Computer Vision for Interactive Computer Graphics

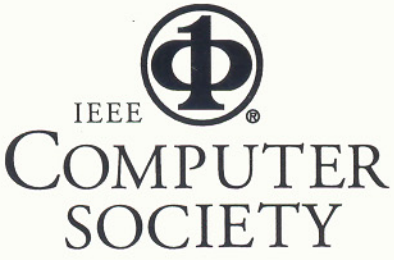
William T. Freeman, David B. Anderson, Paul A. Beardsley, Chris N. Dodge, Michael Roth, Craig D. Weissman, William S. Yerazunis, Hiroshi Kage, Kazuo Kyuma, Yasunari Miyake, and Ken-ichi Tanaka

Reprint

IEEE Computer Graphics and Applications

Volume 18, Number 3

May-June 1998



Washington ◆ Los Alamitos ◆ Brussels ◆ Tokyo

PUBLICATIONS OFFICE, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1314 USA

Computer Vision for Interactive Computer Graphics

William T. Freeman, David B. Anderson,
Paul A. Beardsley, Chris N. Dodge, Michal Roth,
Craig D. Weissman, and William S. Yerazunis
MERL—A Mitsubishi Electric Research Laboratory

Hiroshi Kage, Kazuo Kyuma, Yasunari Miyake, and
Ken-ichi Tanaka
Mitsubishi Electric

We describe vision algorithms for interactive graphics and present vision-controlled graphics applications using these algorithms. Some applications employ an artificial retina chip for image detection or preprocessing.

Vision can be a powerful interface device for computers because of its potential for sensing body position, head orientation, direction of gaze, pointing commands, and gestures. Such unencumbered interaction can make computers easier to use.

Applications could include computer-controlled games or machines, or a more natural interface to the computer itself. Rather than pressing buttons, players could in a computer game pantomime actions or gestures, which the computer would recognize. CAD designers might use their hands to manipulate objects in the computer. People might use hand gestures to give commands to machines or appliances—a potential benefit to surgeons, soldiers, or disabled patients. The vision-based interactions could make the machine interaction more enjoyable or engaging, or perhaps safer.

Interactive applications pose particular challenges. The response time should be very fast. Users should sense no appreciable delay between when they make a gesture or motion and when the computer responds. Computer vision algorithms should be reliable, work for different people, and work against unpredictable backgrounds.

Also, economic constraints exist: vision-based interfaces will replace existing ones, which often cost very little. A hand-held video game controller and a television remote control each cost a few tens of dollars. Even for added functionality, consumers may not want to spend more.

Academic and industrial researchers have recently focused on analyzing images of people.[1] While researchers have made progress, the problem is difficult, and many present-day algorithms are complex, slow, or unreliable. The algorithms that do run near real time do so on very expensive computers compared to the hand-held interface devices.

Fortunately, interactive graphics applications offer particular advantages that make low-cost, real-time vision control possible. First, the application context restricts possible visual interpretations. For example, if a game context requires that the player run (in place), the vision system may only need to ascertain how fast the player runs. This vision problem proves easier to solve than a 3D reconstruction of a player's unknown motion. Second, a human is in the loop. Users can exploit the immediate visual feedback of the graphical display to change their gesture, if necessary, to achieve the desired effect. If a player leans to make a turn in a game and sees that he hasn't turned enough, he can lean some more.

Fortunately, a niche exists for fast, unsophisticated computer vision algorithms that capitalize on the advantages of interactive graphics applications.

We developed or applied various vision algorithms suitable for interactive graphics applications. Here we describe various fundamental visual measurements, in order of increasing complexity: large object tracking, shape recognition, motion analysis, and small object tracking. We used these measurements to make vision-based interfaces for several computer games, plus hand gesture controllers for a toy robot, crane, and television set. We also developed a special image detector/processor that further reduces costs.

Some researchers have undertaken related projects, including the Alive work at the Massachussets Institute of Technology Media Lab,[2] and the pioneering work of Krueger.[3] Our work is similar in spirit to a thrust of computer vision research known as "active vision," which emphasizes real-time response to vision problems.[4]

Large object tracking

In some interactive applications the computer needs to track the position or orientation of a body or hand that is prominent in the camera's visual field. Relevant applications might include computer games or interactive machine control where the camera's viewing con-

0272-1716/98/$10.00 © 1998 IEEE

Image Moments

Image moments provide useful summaries of global image information. The moments involve sums over all pixels and so are robust against small pixel value changes.

If $I(x, y)$ is the image intensity at position x, y, then the image moments, up to second order, are

$$M_{00} = \sum_x \sum_y I(x, y) \qquad M_{11} = \sum_x \sum_y xyI(x, y)$$

$$M_{10} = \sum_x \sum_y xI(x, y) \qquad M_{01} = \sum_x \sum_y yI(x, y)$$

$$M_{20} = \sum_x \sum_y x^2 I(x, y) \qquad M_{02} = \sum_x \sum_y y^2 I(x, y)$$

We can find the position, x_c, y_c, orientation θ, and dimensions L_1 and L_2 of an equivalent rectangle that has the same moments as those measured in the image. Those values give a measure of an object's position, orientation, and aspect ratio. We have

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}}$$

Define the intermediate variables a, b, and c,

$$a = \frac{M_{20}}{M_{00}} - x_c^2$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c\, y_c\right)$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2$$

According to Horn[5] we have

$$\theta = \frac{\arctan\left(b, \left(a - c\right)\right)}{2}$$

and

$$L_1 = \sqrt{6\left(a + c + \sqrt{b^2 + (a - c)^2}\right)}$$

$$L_2 = \sqrt{6\left(a + c - \sqrt{b^2 + (a - c)^2}\right)}$$

The image moments can be calculated from three projections of the image, as described by Horn.[5] Let the vertical, horizontal, and diagonal projections be

$$V(x) = \sum_y I(x, y)$$

$$H(y) = \sum_x I(x, y)$$

and

$$D(t) = \sum_s I\left(\frac{t - s}{\sqrt{2}}, \frac{t + s}{\sqrt{2}}\right)$$

Then the image moments can be written as

$$M_{00} = \sum_x V(x)$$

$$M_{11} \sum_t t^2 D(t) - \frac{M_{20}}{2} - \frac{M_{02}}{2}$$

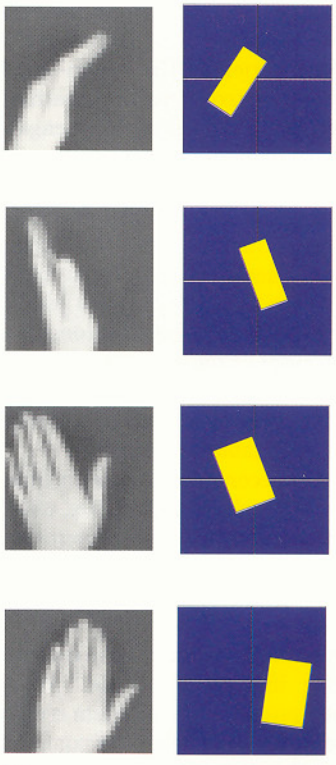
$$M_{10} = \sum_x xV(x) \qquad M_{01} = \sum_y yH(y)$$

$$M_{20} = \sum_x x^2 V(x) \qquad M_{02} = \sum_y y^2 H(y)$$

These equations replace the double sums of Equation 1 with single sums, which speeds up the processing or accommodates processing by fast, special hardware (see the sidebar "Artificial Retina Chip").

ditions are constrained. In such cases, describing the image's overall properties might suffice.

