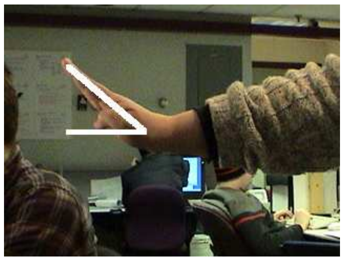
An additional digital camera was set up on a tripod next to the webcam to capture still images of hand positions for “out of plane” rotations, that is, pitch and roll. The user’s hand was placed over the camera so that the palm was oriented parallel to the camera lens, perpendicular to the bottom of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

A tape measure was used to determine how close and how far the hand could be from the camera. Figure 6.1 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.

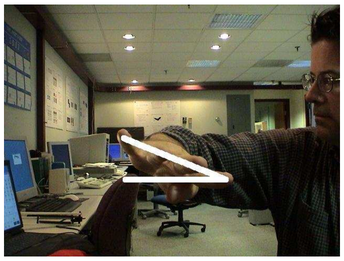
To measure pitch and roll, angles were measured from the images taken by the additional digital camera using an image-manipulation program [2]. Angles



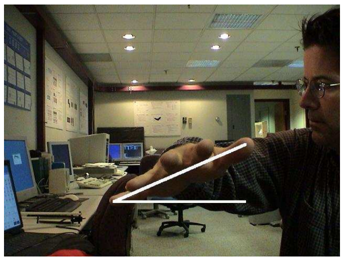
(a) Pitch: wrist in flexion.



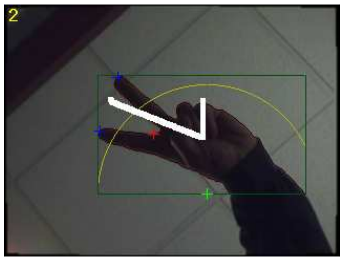
(b) Pitch: wrist in extension.



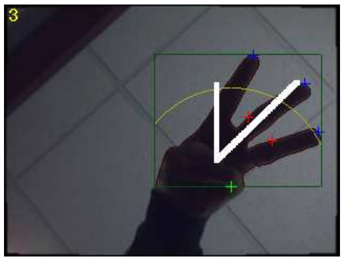
(c) Roll: forearm in pronation.



(d) Roll: forearm in supination.



(e) Yaw: wrist in ulnar deviation.



(f) Yaw: wrist in radial deviation.

Figure 6.2: Operating limits of the *Finger Counter*: Examples of the range of recognized orientations of the hand with respect to the camera. In all cases, the *Finger Counter* camera faced upward. A second camera captured the first four images shown; the second camera was leveled and placed either directly to the left or directly in front of the user. The last two images were captured by the *Finger Counter* camera.

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 6.2 shows sample measurements. The white lines superimposed on the images show the x or y axis and the line used to measure the angle.

For pitch, a negative angle indicates a downward pitch (wrist in flexion) and positive angle, upward (wrist in extension). For roll, a negative angle indicates roll to the left from the user's perspective, assuming the user used his right hand (forearm in pronation). A positive roll angle indicates the forearm is in supination. For yaw, a negative angle measures a movement to the left from the user's point of view, looking at the back of his right hand; that is, ulnar deviation results in a positive angle while radial deviation results in a negative angle.

Tests were conducted on three subjects. Table 6.1 shows how subjects' hand sizes compare with those from anthropomorphic studies. Test Subject 1, a 39-year-old male, had a hand that was 20.4 cm long and 9.5 cm wide, measured at the metacarpale in a relaxed posture. The hand of Test Subject 2, a 23-year-old male, measured 19.1 cm long and 8.9 cm wide. Finally, Test Subject 3, a 21-year-old woman, had a hand that was 17.8 cm long and 10.2 cm wide. The average adult male hand length has been reported as 19.7 cm with standard deviation 0.9 cm in a study of Air Force personnel [23]. In a smaller study including the general public, the mean was 18.7 cm and standard deviation 1.0 cm [8]. The average breadth in the Air Force study was 9.0 cm with standard deviation 0.4 cm and, in the general study, 8.7 cm mean with 0.5 cm standard deviation. In the Air Force study, the average female hand was 17.9 cm long and 7.7 cm wide, with standard deviations 0.9 cm and 0.4 cm respectively. In the general study, the average female hand was 16.7 cm long and 7.5 cm wide, with standard deviations 0.5 cm and 0.3 cm respectively. As shown in Table 6.1, Test Subject 1 had a slightly larger than average hand, but the proportion of length to width was comparable to those in both anthropomorphic

