Computer Vision for Interactive Computer Graphics

W. T. Freeman[*], D. Anderson[*], P. Beardsley[*],
C. Dodge[*†], H. Kage[‡], K. Kyuma[‡], Y. Miyake[‡], M. Roth,
K. Tanaka[‡], C. Weissman[*§], W. Yerazunis[*]

Appeared in IEEE Computer Graphics and Applications, May-June, 1998, pp. 42-53.

Abstract

Computers looking through a camera at people is a potentially powerful technique to facilitate human-computer interaction. The computer can interpret the user's movements, gestures, and glances. Fundamental visual algorithms include tracking, shape recognition, and motion analysis.

For interactive graphics applications, these algorithms need to be robust, fast, and run on inexpensive hardware. Fortunately, the interactive applications also make the vision problems easier: they constrain the possible visual interpretations and provide helpful visual feedback to the user. Thus, some fast and simple vision algorithms can fit well with interactive graphics applications.

We describe several vision algorithms for interactive graphics, and present various vision controlled graphics applications which we have built which use them: vision-based computer games, a hand signal recognition system, and a television set controlled by hand gestures. Some of these applications can employ a special artificial retina chip for image detection or pre-processing.

Keywords:

Human computer interface, computer vision, interactive computer graphics.

1 Introduction

Vision can be a powerful interface device for computers. There is the potential to sense 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. In a computer game application, rather than pressing buttons, the player could pantomime actions or gestures, which the computer would then recognize. A CAD designer might use his hands to manipulate objects in the computer. A user of machinery or appliances might just have to use hand gestures to give commands, 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. The user should sense no appreciable delay between when he or she makes a gesture or motion and when the computer responds. The computer vision algorithms should be reliable, work for different people, and work against unpredictable backgrounds.

There are also economic constraints: the vision based interfaces will be replacing existing ones which are often very low cost. A hand-held video game controller and a television remote control each cost about $40. Even for added functionality, consumers may not want to spend more.

Academic and industrial researchers have recently been focusing on analyzing images of people [4]. While researchers are making progress, the problem is hard and many present day algorithms are complex, slow or unreliable. The algorithms that do run near real-time do so on computers that are very expensive relative to the hand-held interface devices.

Fortunately, interactive graphics applications offer particular advantages that make low-cost, real-time vision control possible. The vision input functions as part of a larger system that includes an interactive graphical display. This may be a computer game, the control of a television set, or control of objects in a virtual or real world. The first advantage is that the application con-

[*] MERL,
a Mitsubishi Electric Research Lab, 201 Broadway, Cambridge, MA 02139. *freeman@merl.com*

[†] present address: MIT Media Lab, 20 Ames St., Cambridge, MA 02139

[‡] Mitsubishi Electric, Advanced Technology R&D Center, 8-1-1, Tsukaguchi-Honmachi, Amagasaki City, Hyogo 661, Japan

[§] present address: Epiphany Software, Belmont, CA

text restricts the possible visual interpretations. Perhaps, from the game context, we know that the player is running, then the vision system may need only to ascertain how fast the player is running. This is a much easier vision problem to solve than a 3-d reconstruction of the player's motion from scratch. Secondly, there is a human in the loop. The user can exploit the immediate visual feedback of the graphical display to change their gesture, if necessary, to achieve the desired effect. If a player is leaning to make a turn in a game and sees that he isn't turning enough, he can lean more.

There is a niche for fast, unsophisticated computer vision algorithms which take advantage of the simplifications that interactive graphics applications provide.

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

Some researchers have undertaken related projects, including the Alive work at the MIT Media Lab [3], and the pioneering work of [8]. It is similar in spirit to a thrust of computer vision research known as "active vision", which emphasizes real-time response to real vision problems [1].

2 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 visual field of the camera. Relevant applications might be computer games, or interactive machine control where the camera viewing conditions are constrained. In such cases, a description of the overall properties of the image may be adequate.

Image moments, which are fast to compute, provide a very coarse summary of global averages of orientation and position (see 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 Fig. 1 (a). We have implemented this to control the motion of a toy robot, Fig. 1 (b). The robot followed the direction that the hand was pointing; tilting the hand perpendicular to the camera caused the robot to stop.

We are able to calculate moments particularly quickly using a low-cost detector/processor that we have developed, called the artificial retina chip (see sidebar, `Artificial Retina`). This chip combines image detection with some low-level image processing (named artificial retina by analogy with those combined abilities of the human retina). The chip can compute various functions useful in the fast algorithms for interactive graphics applications.

An application area to which we have applied several different vision algorithms is interactive computer games. As shown in Fig. 2, we replace the handheld game keypad with a detector, a processor, and interface hardware. The interface hardware, controlled by the processor interpreting detector images, issues commands which look like keypad commands to the Sega Saturn game machine.[1]

The ideal way to design a vision based interactive game is from scratch, building a game on the particular 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. A disadvantage of this approach is that the game is tuned for play on the keypad, and not all of the keypad commands are easy to issue from vision input; 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 on motion analysis.

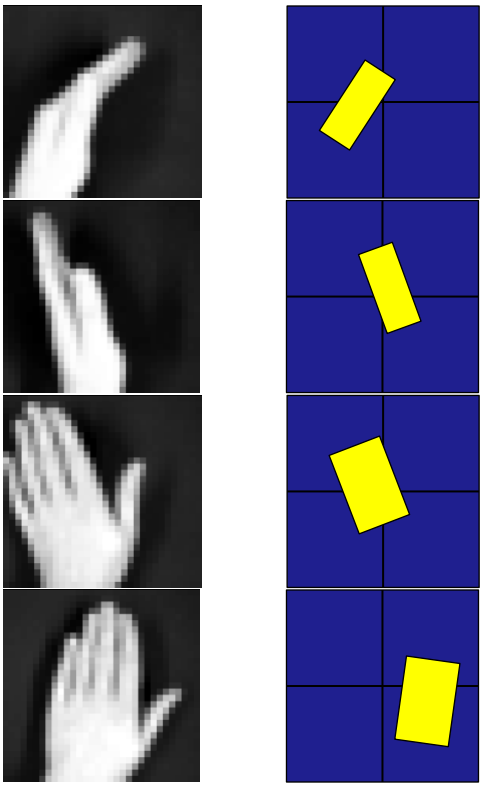
Part of the Sega Saturn game, Nights, involves steering a sprite flying through a magical world. We wanted the user to control the sprite by simple motions or pointing gestures.