Image moments, which are fast to compute, provide a very coarse summary of global averages of orientation and position (see the sidebar "Image Moments"). If the camera views a hand on a uniform background, this method can distinguish hand positions and simple pointing gestures, as shown in Figure 1a. We implemented this to control the motion of the toy robot in Figure 1b. The robot followed the direction in which the hand was pointing; tilting the hand perpendicular to the camera caused the robot to stop.



(a)

(b)

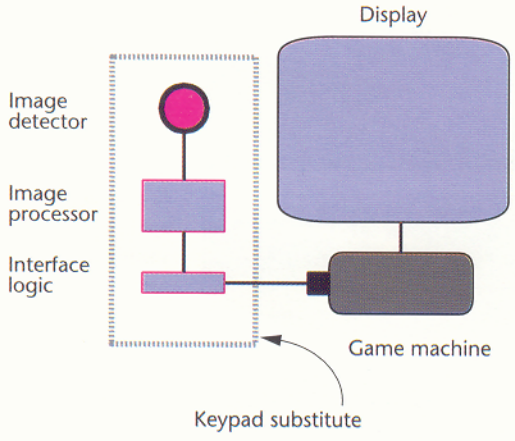
1 (a) Hand images (from a low-resolution detector) and equivalent rectangles, having the same first- and second-order moments. We measured x-y position, orientation, and projected width from the rectangle. (b) Using image moments, we controlled a toy robot's motion.

We can calculate moments particularly quickly using a low-cost detector/processor that we developed, called the artificial retina chip (see the sidebar "Artificial Retina Chip"). This chip combines image detection with some low-level image processing (named artificial retina because the human retina combines those same abilities). The chip computes various functions that prove useful in the fast algorithms for interactive graphics applications.

We applied several different vision algorithms in interactive computer games. As shown in Figure 2, we replaced the hand-held game keypad with a detector, a processor, and interface hardware. The interface hardware, controlled by the processor interpreting detector images, issues commands that look like keypad commands to the Sega Saturn game machine.

Ideally, it's best to design a vision-based interactive game from scratch, building a game on the advantages of the vision interface (good analog adjustments, complex pose configurations). As an easier preliminary step, we selected existing games that we felt were particularly well suited to a vision interface and designed a vision interface that was plug-compatible with the existing keypad interface. Unfortunately for this approach, the game is already tuned for play on the keypad. Not all keypad commands can be issued from vision input, so we omitted some. We developed interfaces for three Sega Saturn games: *Nights*, *Magic Carpet* (both discussed in this section), and *Decathlete* (discussed in the section "Motion analysis").

Part of *Nights* involves steering a sprite flying through a magical world. We wanted users to control the sprite by simple motions or pointing gestures. We had a user position his hand close to the camera so that his hand became a large object in the camera's field of view. To avoid the effects of stationary background clutter, we used a motion-based quantity as input to the moments calculation. To control the sprite's movement, we first calculated a motion energy image using the absolute value of the difference between successive video frames (see Figure 3). The line from the image center to the motion energy image's center of mass indicated the pointing direction. The response to pointing gestures was robust and immediate, allowing the user to control the game character with easy, natural gestures.



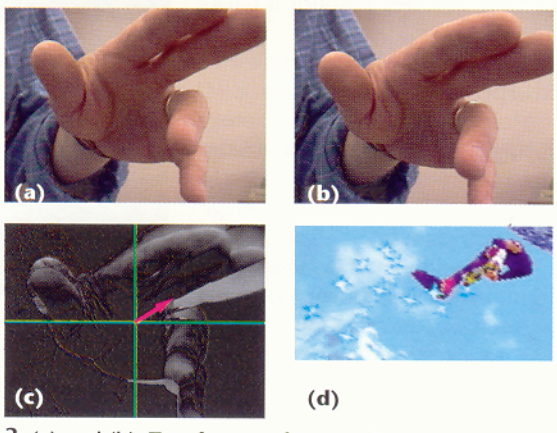
2 System block diagram of an interactive game machine.

Artificial Retina Chip

We developed an image detector that allows programmable on-chip processing. By analogy with the fast, low-level processing that occurs in the eye, we call the detector the *artificial retina* (AR) chip.[6] Figure A shows the elements of the AR chip: a 2D array of variable sensitivity photodetection cells (VSPC), a random-access scanner for sensitivity control, and an output multiplexer. On-chip image processing can be realized by analog operation (addition or subtraction) of each VSPC's output current. The VSPC consists of a pn photo-diode and a differential amplifier that allows for high-detection sensitivity. This structure also realizes a nondestructive readout of the image, essential for image processing. We built detector arrays ranging in resolution from 32×32 to 256×256 pixels. The former was adequate for the game applications presented here. That chip size is 6.5×6.5 square millimeters, with 150×150 micron-sized pixels, using a 1 micron fabrication process. On-chip image processing is obtained from an input image at every time step.

The image processing of the artificial retina can be expressed as a matrix equation. In Figure A, the input image projected onto the chip is the weight matrix *W*. All VSPCs have three electrodes. A direction sensitivity electrode, connected along rows, yields the sensitivity control vector, *S*. The VSPC sensitivities can be set to one of (+1, 0, -1) at

We applied the moments analysis hierarchically, over the image and its four quadrants, to analyze body positions for controlling the game *Magic Carpet* (see Figure 4). Players steer a magic carpet in any direction by leaning or walking in different directions—they can fire a magic spell by holding out an arm. Because players'