Table 6.1: Average hand measurements compared with measurements of test subjects for tests of *Finger Counter* interface's limitations. N indicates the sample size, and standard deviations are provided in parentheses.

				Ratio of
	N	Length (± SD)	Width (± SD)	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

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 6.2 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, in particular, from 0.29 m to 0.96 m from the camera for one test subject. The ✋ symbol was also least robust to rolls, recognized for one test subject in a range of 40°. For pitch, the ☝ symbol was recognized in the narrowest range of 106°, 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° from left to right.

In tests of the earlier version of the *Finger Counter* interface, the minimum ranges were as follows: distance, 0.4 m; pitch, 84°; roll, 14°; yaw, 13°. The earlier

Table 6.2: Narrowest hand-position ranges for 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°
	1.18 m	40°	114°	21°

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.

6.2 Quantitative evaluation of *Finger Counter* performance

To quantitatively evaluate *Finger Counter*'s performance, test subjects were asked to play modified versions of the two game applications. Experiments with the voice-interactive game were analyzed to determine how accurately the system estimated the hand signal and how long recognition took. Experiments with the finger-paint game were conducted to evaluate how well the *Finger Counter* is able to estimate fingertip positions.

6.2.1 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 Chapter 3, the *Finger Counter* uses a number of successive frames to reach a conclusion as to how many fingers are held up. Table 6.3 shows an excerpt from a sample log showing *Finger Counter*'s estimate of a hand signal over time.

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 the room of a house with ambient natural light, as well as incandescent interior lighting. The camera was placed on a countertop facing upward toward a ceiling about two meters overhead. One test was conducted in a computer-science graduate-student laboratory. In that case, lighting consisted of incandescent and florescent lighting. Again, the camera pointed at the ceiling, about two meters overhead. Sixteen tests were conducted in offices at a university. Florescent lights were accompanied by ambient light coming through windows. The camera again pointed at a ceiling about two meters overhead.

Table 6.3: Sample log of *Finger Counter*'s correct recognition of hand signal in the voice-interactive test. For each image captured, the log records the seconds elapsed since the system requested a hand signal from the user, the requested hand signal, and the hand-signal estimates by the Feature Extraction and Classifier modules. Note that the Feature Extraction module estimated hand signal from the frame captured 1.40 seconds after the request was made, when the user was in the preparation phase of gesture-making. In the next frame, and in subsequent frames, the Feature Extraction module estimated the hand signal . The Classifier module waited to estimate a hand signal until it received five consistent estimates from the Feature Extraction module.

		Estimated Signal	
Time (seconds)	Requested Signal	Feature Extraction module	Classifier module
0.06	3	0	0
0.13	3	0	0
0.19	3	0	0
0.25	3	0	0
0.31	3	0	0
0.38	3	0	0
0.44	3	0	0
0.50	3	0	0
0.57	3	0	0
0.63	3	0	0
0.69	3	0	0
0.76	3	0	0
0.82	3	0	0
0.88	3	0	0
0.94	3	0	0
1.01	3	0	0
1.07	3	0	0
1.13	3	0	0
1.19	3	0	0
1.26	3	0	0
1.32	3	0	0
1.40	3	2	0
1.48	3	3	0
1.56	3	3	0
1.63	3	3	0
1.71	3	3	0
1.79	3	3	3

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 Chapter 5. Figure 5.1 on page 43 shows the test in action. In the voice-interactive test, 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; it was assumed that, if the system failed to identify the signal within ten seconds, then the system was unlikely to identify it given more time. In fact, in the tests the longest response time was 6.77 s.