We had the 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, the input to the moments calculation may be a quantity based on motion. To control the movement of the sprite we calculated the center of mass of a motion energy image, the absolute value of the difference between successive video frames (see Fig. 3). The response to pointing gestures is robust and immediate, allowing natural control of

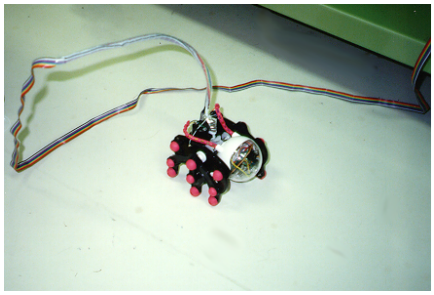
[1] We thank Sega for providing game interface information.

the game character with easy gestures.

We applied the moments analysis hierarchically, over the image and its four quadrants, to analyze body positions for control of the game Magic Carpet (see Fig. 4). The player can steer his magic carpet in any direction by leaning or walking in the different directions; he can fire a spell by holding out his arm. Because the player's 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; the shape-based method requires a uniform background.



(a)



(b)

Figure 1: (a) Hand images and equivalent rectangles, having the same first and second order moments. We measure X-Y position, orientation, and projected width from the rectangle. We used image moments to control the motion of a toy robot, (b).

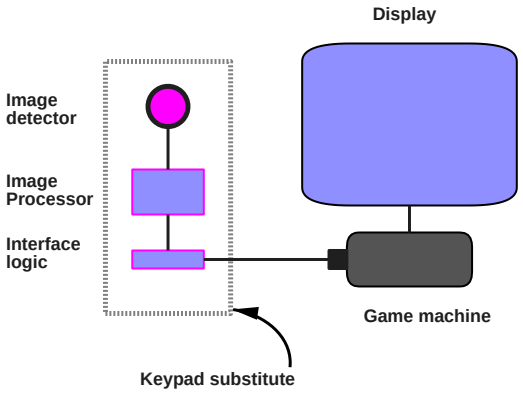


Figure 2: System block diagram of interactive game machine.

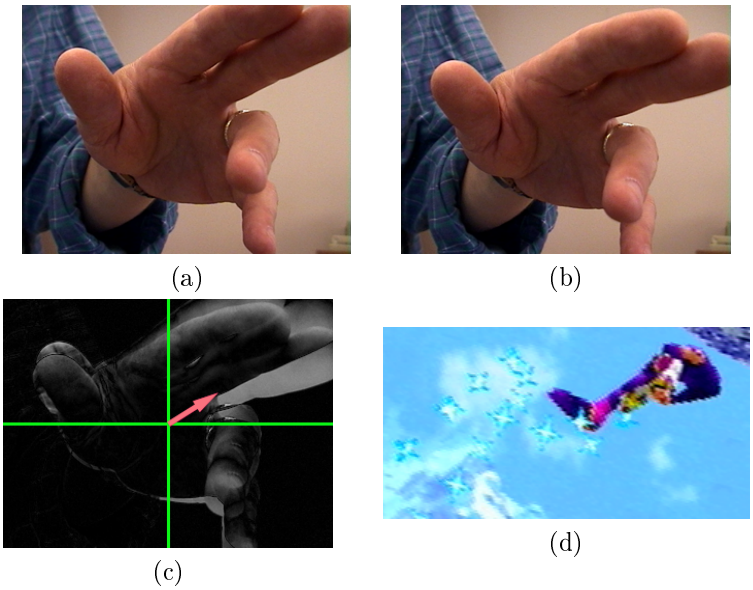


Figure 3: (a) and (b): Two frames of camera input. (c) Absolute value of temporal difference image (a - b). Its center of mass (arrow) controls direction of flight of sprite, (d).

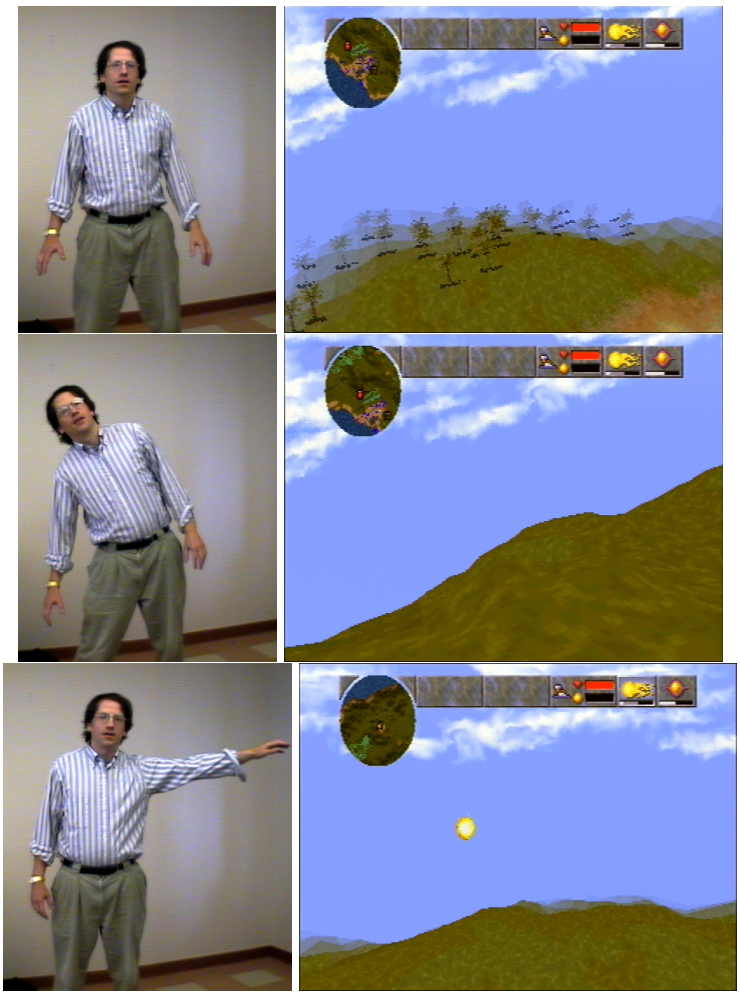


Figure 4: A hierarchy of computed image moments was used to analyze the shape and orientation of the player for the game, Magic Carpet. The player controls the magic carpet flight by leaning or walking. The player can raise his arm to fire a spell.