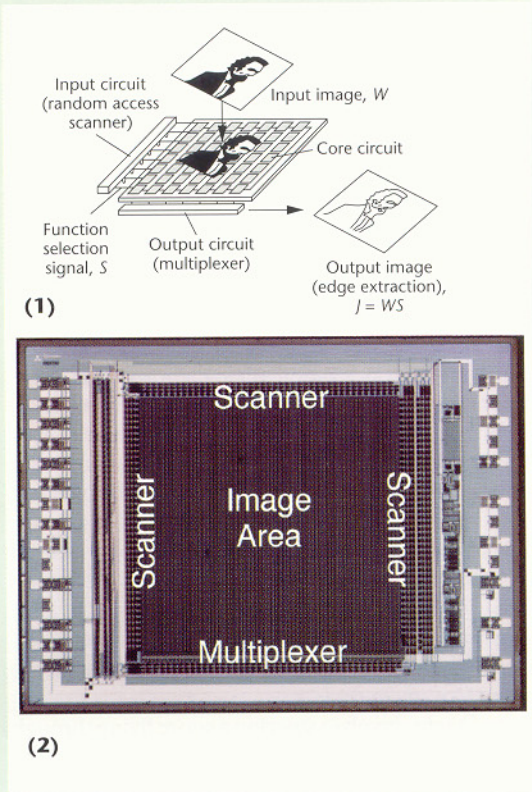
3 (a) and (b): Two frames of camera input. (c) Absolute value of the temporal difference image (Figure 3a minus Figure 3b) and its center of mass (arrow). (d) Figure 3c's center of mass controls the sprite's direction of flight.

each row. An output electrode is connected along columns, yielding an output photocurrent, which is the vector product, $J = WS$. The third electrode resets the accumulated photo-carriers.

By setting the sensitivity control vector, S, appropriately, this hardware can read out the raw image or execute simple linear operations such as local derivatives and image projections. Data acquisition is not limited to video frame rates and can range from 1 to 1,000 Hz. This detector can be manufactured on the same equipment used to process dynamic RAMs (DRAMs) and thus may cost less to manufacture than conventional charge-coupled devices (CCDs).

We integrated this detector/processor chip into an inexpensive *AR module*, which contains a low-resolution (32×32) AR detector chip, support and interface electronics, and a 16-bit 10-MHz microprocessor. Combined, the microprocessor and AR chip can perform general image processing operations quickly. The module is $8 \times 4 \times 3$ cm, and the chip can be fabricated at a considerably lower cost than CCDs because fabrication is based on complementary metal-oxide semiconductor (CMOS) technology.

The artificial retina detector can perform the horizontal and vertical image projections needed for the image moment calculations, saving processor time. The savings depend on the image resolution and microprocessor speed. For the 10-MHz microprocessor of the AR module and 32×32 resolution, the projections would take 10 ms per image on the microprocessor alone, but only 0.3 ms per image using the microprocessor and artificial retina chip.



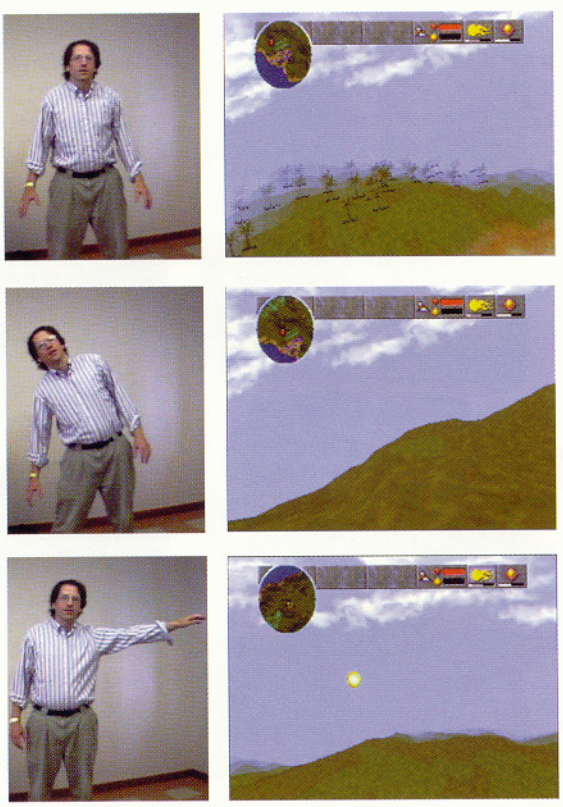
A (1) Schematic structure of the artificial retina chip. An array of variable sensitivity photo–detector cells allows image detection, linear filtering, and projection. (2) Photomicrograph of the chip.

motions match the flying carpet response, the game control is intuitive and simple to learn. Both of the above systems require some control over the background. The motion-based method requires a stationary background, while the shape-based method requires a uniform background.

Shape recognition

Most applications, such as recognizing a particular static hand signal, require a richer description of the input object's shape than image moments provide.

If the hand signals fall in a predetermined set, and the camera views a close-up of the hand, then we can use an example-based approach combined with orientation histograms—a simple method to analyze hand signals. These histograms summarize how much of each shape is oriented in each possible direction, independent of the hand's position inside the camera frame. The



4 We used a hierarchy of computed image moments to analyze the shape and orientation of the player for *Magic Carpet*. The player controls the magic carpet flight by leaning or walking (top and middle). To fire a magic spell, the player simply raises his arm (bottom).

Orientation Histograms

The desire for lighting and position invariance motivates the orientation histogram representation. Figure B shows a comparison of a pixel representation and an orientation representation for a picture of a hand under two different lighting conditions. The hand's pixel values vary considerably with lighting, while the orientation values remain fairly constant. You can calculate the local orientation most simply from the direction of the image gradient. Then the local orientation angle, θ, as a function of position x and y, and image intensities $I(x, y)$, is:

$$\theta(x,y) = \arctan[I(x, y) - I(x-1, y), I(x, y) - I(x, y-1)]$$

We want gestures to be the same regardless of where they occur within the camera's field of view. We chose to achieve this translation invariance by the drastic step of ignoring position altogether, simply tabulating a histogram of how often each orientation direction occurred in the image. Clearly, this throws out information, and some distinct images will be confused by their orientation histograms. In practice, however, you can easily choose a set of training gestures with substantially different orientation histograms from each other (such as Figure 5).

We form a vector, Φ, of N elements, with the ith element showing the number of orientation elements $\theta(x, y)$ between the angles

$$\frac{360^\circ}{N}\left(i - \frac{1}{2}\right) \text{ and } \frac{360^\circ}{N}\left(i + \frac{1}{2}\right):$$

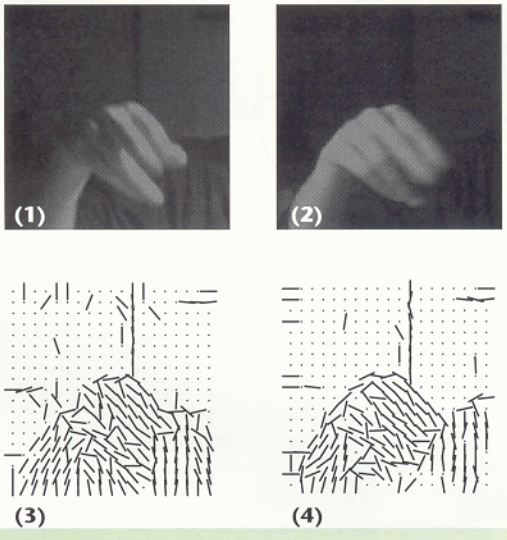
$$\Phi(i) = \sum_{x,y} \begin{cases} 1 \text{ if } \left|\theta(x,y) - \frac{360^\circ}{N} i\right| < \frac{360^\circ}{N} \\ 0 \text{ otherwise} \end{cases}$$