Table 6.4: Confusion matrix for the voice-interactive test. Empty entries indicate 0.0%. There were 20 test subjects, each of whom made two versions of each hand signal.

		Finger Counter's Estimate				
		☝	✌	🤟	🖖	🖐
Ground Truth	☝	100.0%				
	✌		100.0%			
	🤟		2.5%	97.5%		
	🖖			15.0%	82.5%	2.5%
	🖐				15.0%	85.0%

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 *Finger Counter*'s initial estimate of the hand signal. For hand signals ☝ and ✌ , there was no confusion, that is, *Finger Counter* correctly identified the hand signal made on the first try 100.0% 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 15.0% 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 6.5 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

Table 6.5: 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

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, ranges from 2.04 s to 2.73 s.

These numbers are an improvement on test results on an earlier version of the *Finger Counter*. In tests involving 37 test subjects on the version of the *Finger Counter* described at the end of Chapter 1, correct recognition rates ranged from 68% to 97%, depending upon the hand signal. The mean response time was 2.96 s. The 95%-confidence interval for the earlier test ranged from 2.69 s to 3.24 s. For comparison, in the earlier tests, the time it took subjects to respond to a request to press a particular key on the keyboard, starting with the hands away from the keyboard, was also measured: the mean was 2.24 s and the 95%-confidence interval was from 2.04 s to 2.43 s.

6.2.2 Experiments with "finger paint" application

As described in Chapter 5, 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

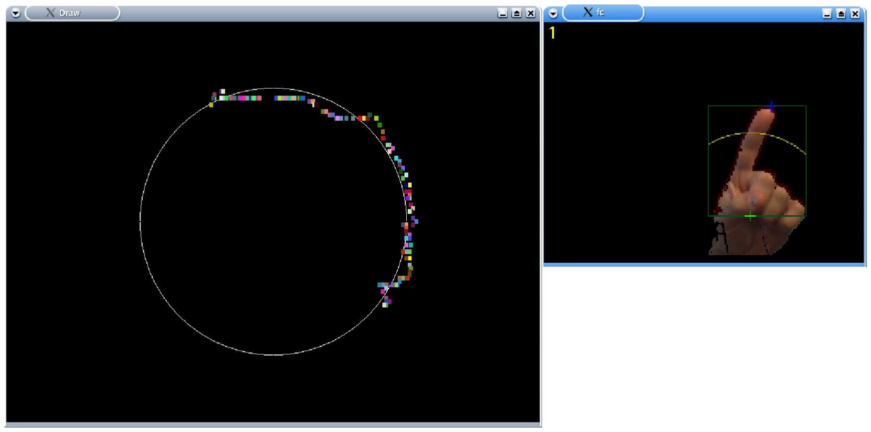


Figure 6.3: *Finger Counter* “finger paint” test: the left screen shows the drawing; the right screen shows the “user feedback” window.

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. Figure 6.3 shows the “finger paint” test in action: the left screen shows the canvas, with circular template, and the right screen shows the user-feedback window. 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 was 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. The test was repeated with the test subject using a computer mouse, instead of the *Finger Counter* interface, for comparison. The protocol for the mouse test was the same as for the test with the *Finger Counter* interface.