2.1 Sidebar: Image moments

Image moments [7] 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:

$$\begin{aligned} M_{00} &= \sum_x \sum_y I(x,y) & M_{11} &= \sum_x \sum_y xy\, I(x,y) \\ M_{10} &= \sum_x \sum_y x\, I(x,y) & M_{01} &= \sum_x \sum_y y\, I(x,y) \\ M_{20} &= \sum_x \sum_y x^2\, I(x,y) & M_{02} &= \sum_x \sum_y y^2\, I(x,y) \end{aligned} \quad (1)$$

We can find the position, x_c, y_c, orientation θ, and dimensions l_1 and l_2 of an equivalent rectangle which has the same moments as those measured in the image [7]. Those values give a measure of the hand's position, orientation, and aspect ratio. We have:

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}} \quad (2)$$

Define the intermediate variables a, b, and c,

$$\begin{aligned} a &= \frac{M_{20}}{M_{00}} - x_c^2 \\ b &= 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \\ c &= \frac{M_{02}}{M_{00}} - y_c^2. \end{aligned} \quad (3)$$

We have (c.f. [7]):

$$\theta = \frac{\arctan(b, (a-c))}{2} \quad (4)$$

and

$$\begin{aligned} l_1 &= \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \\ l_2 &= \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \end{aligned} \quad (5)$$

The image moments can be calculated from three projections of the image [7]. Let the vertical, horizontal, and diagonal projections be

$$V(x) = \sum_y I(x,y), \quad (6)$$

$$H(y) = \sum_x I(x,y), \quad (7)$$

and

$$D(t) = \sum_s I\left(\frac{t-s}{\sqrt{2}}, \frac{t+s}{\sqrt{2}}\right). \quad (8)$$

Then the image moments can be written as [7]:

$$\begin{aligned} M_{00} &= \textstyle\sum_x V(x) \\ M_{11} &= \textstyle\sum_t t^2 D(t) - \frac{M_{20}}{2} - \frac{M_{02}}{2} \\ M_{10} &= \textstyle\sum_x x\, V(x) & M_{01} &= \textstyle\sum_y y\, H(y) \\ M_{20} &= \textstyle\sum_x x^2\, V(x) & M_{02} &= \textstyle\sum_y y^2\, H(y). \end{aligned} \quad (9)$$

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

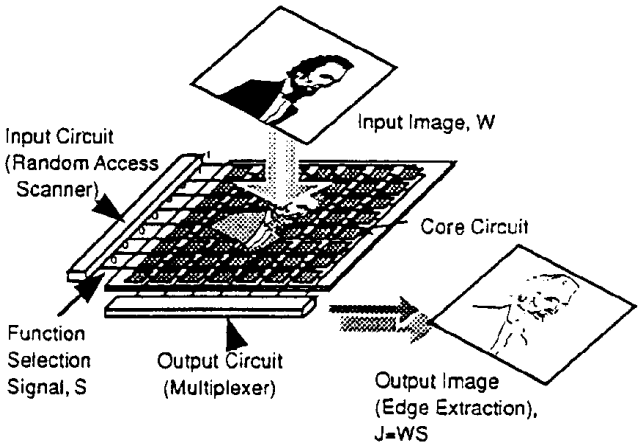
2.2 Sidebar: The artificial retina chip

We have developed an image detector which 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 [9]. Figure 5 shows the element of the AR chip: a 2-D 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 which allows for high detection sensitivity. This structure also realizes nondestructive readout of the image, essential for the image processing. We have built detector arrays ranging in resolution from 32x32 to 256 x 256 pixels; a 32x32 detector array was adequate for the game applications presented here. That chip size is 6.5 by 6.5 square millimeters, with 150 by 150 micron sized pixels, using a 1 micron fabrication process. On-chip image processing is obtained from an input image at every time step.

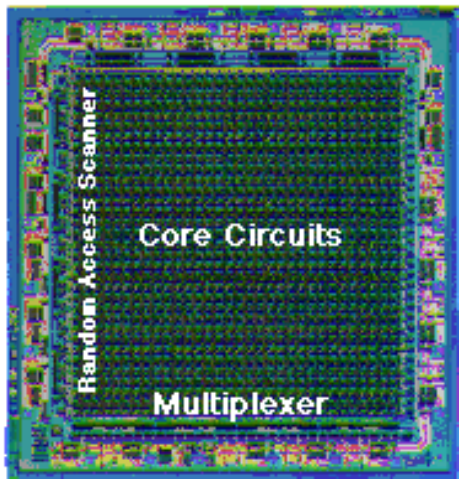
The image processing of the artificial retina can be expressed as a matrix equation. In Fig. 5, the input image projected onto the chip is the weight matrix W. All VSPC's have three electrodes. A direction sensitivity electrode, connected along rows, yields the sensitivity control vector, S. The VSPC sensitivities can be set to one of $(+1, 0, -1)$ at each row. An output electrode is connected along columns, yielding an output photocurrent which is the vector product, $J = WS$. The third electrode is used to reset 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 1000 Hz. This detector can be manufactured on the same equipment used to process DRAM's and thus may cost less to manufacture than conventional CCD's.

We have integrated this detector/processor chip into an inexpensive *AR module*, which contains a low-resolution (32x32) AR detector chip, support and interface electronics, and a 16 bit 10 MHz micro-processor. Taken together, the microprocessor and AR chip can perform general image processing operations quickly. The module is 8 x 4 x 3 cm, and the chip can be fabricated



(a)



(b)

Figure 5: (a) Schematic structure of the artificial retina chip. An array of variable sensitivity photodetector cells allows image detection, linear filtering, and projection. (b) Photomicrograph of the chip.

at a considerably lower cost than CCD's because the chip fabrication is CMOS-based.

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 micro-processor speed. For the 10 MHz micro-processor of the AR module and 32x32 resolution, the calculations would take 10 msec per image on the microprocessor alone, but only 0.3 msec per image using the microprocessor and artificial retina chip.

3 Shape recognition

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

If the hand signals fall in a predetermined set, and the camera views a close-up of the hand, we may use an example-based approach, combined with a simple method to analyze hand signals called orientation histograms. These histograms summarize how much of each shape is oriented in each possible direction, independent of the position of the hand inside the camera frame. The computation involves taking spatial derivatives, followed by a non-linearity 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 particular hand shape. The computer forms and stores the corresponding orientation histograms. In the run phase, the computer compares the orientation histogram of the current image with each of the stored templates and selects the category of the closest match, or interpolates between templates, as appropriate. This method is insensitive to small changes in the size of the hand, but is sensitive to changes in hand orientation. For greater robustness, the user may show several examples and the computer can use the closest matching example to the test image.

We have implemented several interactive graphics applications which rely on orientation histograms for hand gesture recognition [5]. Figure 6 shows a computer graphic crane that we can control by hand signals. The system is first trained on the 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 5 trained categories gives the user feedback for implementing consistent gestures and helps to debug any miscategorizations.

We used the same recognition engine in an interactive game of `rock, scissors, paper`, Figure 8. A computer graphic "robot hand" plays the game against the user. This indicates when the user should make the gesture, allowing 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 have showed these systems to many users, most of whom can use them without practice. Generally, the system must be trained for each user; in the run phase, the response is reliable and fast.