To reduce noise and allow interactions between neighboring orientations, we averaged together adjacent histogram bins. We used $N = 36$ bins for the applications shown here. Simple Euclidean distance,

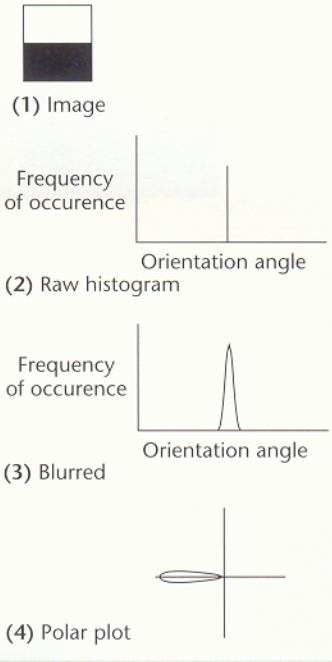
$$\sum_i \left(\Phi_1(i) - \Phi_2(i)\right)^2$$

provides a distance measure between the two images with orientation histograms Φ_1 and Φ_2. Figure C shows the orientation histogram calculation for a simple image.

The resulting orientation histogram is very fast to compute, and can be used as a feature vector to compare with previously analyzed training shapes. McConnell[8] proposed these as a method to analyze the shapes of binary images; we apply them to applications using grayscale images.[7]



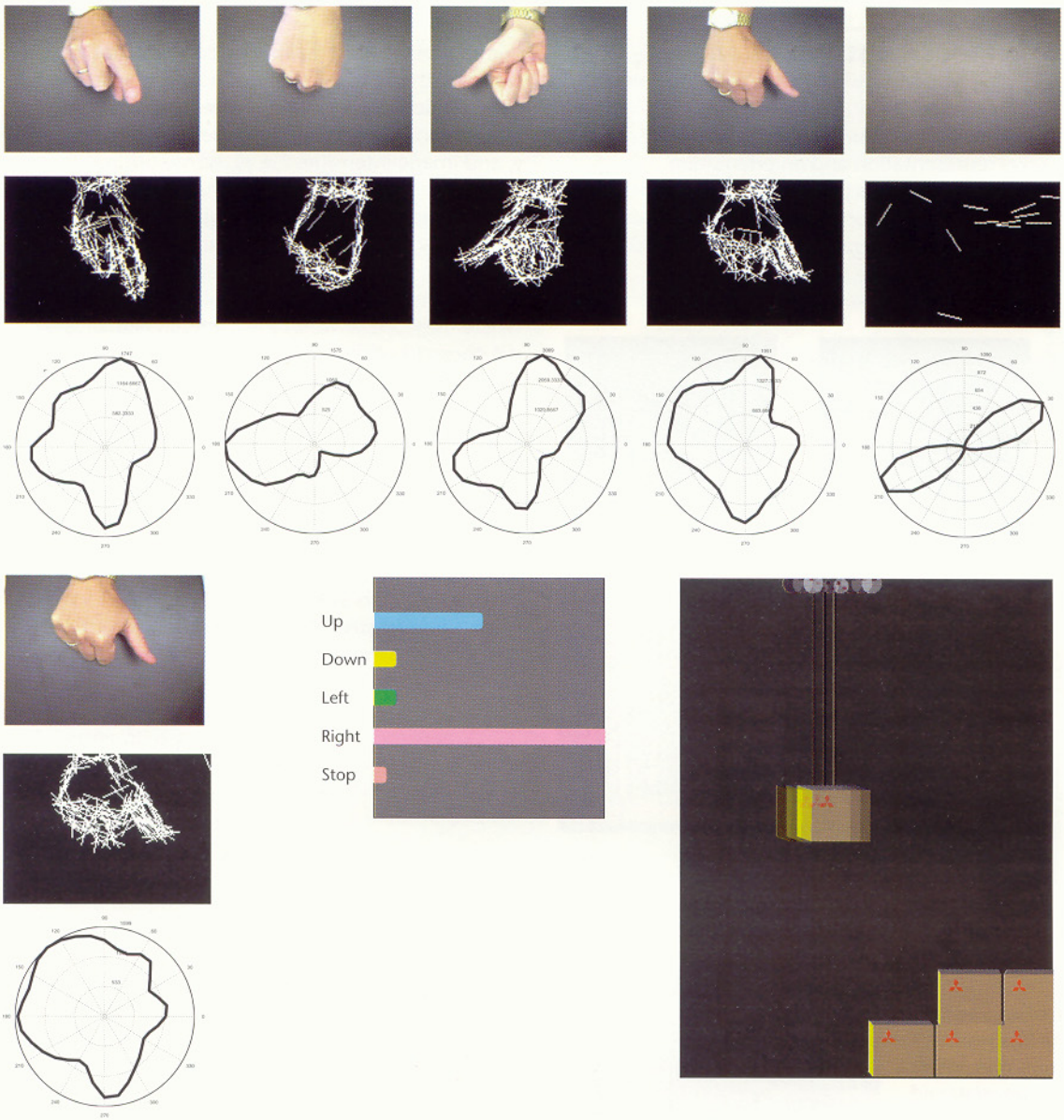
B (1) and (2) A hand under two different lighting conditions (the pixel intensities vary greatly). (3) and (4) Orientation maps of those images are generally more robust to lighting changes than are the pixel intensities.



C Simple illustration of an orientation histogram. (1) An image of a horizontal edge has only one orientation at a sufficiently high contrast. (2) Thus the raw orientation histogram has counts at only one orientation value. (3) To allow neighboring orientations to sense each other, we blurred the raw histogram. (4) The same information, plotted in polar coordinates. We define the orientation to be the direction of the intensity gradient, plus 90 degrees.

computation involves taking spatial derivatives, followed by a nonlinearity, and can be implemented quickly using either conventional or special hardware.

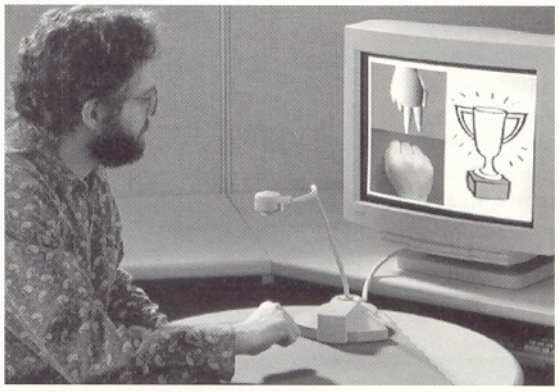
These example-based applications involved two phases: training and running. In the training phase, the user shows the system one or more examples of a hand shape. The computer forms and stores the corresponding orientation histograms. In the run phase, the computer compares the current image's orientation histogram with each of the stored templates and either selects the



5 Orientation histograms in an example-based hand gesture recognition system. Training images with orientation maps and orientation histograms (top). In the run phase, the computer compares the orientation histogram of the current image with those of the training images. A graph showing inverse distances gives user feedback on performance (middle). Hand signals control a toy crane (bottom).