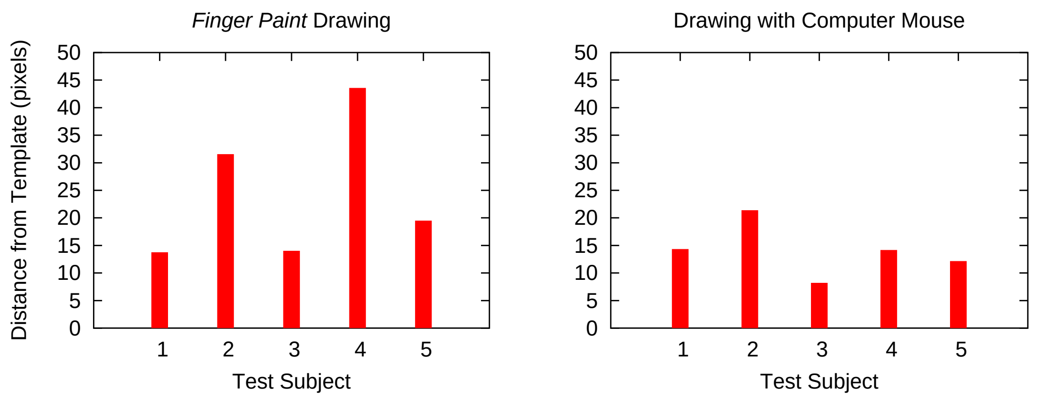


Figure 6.4: Average distances between test subjects' drawings and the circular template

To compare drawings with the template, the average distance was computed to each point in a given drawing from the template. Let $\mathbf{O}$ be the center of the circle template and r be the radius, measured in screen pixel units. Let N be the number of points drawn with the pointing device by the user and $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$ be those points. Then the average distance D for test subject s is computed as follows:

$$D_s = \frac{1}{N} \sum_{i=1}^{N} \left| (||\mathbf{p}_i - \mathbf{O}|| - r) \right| . \tag{6.1}$$

Figure 6.4 shows results of the "finger paint" test. For drawings with the *Finger Counter* interface, D_s ranged from 13.76 pixel units to 43.57 pixel units on an 800×600-pixel draw window containing a circle of radius 200 pixel units. The median of D_s was 19.5 pixel units. By comparison, for drawings with a computer mouse, D_s ranged from 8.21 to 21.4 pixel units, with an median of 14.18 pixel units.

6.3 Qualitative evaluation of ***Finger Counter***'s 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 6.5 shows histograms of the results. The questionnaire responses for the *Finger Counter* and computer mouse 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.