We have observed several conditions where the user is not satisfied with the gesture classification, illustrated in Fig. 7. (a) and (b) show two images which users feel should represent the same gesture. However, their orientation histograms are very different, illustrated in (c). This problem can be addressed by providing multiple training images for the same gesture. Some different gestures have very similar orientation histograms. (d) and (e) show an example of this, with the histograms overlaid in (f). One must choose a vocabulary of gestures that avoids such confusable pairs. Finally, the hand must dominate the image for this simple statistical technique to work. (g) and (h) show images where the hand is a small part of the image. Even though user has very different hand positions, the orientation histograms of the two images are very similar, (i). 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. One can also use special hardware, such as the artificial retina chip described above, for the low-level image processing tasks, see sidebar (*Artificial Retina*).

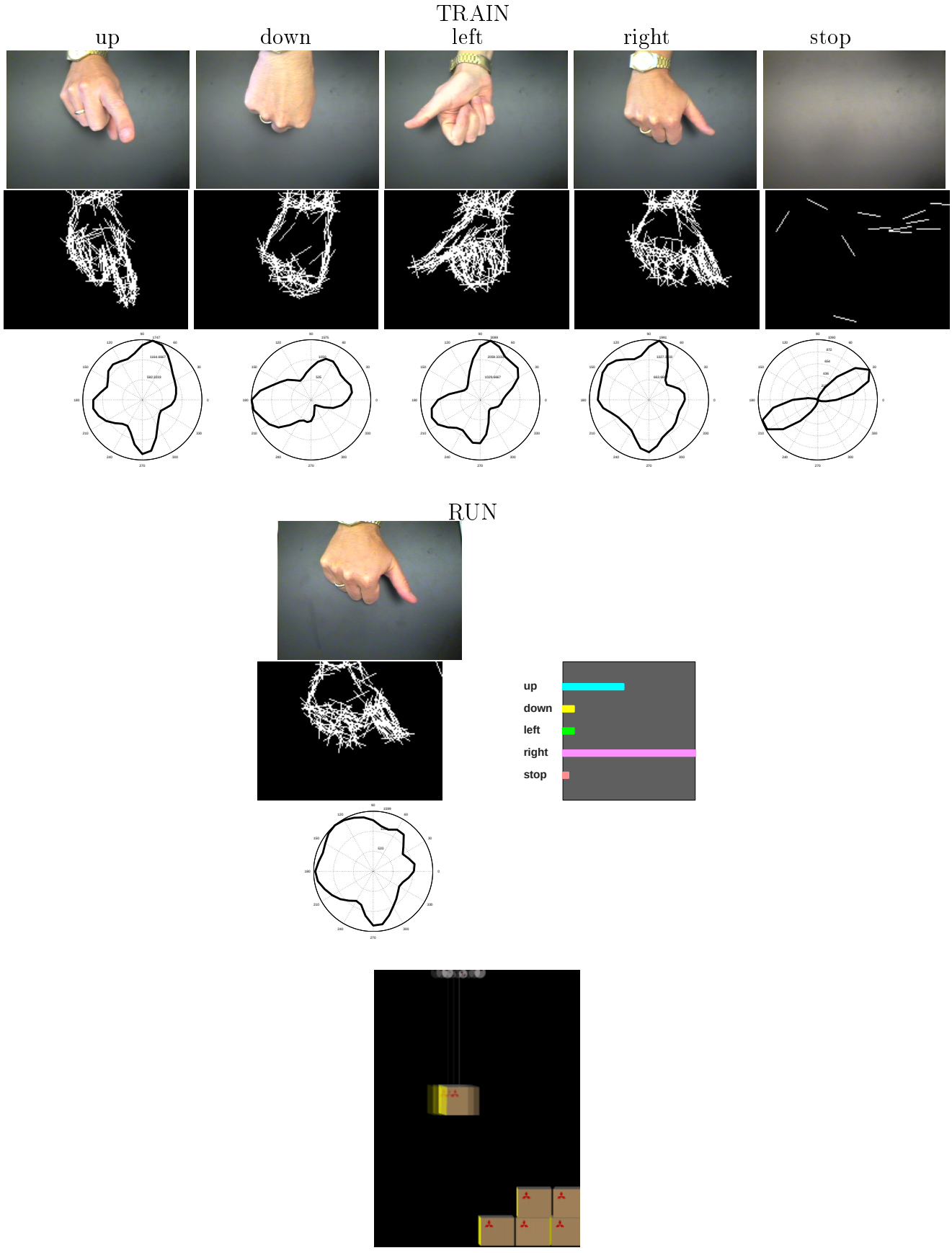


Figure 6: Orientation histograms in an example-based hand gesture recognition system. Top: Training images, with their orientation maps and orientation histograms. Middle: In the run phase, the computer compares the orientation histogram of the current image with those of the training images. Graph showing inverse distances gives user feedback on performance. Bottom: Here, the hand signals control a toy crane.