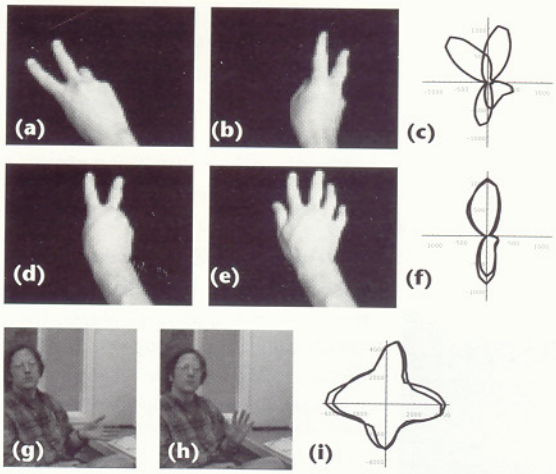
category of the closest match or interpolates between templates, as appropriate. This method, while insensitive to small changes in the size of the hand, is sensitive to changes in hand orientation. For greater robustness, the user may show several training examples, and the computer can use the closest matching example.

We implemented several interactive graphics applications that rely on orientation histograms for hand-gesture recognition.[7] (For more information, see the sidebar "Orientation Histograms.") Figure 5 shows a computer graphic crane that we can control by hand signals. We first trained the system on hand signals for the commands up, down, left, right, and stop, by having the user show an example of each gesture. After training the computer, the user can use those commands to move around a crane under hand-gesture control. A graphical display of the closeness of each hand signal to the five trained categories gives the user feedback for implementing consistent gestures and helps to debug any miscategorizations.



6 Rock, scissors, paper game, based on orientation histograms.

We used the same recognition engine in an interactive game of rock, scissors, paper (Figure 6). A computer graphic "robot hand" plays the game against the user. The robot hand indicates when the user should



7 Problem images for the orientation histogram-based gesture classifier.

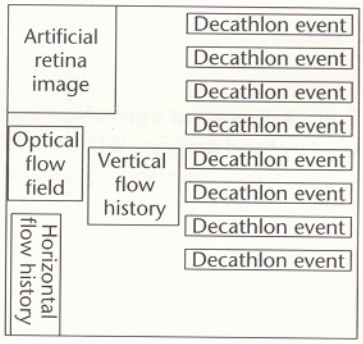
make the gesture, allowing a simple open loop capture of the video gesture.

Each of these systems, while simple and fast, recognizes gestures under the constrained viewing conditions where the hand dominates the image. We showed these systems to many users, most of whom could use them without practice. Generally, the system must be trained for each user; in the run phase, the response is reliable and fast.

We observed the conditions where the user was not satisfied with the gesture classification. Figures 7a and 7b show two images that users feel should represent the same gesture. However, their orientation histograms are very different, as illustrated by their overlaid histograms in Figure 7c. This problem can be addressed by providing training images for each different version of gesture. Some different gestures have very similar orientation histograms. Figures 7d and 7e show an example of this, with the histograms overlaid in Figure 7f. You must choose a vocabulary of gestures that avoids such confusing pairs. Finally, the hand must dominate the image for this simple statistical technique to work. Figures 7g and 7h show images where the hand is a small part of the image. Even though the user has very different hand positions, the orientation histograms of the two images are similar (see Figure 7i). This orientation histogram method is most appropriate for close-ups of the hand. A uniform background provides the best results.

The processing is fast on a conventional general-purpose processor. You can also use special hardware for the low-level image processing tasks (see the sidebar "Artificial Retina Chip").

8 Visual display of analysis of *Decathlete*.



Motion analysis

Often a person's motion signals the important interface information to the computer. Computer vision methods to analyze "optical flow" can be used to sense movements or gestures.

We applied motion analysis to control the Sega Saturn game, *Decathlete* (see Figure 8). The game involves the Olympic events of the decathlon (see Figure 9). The conventional game interface suffers from the limitations of the handheld control—to make the game athlete run faster, the player must press a key faster and faster. We sought to let the user pantomime stationary versions of the athletic events in front of the artificial retina module by running or jumping in place. We

9 An example of a user playing a Decathlon event, the javelin throw. The computer's timing of the set and release for the javelin is based on when the integrated downward and upward motion exceeds predetermined thresholds.

10 Two participants playing the *Decathlete* game (top). Image mosaic showing another player, the *Decathlete* game display, and the artificial retina module located to the right of the display (bottom).

hoped this would add an extra dimension to the game and make it more engaging.

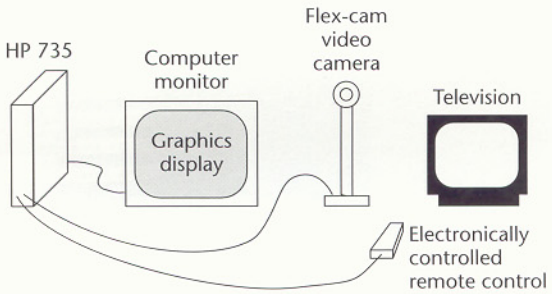
Figure 10 shows examples of the human-computer interaction for the 110-meter hurdles. The user runs in place to make the computer character run. The character's speed is proportional to how fast the player runs in place. To jump over a hurdle, the player raises both hands simultaneously.

Recognizing these actions in a general context would be very challenging, but knowing the game context greatly simplifies the visual recognition. The game context tells us which event the player performs, and we only have to choose between a few different motions the player should perform, or estimate timing or rate parameters. These problems prove much easier to solve. For the 110-meter hurdles, the vision algorithm chooses whether the player runs in place or raises his arms to jump over a hurdle. If he runs in place, the algorithm estimates how fast.

Many methods can analyze optical flow, but relative to our low-cost, high-speed requirements, these can be too slow on conventional machines. Although we don't need a detailed motion analysis, it must be extremely fast.

We developed a fast "optical flow" algorithm (see the sidebar "Fast Optical Flow," next page), which provides an approximation to the screen velocity of moving points on the image appropriate for the large-scale characteristics of the optical flow. We used simple measurements derived from the optical flow to estimate the relevant motion parameters. For example, for the 110-meter hurdles, we tracked the frame averages of the horizontal and vertical components of motion. The frequency of the horizontal velocity's alternation indicates how fast the player runs in place. When the average vertical velocity exceeds a threshold, that indicates a jump command.

