

# Computer vision for computer games

W. T. Freeman †, K. Tanaka §, J. Ohta § and K. Kyuma §

MERL, a Mitsubishi Electric Research Lab. †  
201 Broadway  
Cambridge, MA 02139 USA  
e-mail: freeman@merl.com

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

From: IEEE 2nd Intl. Conf. on Automatic Face  
and Gesture Recognition,  
Killington, VT, October, 1996.

## Abstract

The appeal of computer games may be enhanced by vision-based user inputs. The high speed and low cost requirements for near-term, mass-market game applications make system design challenging. The response time of the vision interface should be less than a video frame time and the interface should cost less than \$50 U.S.

We meet these constraints with algorithms tailored to particular hardware. We have developed a special detector, called the artificial retina chip, which allows for fast, on-chip image processing. We describe two algorithms, based on image moments and orientation histograms, which exploit the capabilities of the chip to provide interactive response to the player's hand or body positions at 10 msec frame time and at low-cost. We show several possible game interactions.

## 1 Introduction

Computer games are a popular consumer electronics item. The game players find it captivating to interact with games via joysticks, buttons, trackballs, or wired gloves. They may find it even more engaging to interact through natural, unencumbered hand or body motions. A computer vision-based user interface could provide these capabilities. Computer games represent a possible mass-market application for computer vision.

These applications present unique challenges. The 33 msec delay time of NTSC video is too slow; the system should respond within 10 msec or less to avoid noticeable delays. The system must be low-cost, roughly comparable with existing game user input components which cost several tens of dollars, and it should be robust. To be successful, the vision interface should add some new dimension to the game itself.

Some features of computer games lessen the design difficulties. In most games, the user can exploit immediate visual feedback to reach the desired effect. If the player is leaning to make a turn in the game, and he (or she) sees that he isn't turning enough, he can lean more. The structure of the games provides a context which can allow dramatic, appropriate responses from simple visual measurements. The vision system may merely track the position of the visual center of mass of the player, but the game can turn that into running, jumping or crouching, depending on position and game context.

There has been much recent work on computer vision analysis of faces and gestures (e.g. [3, 12, 4, 2]). The focus of this research has been on high performance algorithms, not cost optimization. While some systems perform at 10 or 30 Hz on workstation or Pentium systems, this is still too slow and far too expensive for the game requirements described above. Near-term game applications require simpler algorithms.

Computer game applications represent a niche for high speed, low cost vision systems. We developed hardware and algorithms for a vision system aimed at this niche. Our approach has been to combine simple algorithms with a flexible and inexpensive detector/processing module.

## 2 The Hardware

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 [10]. Figure 1 shows the elements 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 [7]. The VSPC consists of a *pn* photo-diode and a differential amplifier which allows for high detection sensitivity of either positive or negative polarity. This structure also realizes nondestructive readout of the image, essential for the image processing. The detector arrays can range in resolution from 32x32 to 256 x 256 pixels; for this paper we assume a 32x32 detector array.

The image processing of the artificial retina can be expressed as a matrix equation. In Fig. 1, 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. This hardware can sense the raw image and execute simple linear operations such as local derivatives and image projections.

We have integrated this detector/processor chip into an inexpensive *AR module*, which contains a low-resolution (32x32) A. R. detector chip, support and interface electronics, and a 16 bit 1MHz micro-processor. The module is 8 x 4 x 3 cm and is inexpensive enough to cost only several tens of dollars.

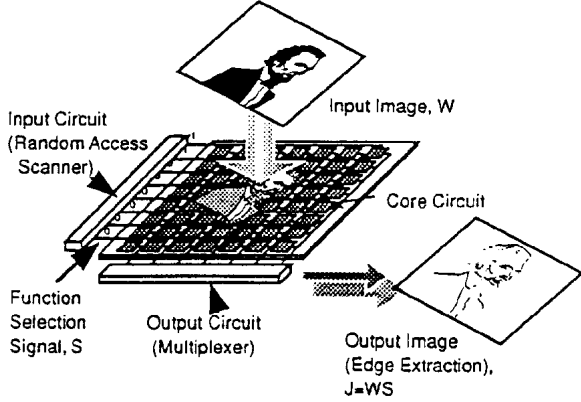


Figure 1: Schematic structure of artificial retina chip.

### 3 Algorithms

Our goal is to infer useful information about the position, size, orientation, or configuration of the player's body or hands. We seek fast, reliable algorithms for the inexpensive AR processor module.