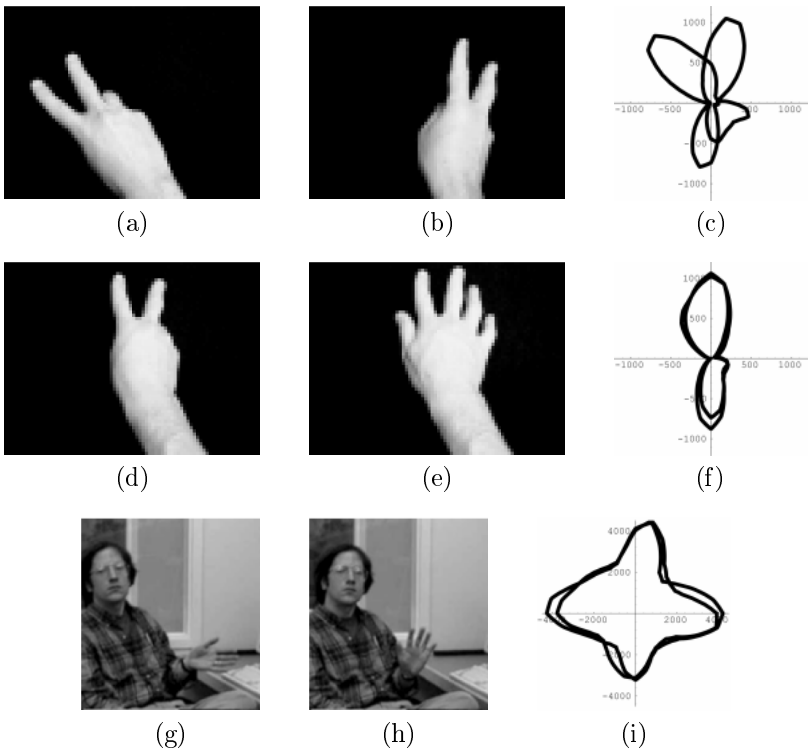


Figure 7: Problem images for the orientation histogram based gesture classifier. Users may feel that (a) and (b) represent the same gesture, but their orientation histograms are different, shown overlaid in (c). A remedy is to provide training images of the gesture at various orientations.
Sometimes small changes in the image can cause large semantic differences, while changing the orientation histograms little. Users classify (d) and (e) as different gestures, yet their orientation histograms are nearly the same, (f). One has to construct a gesture vocabulary which avoids gestures with similar orientation histograms.
Finally, for this simple statistical technique to work, the hand must dominate the image. If it does not, then even large changes in the hand pose can cause negligible changes to the orientation histogram (g) – (i).

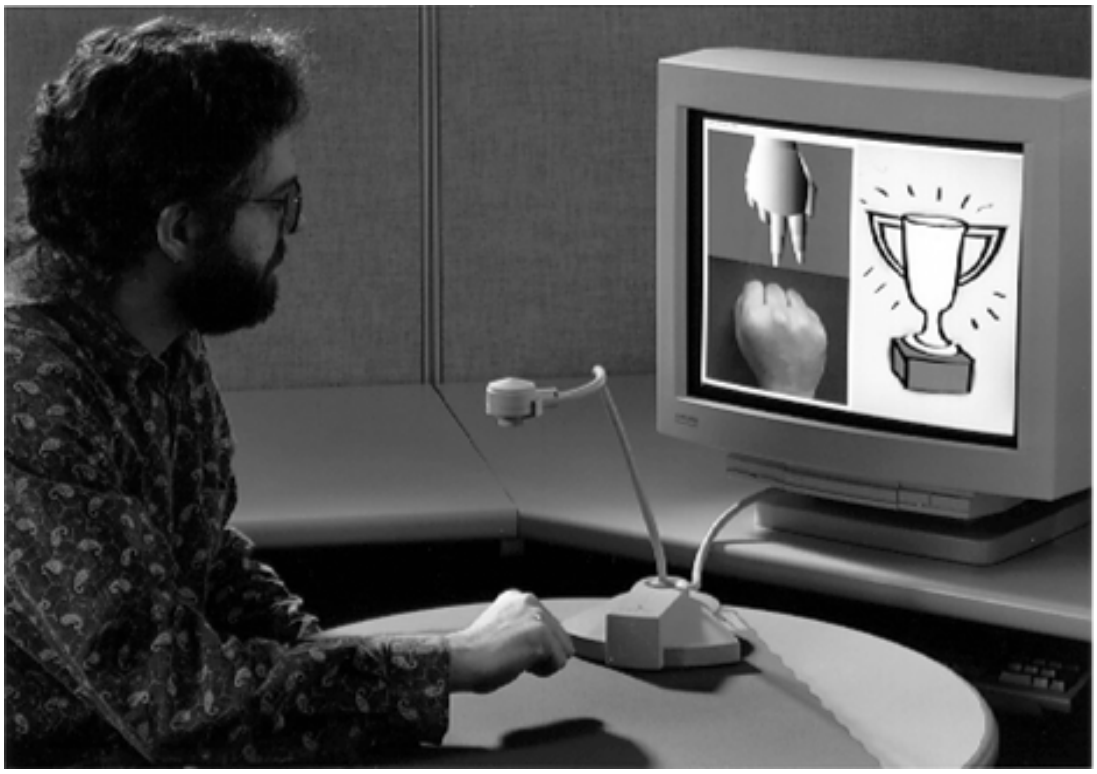


Figure 8: Rock, scissors paper game, based on orientation histograms.

3.1 Sidebar: Orientation histograms

The desire for lighting and position invariance motivates the orientation histogram representation. Figure 9 shows a comparison of a pixel representation and an orientation representation for a picture of a hand under two different lighting conditions. The pixel values of the hand vary considerably with lighting; the orientation values remain fairly constant. One can calculate the local orientation using image gradients. The phase angle of a complex number defined by horizontal and vertical image pixel differences yields the local orientation angle, θ, as a function of position x and y:

$$\theta(x,y) = \arctan[I(x,y) - I(x-1,y), I(x,y) - I(x,y-1)] \quad (10)$$

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 rather drastic step of ignoring position altogether, and simply tabulating a histogram of how often each orientation element occurred in the image. Clearly, this throws out information and some distinct images will be confused by their orientation histograms. In practice, however, one can easily choose a set of training gestures with substantially different orientation histograms from each other (e.g. Fig. 6).

We form a vector, Φ, of N elements, with the ith element showing the number of orientation elements $\theta(x,y)$ between the angles $\frac{360^\circ}{N}(i - \frac{1}{2})$ and $\frac{360^\circ}{N}(i + \frac{1}{2})$:

$$\Phi(i) = \sum_{x,y} \begin{cases} 1 & \text{if } |\theta(x,y) - \frac{360^\circ}{N} i| < \frac{360^\circ}{N} \\ 0 & \text{otherwise} \end{cases} . \quad (11)$$

To reduce noise and allow interactions between neighboring orientations, we average together adjacent histogram bins. We used $N = 36$ bins for the applications shown here. Simple Euclidean distance, $(\Phi_1 - \Phi_2)^2$, provides a distance measure between the two images with orientation histograms Φ_1 and Φ_2. Figure 10 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 [10] proposed these as a method to analyze the shapes of binary images; we apply them to applications using grey scale images [5].

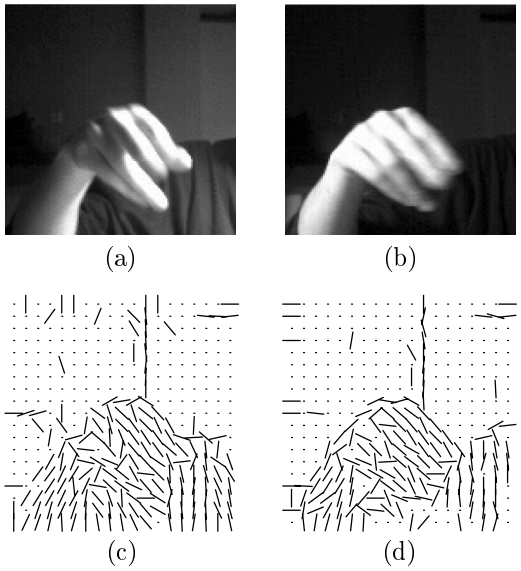


Figure 9: Orientation maps, (c) and (d), are generally more robust to lighting changes than are images of pixel intensities.