With this simple processing, players can believe that the computer understands their physical gestures. We have demonstrated this game at various public venues,



11 Block diagram of a prototype television set controlled by hand gestures. While the prototype uses two display screens, the graphics could be overlaid directly on the television image.

including the computer trade show Comdex in 1996. We used a stationary background to give the most reliable recognition. Players almost always controlled the characters well on their first attempts and immediately became engaged in the game itself.

Small object tracking

The previous algorithms involved tracking or characterizing objects that appear large in the camera frame. Many interactive applications also require tracking objects, such as the user's hand, that comprise only a small part of the image. Here we describe one such application and our system solution.

Our target application aimed to control a television set by hand signals, thus replacing a remote control. Figure 11 shows the prototype of such a television set. This application forced us to face two design problems: one from the human's point of view and one from the computer's point of view. On the human side, we wanted to give a broad set of commands to the television set by hand sig-

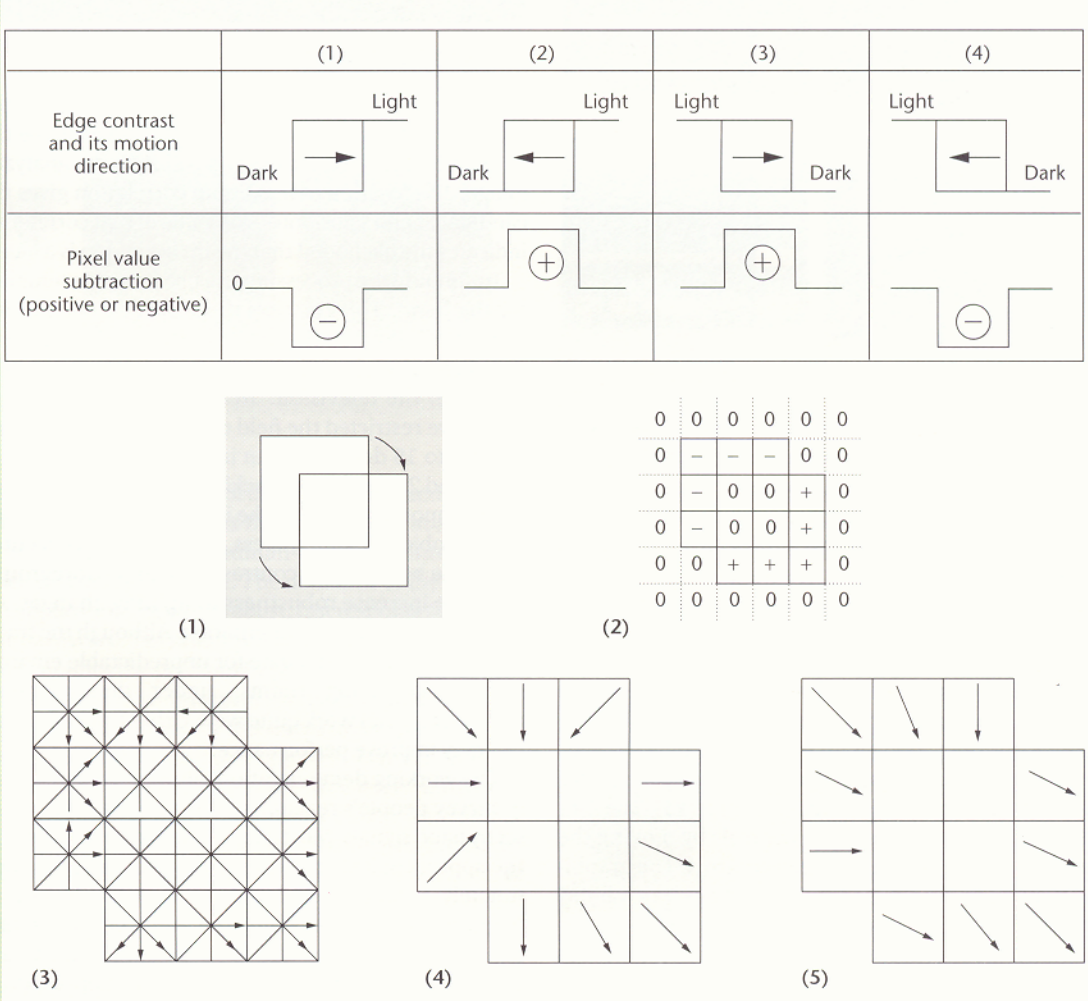
Fast Optical Flow

For an approximation to the optical flow field, we developed a fast algorithm that works well for the coarse-scale analysis we use in interactive game applications. The algorithm classifies the local motion of edges into various possibilities, then pools local estimates across orientation and across space to estimate large-scale optical flow.

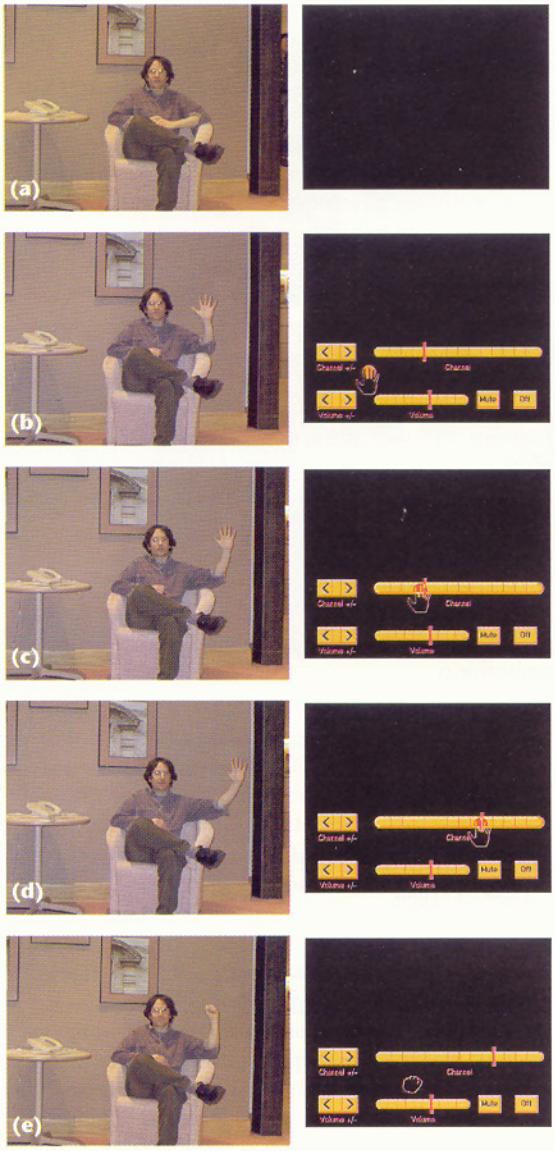
The algorithm follows:

1. Subtract the current frame from the previous one, yielding a temporal difference image.
2. For pixels where the temporal difference is nonzero, enumerate the possible motion directions consistent with the local image measurements. Consider the 1D motion shown in Figure D1. From this example, we derive two rules for motion direction estimation:
   - If the temporal difference is negative, the motion direction is toward the adjacent pixel with higher luminance in the current frame.
   - If the temporal difference is positive, the motion direction is toward the adjacent pixel with lower luminance in the current frame.
3. Apply the 1D direction estimation rules to four orientations (vertical, horizontal, and the two diagonals) at each pixel.
4. Treat each possible motion direction estimate as a vector and average the possible motion estimates at each pixel.
5. Finally, average the above flow estimate at each pixel with the flow estimates of each of its eight neighbors.