We have chosen two algorithms. One uses image moments to calculate an equivalent rectangle for the current image. Another uses orientation histograms to select the body pose from a menu of templates. The first exploits the image projection capabilities of the AR module; the second uses its ability to quickly calculate  $x$  and  $y$  derivatives.

#### 3.1 Image Moments

Image moments [9, 1] provide useful summaries of global image information, and have been applied to shape analysis or other tasks, often for binary images. The moments involve sums over all pixels, and so are robust against small pixel value changes. Within the structure of a computer game, they can also provide sufficient information for the computer to reliably interpret control inputs from the user's body position. Characteristics of the AR chip allow fast calculation of these moments.

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

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

We can find the position,  $x_c, y_c$ , orientation  $\theta$ , and dimensions  $l_1$  and  $l_2$  of an *equivalent rectangle* which has the same moments as those measured in the image [9]. Those values give a measure of the hand's position, orientation, and aspect ratio. We have:

$$x_c = \frac{M_{10}}{M_{00}} \quad y_c = \frac{M_{01}}{M_{00}} \tag{2}$$

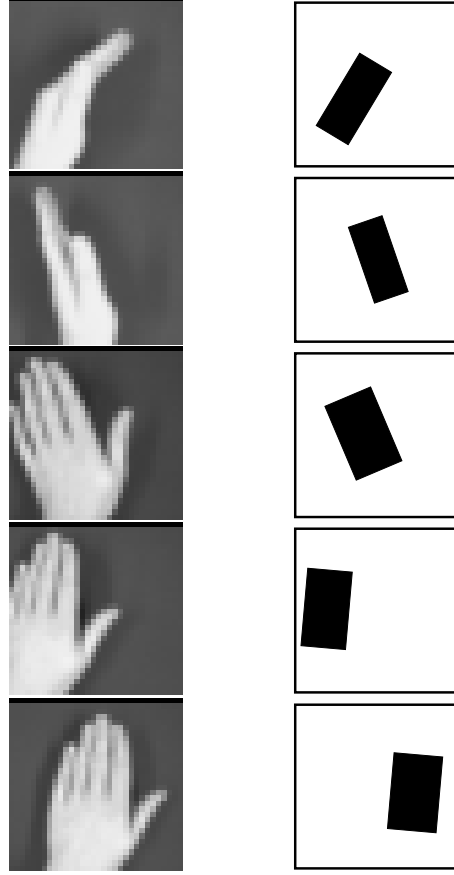


Figure 2:  $32 \times 32$  input images of user's hand, and the equivalent rectangle having the same first and second order moments as those of the image. X-Y position, orientation, and projected width is measured from the rectangle. (Projected height is also measured, but with the hand extending off the picture as shown here, height is redundant with the vertical position of the center of mass).

Define the intermediate variables  $a, b$ , and  $c$ ,

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

We have (c.f. [9]):

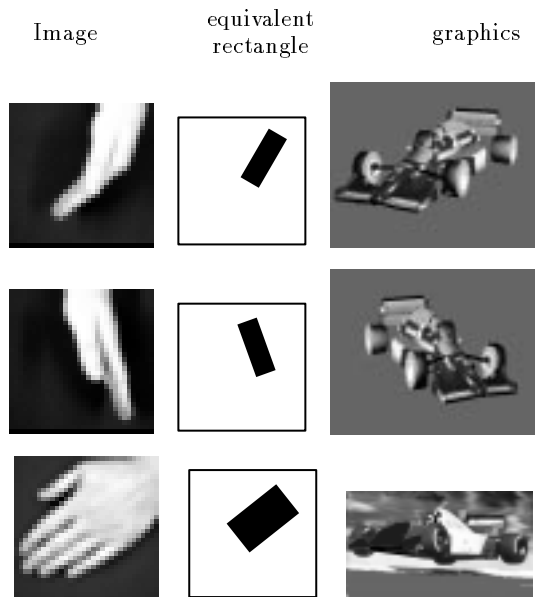
$$\theta = \frac{\arctan(b, (a - c))}{2} \tag{4}$$

and

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

The extracted parameters are independent of the overall image intensity.

Figure 2 shows an image of a hand, as it could appear within a “playing area” of a game machine, and the corresponding equivalent rectangle calculated for each image from the image moments. Note that the rectangle accurately reflects four variables: the x-y position of the hand, its orientation, and its width. In the context of a game, these image moment measurements provide rich opportunities for control. Figures 3 and 4 illustrate two example games based on equivalent rectangle measurements, for car racing and skateboard games.