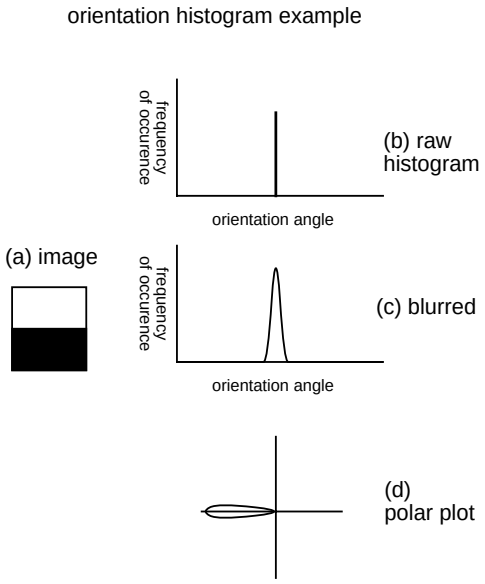


Figure 10: Simple illustration of orientation histogram. An image of a horizontal edge (a) has only one orientation at a sufficiently high contrast. Thus the raw orientation histogram (b) has counts at only one orientation value. To allow neighboring orientations to sense each other, we blur the raw histogram, giving (c). (d) shows the same information, plotted in polar coordinates. We define the orientation to be the direction of the intensity gradient, plus 90°.

4 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. The game involves the Olympic events of the decathlon. 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 allow the user to pantomime stationary versions of the athletic events in front of the camera, by running or jumping in place. We hoped this would add an extra dimension to the game and make it more engaging.

Figure 11 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 at the same time.

Recognition of these actions in a general context would be very challenging. But knowing the game context greatly simplifies the visual recognition. The graphics application tells us which event the player is performing, and we only have to choose between a few different motions the player should be performing, or to estimate timing or rate parameters. These are much easier problems to solve. For the 110 meter hurdles, the vision algorithm has to choose whether the player is running, or raising his arms to jump over a hurdle. If he is running in place, the algorithm has to estimate how fast.

There are many methods to analyze optical flow, but relative to our low-cost, high-speed requirements, these can be too slow on conventional machines. Our motion analysis does not need to be detailed, but needs to be extremely fast.

We have developed a fast "optical flow" algorithm, 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 use simple measurements derived from the optical flow to estimate the relevant motion parameters. For example, for the 110 meter hurdles, we track the frame averages of the horizontal and vertical components of motion. The frequency of the alternation of the horizontal velocity indicates how fast the player is running in place. When the average vertical velocity exceeds a threshold, that indicates a jump command.

The impression on the player of this simple processing is that the computer understands his physical gestures. We have demonstrated this game at various public venues, including at the computer trade show Comdex, 1996. A stationary background was used, to give the most reliable recognition. We used the artificial retina chip in detector mode. Players can almost always control the characters well on their first attempts, and became immediately engaged in the game itself.

4.1 Sidebar: Fast optical flow

For an approximation to the optical flow field, we have developed a fast algorithm, which 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 is as follows:

1. Subtract the current frame from the previous one, yielding a temporal difference image.

2. For pixels where the temporal difference is non-zero, enumerate the possible motion directions consistent with the local image measurements. Consider the the 1-d motion shown in Fig. 14. From this example, we derive two rules for motion direction estimation:

   (a) If the temporal difference is negative, the motion direction is toward the adjacent pixel with higher luminance in the current frame.

   (b) If the temporal difference is positive, the motion direction is toward the adjacent pixel with lower luminance in the current frame.

3. Apply the 1-d 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 11: Top: Participants playing Decathlete game. Bottom: image mosaic showing player, Decathlete game display, and artificial retina module.

Fig. 14 illustrates our algorithm for the case of a 4x4 pixel moving square. Each subfigure corresponds with one enumerated step of the algorithm above. When the square moves (Fig. 14(a)), there exist some pixel value changes. Fig. 14 (b) represents the change of each pixel value by taking their difference of the frame interval. Fig. 14(c) is a set of orientation vectors at each pixel, and motion vectors are shown in Fig. 14(d). After spatial smoothing, the optical flow pattern is as in Fig. 14(e)

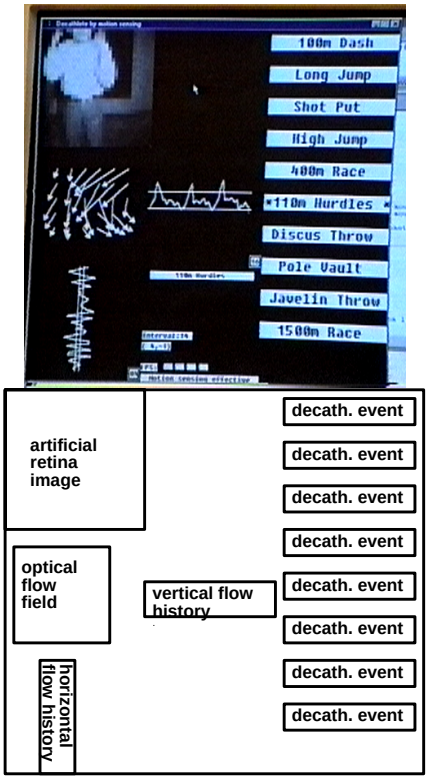


Figure 12: Visual display of analysis of Decathlete game.

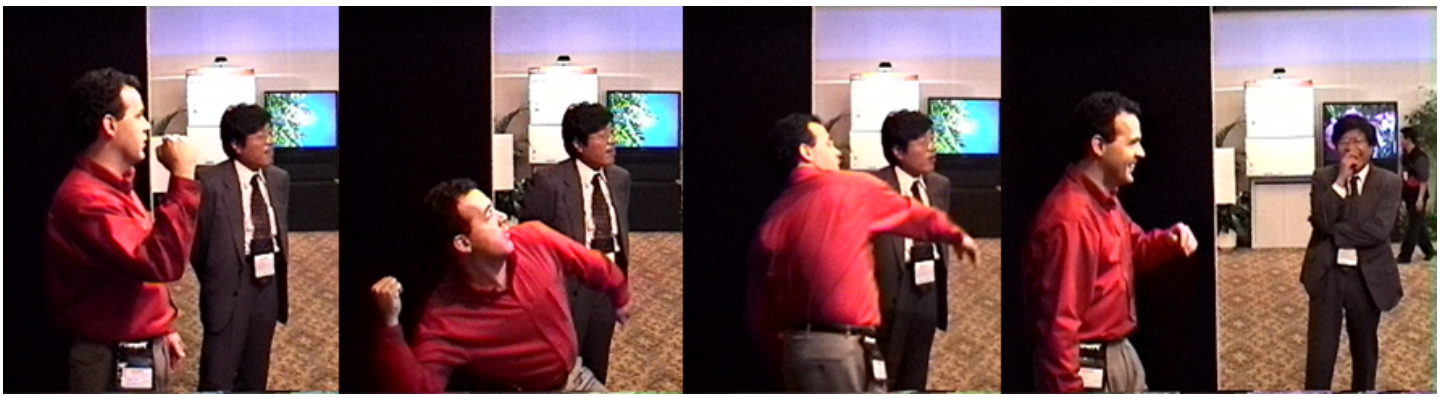


Figure 13: Example of user playing another Decathlon event, the javelin throw. The computer's timing of the set and release for the javelin is based on when the integrated downward and upward motion exceeds predetermined thresholds.