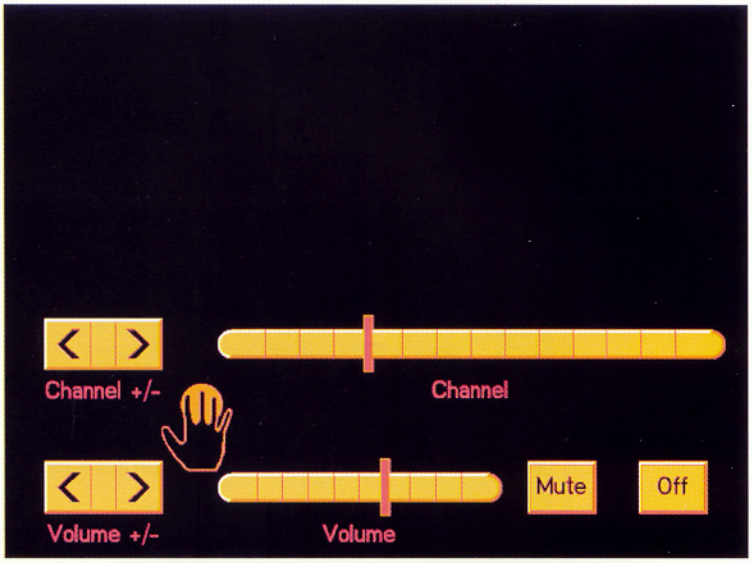
Figure D illustrates our algorithm for the case of a 4×4 pixel moving square. Each subfigure corresponds with one enumerated step of the algorithm above. When the square moves (Figure D1), some pixel value changes occur. Figure D2 represents the change of each pixel value by taking their difference of the frame interval. Figure D3 is a set of orientation vectors at each pixel, and motion vectors are shown in Figure D4. Figure D5 shows the optical flow pattern after spatial smoothing.



D Fast motion estimation algorithm. The top image shows how contrast direction and motion direction interact to yield temporal difference pixel values (algorithm step 2). Figure D1: a moving square. Figures D2 through D5 show steps in the algorithm to calculate optical flow and correspond to algorithm steps 1, 3, 4, and 5, respectively.



12 Sample session of television viewing. (a) Television is off, but searching for the trigger gesture. (b) Viewer shows trigger gesture (open hand). Television set turns on and hand icon and graphics overlays appear. (c) The hand icon tracks the user's hand movement. User changes controls as with a mouse. (d) User has moved hand icon to change channel. (e) User closes hand to leave control mode. After one second, the hand icon and controls then disappear.



13 Under vision input, the hand icon tracks the user's hand, allowing him to use his hand like a computer mouse.

nals, yet we didn't want to require an intensive training period before a user could control the television. On the machine side, recognizing a broad set of hand gestures made within a complex, unpredictable visual scene—such as a living room—proves difficult and remains beyond the realm of current vision algorithms.

We addressed both these design constraints by exploiting the television screen's ability to provide graphical feedback.[9] Our interface design is simple (see Figure 12). To turn on the television, the user holds up his hand. Then a graphical hand icon appears on the television screen, along with graphical sliders and buttons for television adjustments. The hand icon tracks the motions of the user's hand (see Figure 13). The user adjusts the various television controls by moving the hand icon on top of the onscreen controls. The graphical displays and position feedback allows a rich interaction using only simple actions from the user.

The method of moments and orientation histograms of the previous sections aren't adequate to track a small object through the scene. We adopted a template-based technique, called *normalized correlation* (see the sidebar "Normalized Correlation," next page). We examine the fit of a hand template to every position in the analyzed image. The location of maximum correlation gives the candidate hand's position—the value of that correlation indicates the likelihood that the image region is a hand.

To simplify the processing, we used a single template for the hand. This restricts the scale and orientation changes allowed by the image of the user's hand. The working range of the single template system was 6 to 10 feet from the television. To increase the processing speed, we restricted the field of view of the television's camera to 15 degrees when initially searching for the hand, and 25 degrees in tracking mode. We used a running temporal average of the image to subtract out stationary objects. Nonetheless, the best results occured when the background contrasted with the foreground hand. To increase robustness to lighting changes, we used an orientation representation. Although the tracking method is not adequate for unpredictable environments like living rooms, under demonstration conditions it can work quite well. Other tracking methods may improve performance.[10]

The working demonstration allowed us to informally survey people's reactions to controlling a television set by hand signals. Most people seemed quite excited by the approach; it seemed in some ways magical. Unfortunately, to hold up the hand in the required way for a long time is tiresome. While the initial gesture to turn the television set on may be adequate, for channel surfers a more relaxed signal must be developed to indicate a new channel.

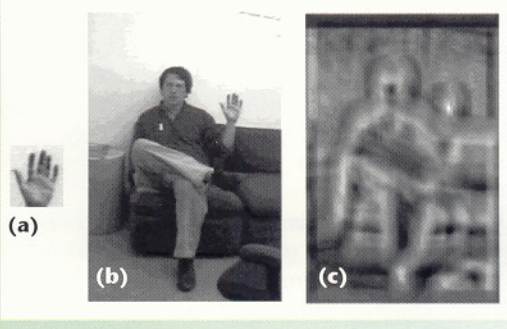
Normalized Correlation

To find instances of an intensity pattern within another image, we can examine the correlation between each small patch of the image and the target pattern. To remove effects due to large, overall changes in brightness, we calculate the normalized correlation between every point of the image and the target pattern.

Consider the $M \times N$ pixel target pattern to be an MN dimensional vector, **a**. For each possible placement of the target pattern within the image, let the corresponding $M \times N$ pixels of the image be an MN dimensional vector, **b**. The normalized correlation for that particular offset of the target pattern within the image is the cosine of the angle between the vectors **a** and **b**:

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\sqrt{(\mathbf{a} \cdot \mathbf{a})\ (\mathbf{b} \cdot \mathbf{b})}}$$

Figure E shows the normalized correlation between an image and a hand template. Note the peak correlation intensity at the true position of the hand.



E (a) Stored hand template. (b) Image to be analyzed. (c) Normalized correlation values at each pixel. The bright spot of the normalized correlation indicates the position of the best match.