**Figure 3:** Left: 32x32 pixel input images of player’s hand. Middle: a rectangle having the equivalent image moments as the input image. The x-y position, orientation, and length and width of this rectangle represent game control parameters. Right: Possible game responses to hand inputs. Car orientation and position follow the hand; the hand’s apparent width controls the throttle.

### 3.1.1 Fast Calculation of Image Moments

The image moments can be calculated quickly from three projections of the image [9], and we exploit that fact to speed up the processing with the AR chip.

Let the vertical, horizontal, and diagonal projections be

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

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

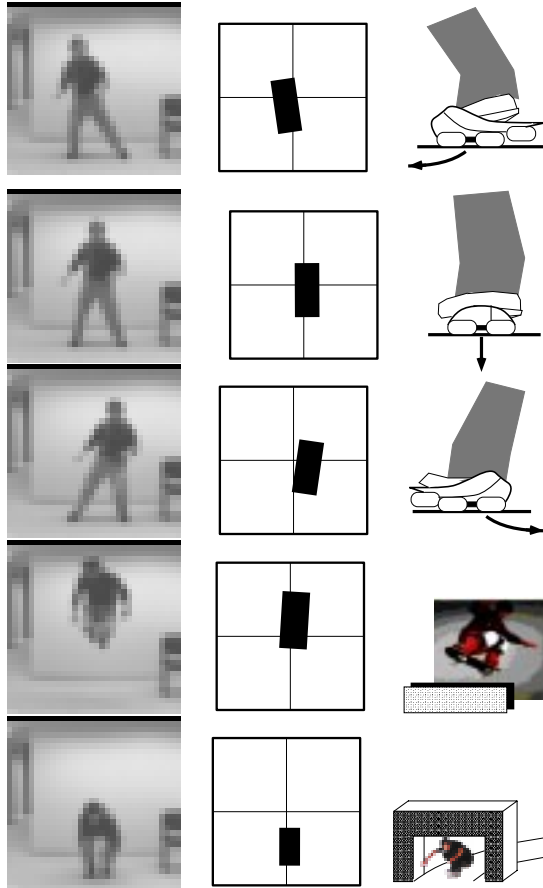
and

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

Then the image moments are [9]:

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

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



**Figure 4:** Left: Input images for a skateboarding game. Middle: equivalent rectangle for each image (after background subtraction and clipping to positive values) with crosshairs superimposed for reference. Right: Possible game response to rectangle states. Horizontal position determines right/left steering. Jumps and crouches can be inferred from vertical position and rectangle height.

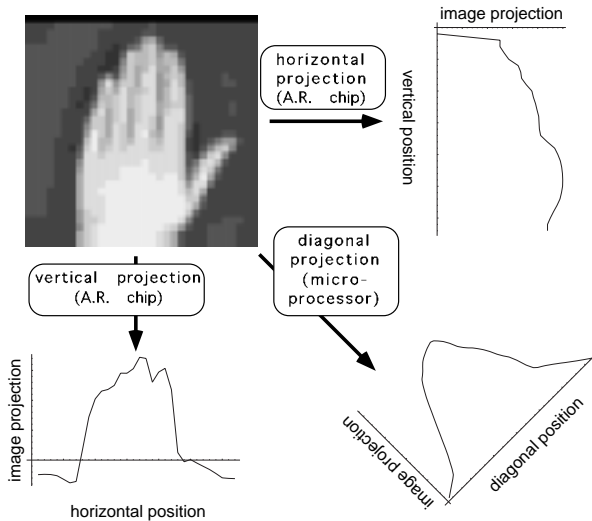
$$M_{10} = \sum_x x V(x) \quad M_{01} = \sum_y y H(y)$$

$$M_{20} = \sum_x x^2 V(x) \quad M_{02} = \sum_y y^2 H(y). \quad (9)$$

The horizontal and vertical image projections can be performed on the artificial retina detector, saving processor time. The savings depend on the image resolution and micro-processor speed. For the 1 MHz. micro-processor of the AR module and 32x32 resolution, the calculations would take 20 msec per image on the microprocessor alone, but only 10 msec per image using the microprocessor and artificial retina chip. The on-chip processing of the artificial retina chip brings the algorithm into the range in speed and cost need for today’s games.