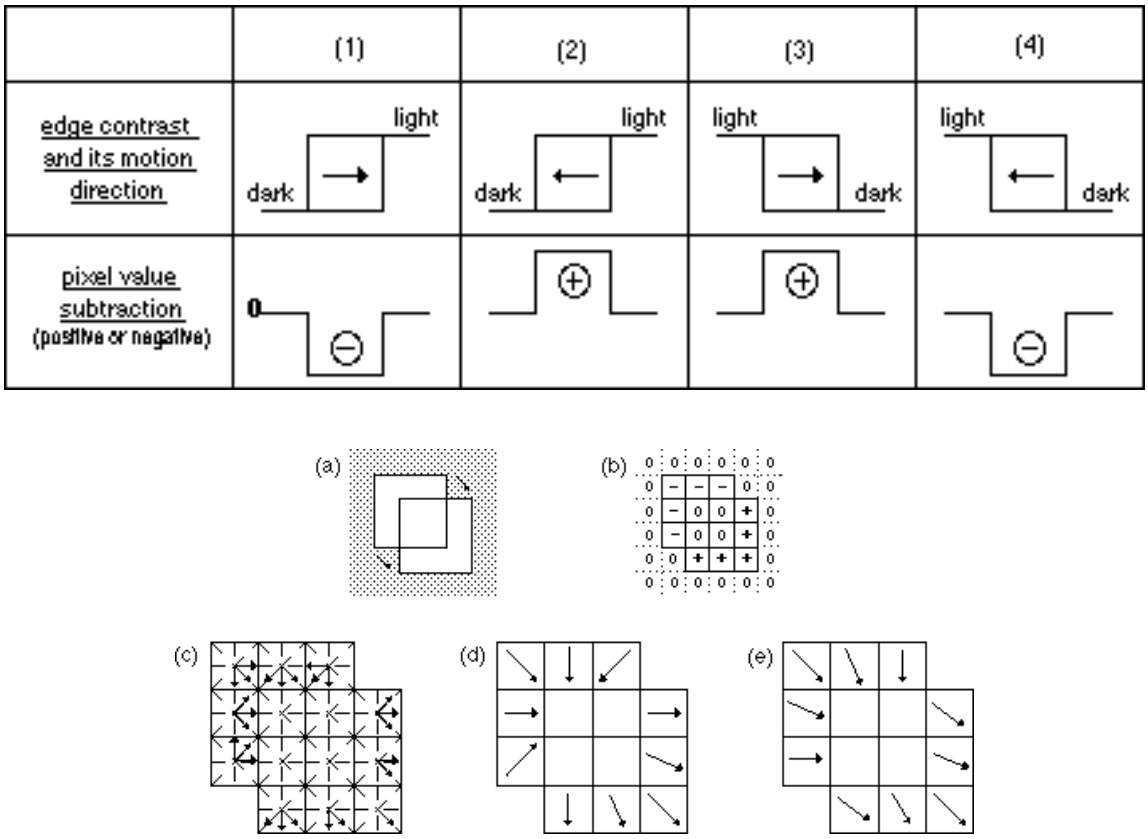


Figure 14: Fast motion estimation algorithm. Top: Showing how contrast direction and motion direction interact to yield temporal difference pixel values (algorithm, step 2). Bottom: Steps in the algorithm to calculate optical flow. (b), (c), (d), and (e) correspond to algorithm steps 1, 3, 4, and 5, respectively.

5 Small object tracking

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

The target application is to control a television set by hand signals, replacing a remote control. This application forces 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 want to give a broad set of commands to the television set by hand signals, yet we don't want to require an intensive training period before one can control the television. On the machine side, recognition of a broad set of hand gestures made within a complex, unpredictable visual scene, such as a living room, is difficult, beyond the realm of present day vision algorithms.

We addressed both these design constraints by exploiting the ability of the television screen for graphical feedback [6]. Our interface design is simple (see Fig. 16). 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. The user adjusts the various television controls by moving the hand icon on top of the on-screen 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 sidebar: `normalized correlation`). We examine the fit of a hand template to every position in the analyzed image. The location of maximum correlation gives the position of the candidate hand; the value of that correlation indicates how likely the image region is to be a hand.

To simplify the processing, we just 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 - 10 feet from the television. To increase the processing speed, we restricted the field of view of the television's camera to 15° when initially searching for the hand, and 25° in tracking mode. We used a running temporal average of the image in order to subtract out stationary objects. Nonetheless, best results are achieved when the background provides good contrast 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 [2].

The working demonstration allowed us to in-

formally 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 tiring. 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.