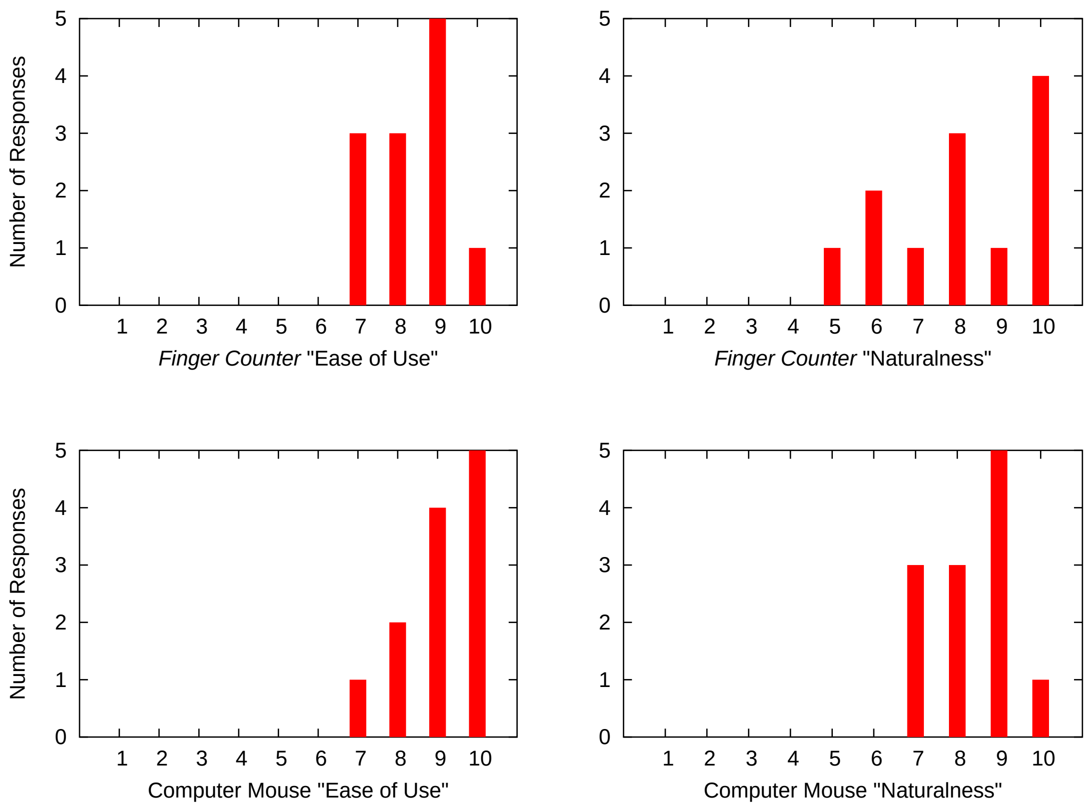


Figure 6.5: Histograms showing ratings of 12 respondents for "ease of use" and "naturalness" of the *Finger Counter* interface versus a computer mouse

Chapter 7

Discussion

The *Finger Counter* extends prior work to produce a simple and reliable hand-signal recognizer. The two main algorithmic contributions are *Finger Counter*'s background-differencing algorithm and its rules-based protrusion estimator using a polar-coordinate representation of the hand contour. The experimental results show that the *Finger Counter* 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 background-differencing algorithm in its present form was motivated by the failure of an earlier version of the system to work properly when subject to vibrations coming from an air conditioner. The algorithm reported here is designed to compensate for small camera motion. The system captures a single frame to use as its estimate of the background. In the literature, adaptive background models often are constructed by gathering statistical information over a series of frames [59]. When the camera is subject to motion, however, a single frame may provide a more accurate estimate of the background. For example, if the system captures two frames of the image and, at a particular pixel location the intensity values vary, the system cannot determine whether the variance is due to a change in the background or camera motion. Moreover, if the camera is vibrating, then averaging the values of pixel at borders between objects of highly disparate brightness may create a background model that fails to represent either of the adjacent objects well. Given that the noise levels on a CCD camera are fairly small, a single frame provides an adequate estimate of the background. Another advantage of this technique is that it is much faster to initialize; to capture enough frames to build up a multi-modal Gaussian model would take several seconds whereas capturing a single frame takes a fraction of a second.

Other gesture-recognition systems [38, 13, 37, 31, 52, 30, 20, 21, 14, 12] that, like the *Finger Counter*, did not require training were not reported with quantitative experimental results that could be compared with the *Finger Counter*'s recognition rates. Hand-gesture recognition systems that did require training are comparable to the *Finger Counter* in successfully recognizing hand gestures. For instance, Triesch and von der Malsburg report that their system correctly recognized gestures 92.9% of the time against a uniform background and 85.8% of the time against a complex background [62, 63]. Flórez et al. report gesture-detection success rates ranging from 90-100%, depending upon the particular gesture. Lockton and Fitzgibbon report a 99.87% success rate, defined as one minus the percentage of false positives, however, their test was conducted on a video stream of images and thus one successful recognition was likely counted a number of times at frame rate until the user changed hand pose [40]. Also, Lockton and Fitzgibbon's testing protocol was limited to test images under similar lighting and background conditions as the training images. Kjeldsen and Kender's system achieved correct pose-recognition rates ranging from 85-90%

of the time, depending upon the selection of training and test images [34]. Starner et al. recognized dynamic hand gestures with accuracy rates ranging from 74.5% to 97.8%, depending upon test conditions [58]. Shamaie and Sutherland reported an 89.6% recognition rate for dynamic hand gestures [54].

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 *Finger Counter*. Indeed, anthropomorphic studies show relatively little variation in hand length or width [8, 23].

Experiments detailed in Chapter 6 show that the *Finger Counter* is robust to variations in hand position with respect to the camera. Although the *Finger Counter* 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 *Finger Counter* tolerates a wide range of motion.