### 3.2 Orientation histograms

The artificial retina chip can also quickly calculate horizontal and vertical derivatives, which indicate local orientation. Histograms of local orientation can be used for fast pattern recognition [11, 6]. We show that these algorithms can distinguish poses of the



**Figure 5:** Three image projections determine the image moments. The horizontal and vertical projections can be performed on the artificial retina detector itself, approximately doubling the throughput.

human figure, and present computation times for the AR module.

Others have developed algorithms to identify the pose of a figure [8, 12, 5], but those algorithms do not meet our speed and cost requirements. The orientation histogram algorithm has limitations: the arms should not cross the body, and we have not studied recognition choosing from among a very large set of possible poses. However, the algorithm operates at the rate and cost level necessary for computer games.

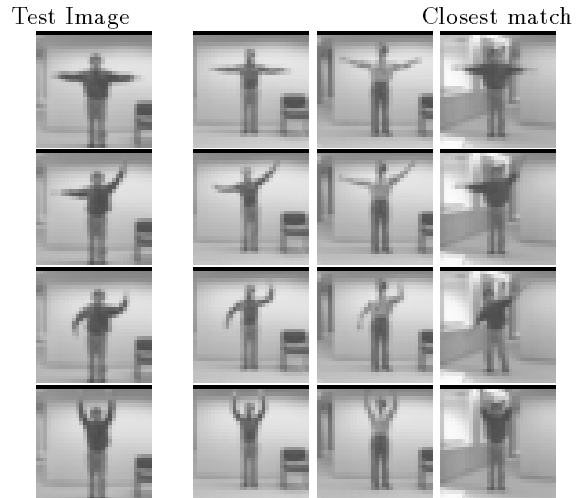
To use orientation histograms for recognition, we first calculate the spatial derivatives of the image,  $g_x$  and  $g_y$ . At positions where the image contrast,  $g_x^2 + g_y^2$ , is above a pre-set threshold, we compute the orientation from  $\theta = \arctan(g_x, g_y)$ . (This can be pre-computed for all pairs  $(g_x, g_y)$  with a look-up-table accessed by the micro-processor). We then divide orientation into bins (e.g.  $10^\circ$  per bin) and calculate the orientation histogram over some region. Measuring the Euclidean distance [6], or mutual information [11], between the orientation histograms of different images provides a measure of similarity which is contrast and position independent.

A global orientation histogram of a figure would average too much spatial information to infer pose. We divide the image of the figure into two or four sub-images, compute the orientation histogram independently for each one, and stack the resulting two or four histograms into one large feature vector. For the images shown, we only used the orientation histograms corresponding to the upper two quadrants.

We calculate a feature vector for each image, and compare the distance from the feature vector to a set of training images. These distances can be used to select the template corresponding to the nearest

pose, or to interpolate arm or leg positions between those of the templates.

Figure 6 illustrates that this method can work even for the 32x32 pixel images. Four test images are shown along with the closest matching pose from several different data sets of poses. The algorithm correctly picked out the closest pose in each case. Figure 7 shows the small test set of possible poses from which the algorithm chose.



**Figure 6:** Test images, and closest match to each from a set of 10 poses (Figure 7). The algorithm correctly picked out the most similar pose from each set to that of each test image.

Figure 8 shows a hypothetical game application of this coarse body position measurement—a jet flying game. The tilt of the plane in the game reflects the positions of the user's arms; some arm configurations can correspond to game commands, such as eject.

### 3.2.1 Fast calculation of orientation histograms

The AR module can speed-up the calculation of the orientation histograms. Figure 9 shows the processing steps: image derivatives are calculated on the detector; background subtraction and orientation histograms are calculated at the microprocessor. For this algorithm, and 32x32 resolution, the expected timing is roughly the same as before: 20 msec if all calculations are performed on the micro-processor, and 10 msec if the x and y derivatives are calculated on the AR chip and the rest calculated on the the microprocessor.

## 4 Summary

User interface for computer games represents a low-end niche in the spectrum of computer vision algorithms. The speed and cost constraints are severe, necessitating simple algorithms. Yet the interactivity and structure of the games allow for a rich response even with those simple algorithms.