Conclusion

Fast, simple vision algorithms and interactive computer graphic applications fit together well at a system level to accommodate human-computer interaction based on computer vision. The demands of the interactive application require a robust, fast response with low-cost hardware. Fortunately, the graphical application also simplifies the problem by providing context to limit the range of visual interpretations and providing user feedback. This allows for interfaces to computer graphic applications based on simple and fast vision algorithms, and possibly special, low-cost hardware. Advances in algorithms, processing power, and memory will continually improve these vision-based interfaces which, over time, may become commonplace. ■

Acknowledgments

We thank Sega for providing the game interface information.

References

1. *Proc. Int'l Workshop on Automatic Face- and Gesture-Recognition*, I. Essa, ed., IEEE Computer Society Press, Los Alamitos, Calif., 1997.
2. T. Darrell et al., "A Novel Environment for Situated Vision and Behavior," M. Landy, L. Maloney, and M. Pavel, eds., *Exploratory Vision: The Active Eye*, Springer-Verlag, Heidelberg, Germany 1995.
3. M. Krueger, *Artificial Reality*, Addison-Wesley, Reading, Mass., 1983.
4. R. Bajcy, "Active Perception," *IEEE Proc.*, Vol. 76, No. 8, 1988, pp. 996-1006.
5. B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, Mass., 1986.
6. K. Kyuma et al., "Artificial Retinas—Fast, Versatile Image Processors," *Nature*, Vol. 372, No. 6502, 1994, pp. 197-198.
7. W.T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition," *Int'l Workshop on Automatic Face- and Gesture-Recognition*, Dept. of Computer Science, Univ. of Zurich, Zurich, Switzerland, 1995, pp. 296-301.
8. R.K. McConnell, *Method of and Apparatus for Pattern Recognition*, U.S. Patent No. 4,567,610, Washington, DC, Jan. 1986.
9. W.T. Freeman and C. Weissman, "Television Control by Hand Gestures," *Int'l Workshop on Automatic Face- and Gesture-Recognition*, Dept. of Computer Science, University of Zurich, Zurich, Switzerland, 1995, pp. 179-183.
10. A. Blake and M. Isard, "3D Position, Attitude, and Shape Input using Video Tracking of Hands and Lips," *Computer Graphics* (Proc. Siggraph 94), ACM Press, New York, 1994, pp. 185-192.

William T. Freeman *is a senior research scientist at MERL, a Mitsubishi Electric Research Lab in Cambridge, Massachusetts where he studies Bayesian models of perception and interactive applications of computer vision. As part of his doctoral work at the Massachusetts Institute of Technology (1992), he developed "steerable filters," a class of oriented filters useful in image processing and computer vision. In 1997 he received the Outstanding Paper prize at the Conference on Computer Vision and Pattern Recognition for work on applying bilinear models to separate "style and content."*

David B. Anderson *is a senior research scientist and evangelist at MERL in Cambridge, Massachusetts. His research interests include distributed virtual environments and ubiquitous computing. Before joining MERL, he managed the Andrew Toolkit project at Carnegie Mellon University and taught computer science in the Pennsylvania Governor's School for the Sciences. He is a director of the VRML Consortium.*

Paul A. Beardsley *is a research scientist at MERL in Cambridge, Massachusetts working in computer vision. His research interests include 3D reconstruction and development of partial 3D representations for image-based rendering. He received a PhD in computer vision from the University of Oxford in 1992. His postdoctoral work was on the recovery of 3D structures from "uncalibrated" image sequences, with a particular application to robot navigation.*

Chris N. Dodge *is an independent interactive media producer and software developer in Cambridge, Massachusetts. He has worked several years in the field of digital signal processing software development. He has a BA from New York University (1991) in an interdisciplinary study of computer science, film/video, and music composition. He earned his MS at the MIT Media Laboratory (1997), focusing on the development of real-time interactive media environments. He was awarded a two-year Artist/Researcher-in-Residency at the Center for Art and Media Technology (ZKM) in Karlsruhe, Germany. As an interactive computer artist, his works have been shown around the world. He won the grand prize at the Artec 97 Media Art Biennial.*

Michal Roth *has worked on natural language applications and intelligent visual processing. Her research interests include developing clustering and neural network methods for use in cytology and disease detection. She earned a BA (cum laude) in mathematics at Technion, Israel in 1983.*

Craig D. Weissman *works in product design and development at Epiphany Software in Belmont, California, a startup company working on the problem of providing packaged data warehousing and data mining. He graduated in 1995 from Harvard University with an BA/MS in applied math and computer science.*

William S. Yerazunis *is a research scientist at MERL in Cambridge, Massachusetts. He has worked in a number of fields, from optics and signal processing (for General Electric's jet engine manufacturing) to computer graphics (at Rensselaer's Center for Interactive Computer Graphics), to artificial intelligence and parallel symbolic computation (for DEC's OPS5, Xcon, and the successor products such as RuleWorks), to radioastronomy and exobiology (at Harvard University), to transplant immunology (for the American Red Cross). He holds 14 US patents. He received his PhD in 1987 from Rensselaer Polytechnic Institute in New York.*

Hiroshi Kage *is a researcher in the Department of Neural and Parallel Processing Technology in the Advanced Technology R&D Center Mitsubishi Electric in Amagesaki, Japan where he has been engaged in developing several machine vision algorithms for artificial retina chips. He received his BE and ME degrees in information science from Kyoto University, Japan in 1988 and 1990, respectively.*

Kazuo Kyuma *is the manager of the Neural and Parallel Processing Technology Department, Advanced Technology R&D Center of Mitsubishi Electric in Amagesaki, Japan. He is also a professor at the Graduate School of Science and Technology, Kobe University, Japan, and a lecturer at Osaka University and the Tokyo Institute of Technology. His research interests cover optoelectronics, advanced LSI systems, and neurocomputing. He received BS, MS, and PhD degrees in electronic engineering from the Tokyo Institute of Technology, Japan, in 1972, 1974, and 1977, respectively.*

Yasunari Miyake *is a researcher in the Department of Neural and Parallel Processing Technology in the Advanced Technology R&D Center of Mitsubishi Electric in Amagesaki, Japan where he has researched optical neural networks and their related devices. He received his BE, ME, and PhD degrees in physical electronics from the Tokyo Institute of Technology, Tokyo, Japan, in 1988, 1990, and 1993, respectively.*

Ken-ichi Tanaka *is manager of the System Integration Technology Group of the Department of Neural and Parallel Processing Technology in the Advanced Technology R&D Center of Mitsubishi Electric in Amagesaki, Japan. His research interests include electric propulsion systems for spacecraft and neural networks. He received his BE and ME degrees in aeronautical engineering from Kyoto University, Japan in 1979 and 1981, respectively.*

Contact Freeman at MERL, A Mitsubishi Electric Research Lab, 201 Broadway, Cambridge, MA 02139, e-mail freeman@merl.com.