Tests show the *Finger Counter* interface to be reliable and efficient. The *Finger Counter* 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.

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 *Finger Counter* is adequate for rough drawing tasks, but, for fine pointer control, a computer mouse is superior. Of course, given that all of the test subjects had at least a year of computer experience using a mouse, and they were exposed to the *Finger Counter* for only a few minutes, it is understandable that they might not find it as natural or easy to use as a computer mouse.

In further testing of the version of the *Finger Counter* interface reported by Crampton and Betke [11], the system 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, as reported in Chapter 6, 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.

Chapter 8

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 parallel to the image plane, the system is able to make reliable recognitions in real time. Additionally, by compensating for small camera movements, 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 *Finger Counter* is a useful interface for selecting among a small number of menu items. As a pointing device, 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 (auto mechanic) or sterile (surgeon) or because one hand is occupied with another task. For these situations and others, the *Finger Counter* may prove a useful interface.

Bibliography

[1] Common hand signals for crane operations. Retrieved November 4, 2003, from http://www.signweb.com/installation/cont/cranesignals.htm.

[2] The gimp homepage. Retrieved October 21, 2003, from http://www.gimp.org.

[3] Linda Acredolo and Susan Goodwyn. *Baby Signs: How To Talk with Your Baby Before Your Baby Can Talk*. McGraw-Hill, 1996.

[4] Active Divers Association. Underwater communication and hand signals. Retrieved November 4, 2003, from http://www.activedivers.org/handsignc.htm.

[5] 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.

[6] Margrit Betke and Nicholas Makris. Recognition, resolution and complexity of objects subject to affine transformation. *International Journal of Computer Vision*, 44(1):5–40, 2001.

[7] Ulrich Bröckl-Fox, L. Kettner, A. Klingert, and L. Kobbelt. Hand gesture recognition as a 3-D input technique. In Nadia Magnenat-Thalmann and Daniel

Thalmann, editors, *Artificial Life and Virtual Reality*, pages 173–187, New York, NY, 1994. Wiley.

[8] Bryan Buchholz and Thomas J. Armstrong. An ellipsoidal representation of human hand anthropometry. *Human Factors*, 33(4):429–441, 1991.

[9] David R. Butenhof. *Programming with POSIX® Threads*. Addison-Wesley, 1997.

[10] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image and Vision Computing*, 21(8):745–758, 2003.

[11] Stephen C. Crampton and Margrit Betke. Counting fingers in real time: A webcam-based human-computer interface with game applications. In *Proceedings of the Universal Access in Human-Computer Interaction Conference*, pages 1357–1361, Crete, Greece, June 2003.

[12] 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.

[13] Ross Cutler and Matthew 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.

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

[15] Zoran Duric and Azriel Rosenfeld. Shooting a smooth video with a shaky camera. *Machine Vision and Applications*, 13(5-6):303–13, 2003.

[16] Scott Dybiec. POSIX threads. Retrieved July 24, 2003, from http://www.humanfactor.com/pthreads/posix-threads.html.

[17] Chicago Mercantile Exchange. Hand signals. Retrieved November 4, 2003, from http://www.cme.com/edu/getstr/handsgnl648.html.

[18] Martin A. Fischler and Robert C. Bolles. Random sample consesus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–95, June 1981.

[19] Francisco Flórez, Juan Manuel García, José García, and Antonio 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, DC, May 2002.

[20] W. Freeman, D. Anderson, P. Beardsley, C. Dodge, M. Roth, C. Weissman, and W. Yerazunis. Computer vision for interactive computer graphics. *IEEE Computer Graphics*, 18(3):42–53, May/June 1998.

[21] W.T. Freeman, P.A. Beardsley, H. Kage, K. Tanaka, C. Kyuman, and C. Weissman. Computer vision for computer interaction. In *ACM SIGGRAPH Computer Graphics*, volume 33, pages 65–68, November 1999.