We have developed an artificial retina module, which combines a detector with on-chip image processing with a microprocessor. We devised two algorithms tuned to the AR module, suitable for com-

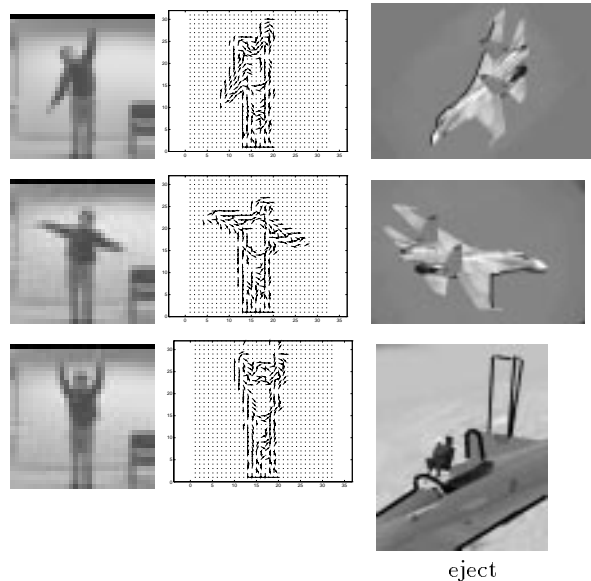


**Figure 7:** Test images for Fig. 6. Three people made ten poses, with four poses in common. Backgrounds were removed by subtraction. That can leave a ghost of the background inside the figure, but the effects of such residuals were negligible in the recognition performance.

puter game applications. One is based on image moments, and the other on orientation histograms. These algorithms respond to the user's hand or body positions, within 10 msec, with hardware that will cost several tens of dollars. We are developing computer games with these algorithms and this hardware.

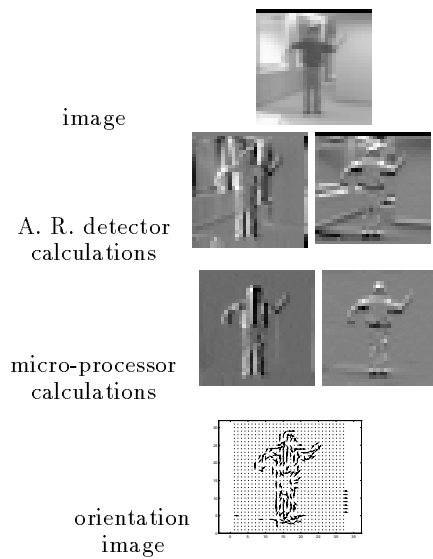
## References

- [1] D. H. Ballard and C. M. Brown, editors. *Computer Vision*. Prentice Hall, 1982.
- [2] D. Beymer and T. Poggio. Face recognition from one example view. In *Proc. 5th Intl. Conf. on Computer Vision*, pages 500–507. IEEE, 1995.
- [3] M. Bichsel. *International Workshop on Automatic Face- and Gesture- Recognition*. IEEE Computer Society, 1995.
- [4] M. J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. 5th Intl. Conf. on Computer Vision*, pages 374–381. IEEE, 1995.
- [5] J. S. E. Hunter and R. Jain. Posture estimation in reduced-model gesture input systems. In M. Bichsel, editor, *Intl. Workshop on automatic face- and gesture-recognition*, pages 290–295, Zurich, Switzerland, 1995. Dept. of Computer Science, University of Zurich, CH-8057.



**Figure 8:** Left: sample input images for flying game. Middle: orientation images, from which orientation histograms are calculated. Right: corresponding game action.

- [6] 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.
- [7] E. Funatsu, Y. Nitta, M. Miyake, T. Toyoda, K. Hara, H. Yagi, J. Ohta, and K. Kyuma. *SPIE*, 2597(283), 1995.
- [8] D. M. Gavrilu and L. S. Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In M. Bichsel, editor, *Intl. Workshop on automatic face- and gesture-recognition*, pages 272–277, Zurich, Switzerland, 1995. Dept. of Computer Science, University of Zurich, CH-8057.
- [9] B. K. P. Horn. *Robot vision*. MIT Press, 1986.
- [10] K. Kyuma, E. Lange, J. Ohta, A. Hermanns, B. Banish, and M. Oita. *Nature*, 372(197), 1994.
- [11] R. K. McConnell. Method of and apparatus for pattern recognition. U. S. Patent No. 4,567,610, Jan. 1986.
- [12] A. P. Pentland. Smart rooms. *Scientific American*, 274(4):68–76, 1996.



**Figure 9:** Processing steps in the orientation histogram algorithm. Top to bottom: Original image; x and y derivatives, calculated on the artificial retina detector; derivatives with the background image subtracted; orientation angle calculated for each pixel above a contrast threshold. Histograms of the orientations of the upper two 16x16 blocks of the orientation image were concatenated into one feature vector, which was compared with existing prototypes for recognition.