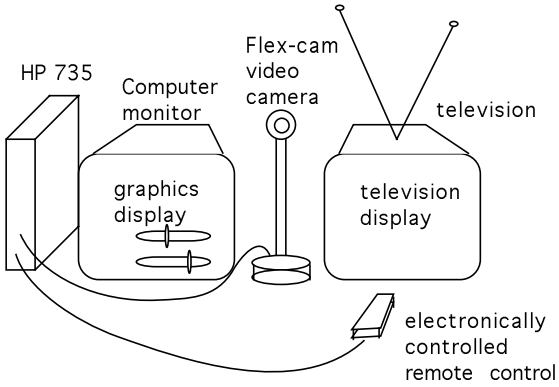
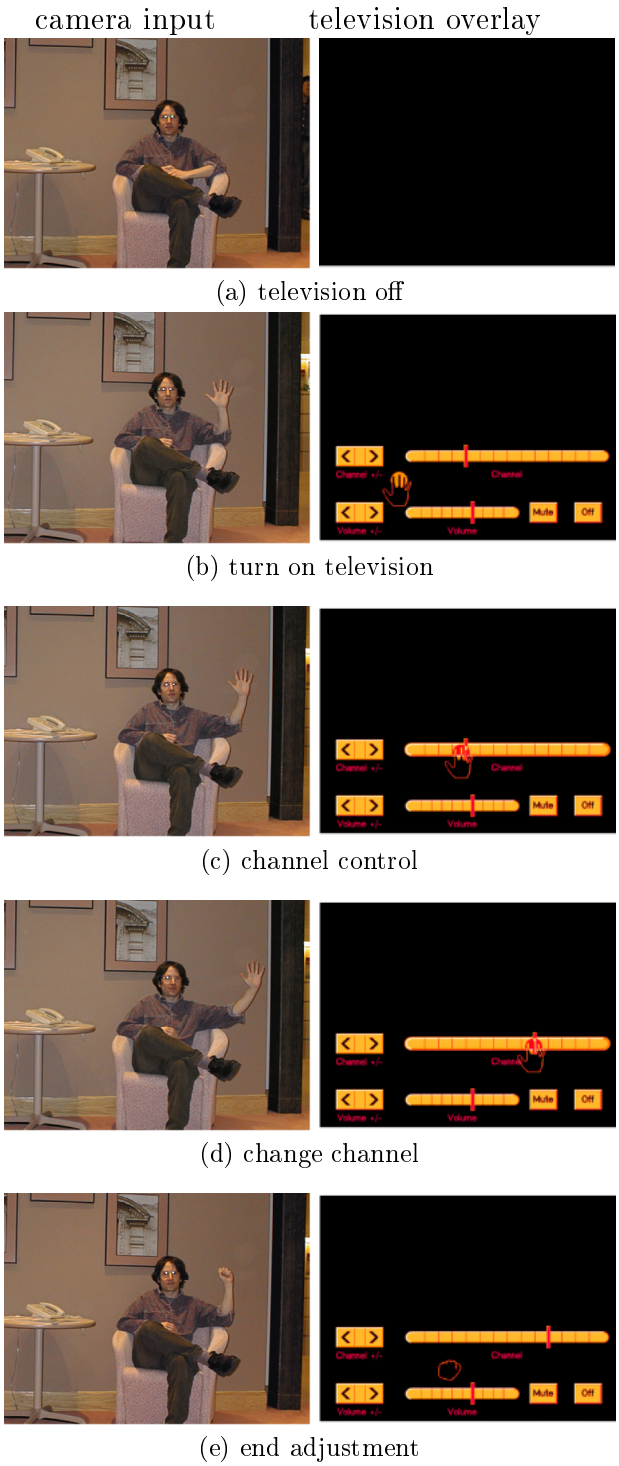


Figure 15: Block diagram of prototype of television set controlled by hand gestures. While the prototype uses two display screens, the graphics could be overlaid directly on the television image.



(a) television off

(b) turn on television

(c) channel control

(d) change channel

(e) end adjustment

Figure 16: 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.

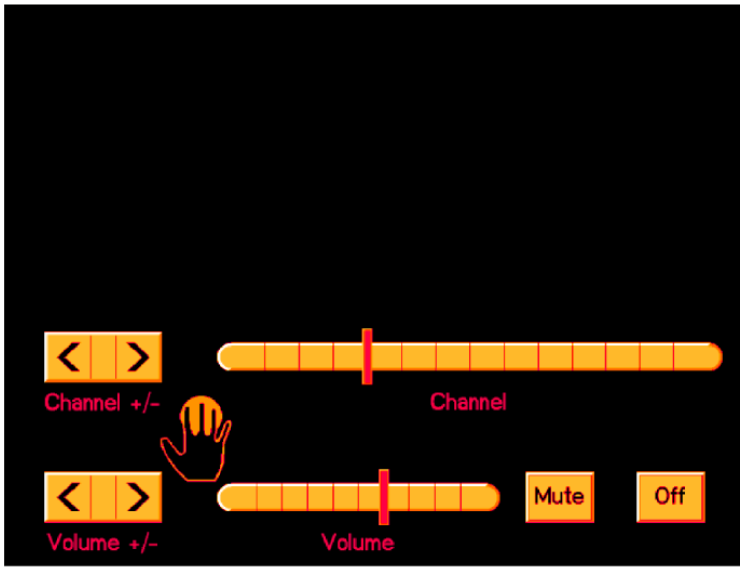


Figure 17: For television set controlled by hand gestures. Under vision input, the hand icon tracks the user's hand, allowing him to use his hand like a computer mouse.

5.1 Sidebar: 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 what is called 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, $\vec{a}$. 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, $\vec{b}$. The normalized correlation for that particular offset of the target pattern within the image is simply the cosine of the angle between the vectors $\vec{a}$ and $\vec{b}$: $\frac{\vec{a} \cdot \vec{b}}{\sqrt{(\vec{a} \cdot \vec{a})(\vec{b} \cdot \vec{b})}}$. Figure 18 shows the normalized correlation between an image and a hand template. Note the peak correlation intensity at the true position of the hand.

(a) (b) (c)

Figure 18: (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.

6 Summary

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 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.

References

[1] R. Bajcsy. Active perception. *IEEE Proceedings*, 76(8):996–1006, 1988.

[2] A. Blake and M. Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *Proc. SIGGRAPH 94*, pages 185–192, 1994. In *Computer Graphics*, Annual Conference Series.

[3] T. Darrell, P. Maes, B. Blumberg, and A. P. Pentland. Situated vision and behavior for interactive environments. Technical Report 261, M.I.T. Media Laboratory, Perceptual Computing Group, 20 Ames St., Cambridge, MA 02139, 1994.

[4] I. Essa, editor. *International Workshop on Automatic Face- and Gesture- Recognition.* IEEE Computer Society, Killington, Vermont, 1997.

[5] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In M. Bichsel, editor, *Intl. Workshop on automatic face- and gesture-recognition*, Zurich, Switzerland, 1995. Dept. of Computer Science, University of Zurich, CH-8057.

[6] W. T. Freeman and C. Weissman. Television control by hand gestures. In M. Bichsel, editor, *Intl. Workshop on automatic face- and gesture-recognition*, Zurich, Switzerland, 1995. Dept. of Computer Science, University of Zurich, CH-8057.

[7] B. K. P. Horn. *Robot vision.* MIT Press, 1986.

[8] M. Krueger. *Artificial Reality.* Addison-Wesley, 1983.

[9] K. Kyuma, E. Lange, J. Ohta, A. Hermanns, B. Banish, and M. Oita. *Nature*, 372(197), 1994.

[10] R. K. McConnell. Method of and apparatus for pattern recognition. U. S. Patent No. 4,567,610, Jan. 1986.