[22] Yoav Freund and Robert 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.

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

[24] Dong Guo, Yonghua 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.

[25] Lalit Gupta and Suwei Ma. Gesture-based interaction and communication: Automatic classification of hand gesture contours. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(1):114–120, February 2001.

[26] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[27] Intel Corporation. Intel research — microprocessor research — media. Retrieved January 21, 2004, from http://www.intel.com/research/mrl/research/opencv/, 2004.

[28] R. Jain, R. Kasturi, and B. Schunck. *Machine Vision*. McGraw-Hill, 1995.

[29] Cullen 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.

[30] Eu-Mi Ji, Ho-Sub Yoon, and Younglae Bae. Touring into the picture using hand shape recognition (poster session). In *Proceedings of the Eighth ACM International Conference on Multimedia*, pages 388–390, Los Angeles, CA, November 2000.

[31] Kang-Hyun Jo, Yoshinori Kuno, and Yoshiaki 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.

[32] Adam 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.

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

[34] Rick Kjeldsen and John 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.

[35] Markus Kohler. ZYKLOP visual human-computer interaction through hand gestures. Retrieved November 18, 2003, from http://ls7-www.cs.uni-dortmund.de/research/gesture/zyklop, 2000.

[36] Myron W. Krueger. *Artificial Reality.* Addison-Wesley, 1991.

[37] Senthil Kumar and Jakub 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.

[38] Ivan Laptev and Tony 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. Vancouver, Canada, July 2001.

[39] Andre Leroi-Gourhan. *Gesture and Speech.* MIT Press, 1993.

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

[41] Scott I. MacKenzie. Input devices and interaction techniques for advanced computing. In W. Barfield and T.A. Furness III, editors, *Virtual environments and advanced interface design*, pages 437–470. Oxford University Press, 1995.

[42] David McNeill. *Hand and mind.* University of Chicago Press, 1992.

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

[44] Gérard Medioni, Isaac Cohen, François Brémond, and Ramakant Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):873–889, 2001.

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

[46] 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.

[47] Department of the Army. Field manual no. 21-60, chapter 2, arm-and-hand signals for ground forces. Retrieved November 4, 2003, from http://www.adtdl.army.mil/cgi-bin/atdl.dll/fm/21-60/Ch2.htm, 1987.

[48] Kenji Oka, Yoichi Sato, and Hideki Koike. Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 429–434, Washington, DC, 2002.

[49] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[50] Indrajit Poddar, Yogesh Sethi, Erean Ozyildiz, and Rajeev Sharma. Toward natural gestures/speech HCI: A case study of weather narration. In *Proceedings of the Workshop on Perceptual User Interfaces*, pages 1–6, San Fransisco, CA, November 1998.

[51] Marco Porta. Vision-based interfaces: methods and applications. *International Journal of Human-Computer Studies*, 57:27–73, 2002.

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

[53] 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.

[54] Atid Shamaie and Alistair 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. K.N. Toosi University of Technology.

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

[56] Eric Soares. Hand signals for sea kayakers. Retrieved November 4, 2003, from http://www.baskers.org/signals.html.

[57] Michael Stark, Markus Kohler, and Projektgruppe ZYKLOP. Video based gesture recognition for human computer interaction. Technical Report 593/1995, Universität Dortmund, Dortmund, Germany, 1995.

[58] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[59] Chris 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, 1999.

[60] Richard Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics*, 16(3):22–30, 1996.

[61] Tomoichi Takahashi and Fumio Kishino. A hand gesture recognition method and its applications. *Systems and Computers in Japan*, 23(3):38–48, 1991.

[62] Jochen Triesch and Christoph 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, 1996.

[63] Jochen Triesch and Christoph von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1449–1453, 2001.

[64] Christian Vogler and Dimitris N. Metaxas. ASL recognition based on a coupling between HMMs and 3d motion analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 363–369, 1998.

[65] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[66] Ying Wu and Thomas 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.

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

[68] Hanning Zhou and Thomas 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.