Artificial retina chips as on-chip image processors and gesture-oriented interfaces

Hiroshi Kage
Mitsubishi Electric Corporation
System LSI Division
Business Promotion Project of Artificial Retinas
Miyashimo 1-1-57
Sagamihara, 229-1195 Japan
E-mail: kage@ar.sow.melco.co.jp

Abstract. Players of a video game may sometimes find the use of conventional interfaces inappropriate. In such cases, we think that interfaces realized with a vision-based gesture recognition system may find favor. The artificial retina (AR) chip is a versatile image sensor whose use ranges from normal image acquisition to on-chip image processing, including on-chip image convolution. In this paper, we describe a gesture-input video game system, with the AR module including the AR chip, and motion-based gesture recognition algorithms. We showed that the algorithms can be accelerated by projection data, the direct output from the AR chip. To show its performance, we have applied our system to two commercially available video games. © 1999 Society of Photo-Optical Instrumentation Engineers. [S0091-3286(99)02212-6]

Subject terms: artificial retina chip; on-chip image processing; gesture recognition; motion analysis; optical flow; image sensor.

William T. Freeman
Mitsubishi Electric Research Laboratory
ITA
201 Broadway
Cambridge, Massachusetts 02139

Yasunari Miyake
Mitsubishi Electric Corporation
Advanced Technology R&D Center
Tsukaguchi-Honmachi 8-1-1
Amagasaki, 661 Japan

Eiichi Funatsu
Ken-ichi Tanaka
Kazuo Kyuma
Mitsubishi Electric Corporation
System LSI Division
Business Promotion Project of Artificial Retinas
Miyashimo 1-1-57
Sagamihara, 229-1195 Japan

1 Introduction

When we play commercially available video games, we frequently use conventional interfaces such as joysticks or buttons. Such interfaces are useful, but in some situations the use of our own gestures is more effective and leads to more satisfying gameplay. Unfortunately, most people cannot choose but use attached interfaces to video game machines. We believe that image recognition may be the key to realizing a gesture-oriented interface.

In spite of many researches on image recognition so far, there hardly exist satisfactory gesture-oriented interfaces. To realize it requires three items in general: image sensors such as CCDs, dedicated image-processing hardware such as DSPs, and gesture-sensing algorithms. The DSPs will be necessary in most cases because the CCD can only do image acquisition, so that subsequent image processing must be performed by a computer. Moreover, unless the gesture-sensing algorithms are tailored to the dedicated hardware, additional hardware will be necessary to speed up the processing. This will make the whole system large and its cost high.

The artificial retina (AR) chip1,2 senses the image and can also do on-chip image processing such as edge detection, image projection, and random access (i.e., arbitrary partial image acquisition). We have already developed two kinds of AR chips, each of which performs 1-D spatial filtering3,4 and 2-D filtering respectively. A silicon-based AR chip developed recently at Mitsubishi Electric can acquire images at a rate of more than 100 frames per second.3 We believe that this chip can overcome the above-mentioned problems pertaining to image processing, thereby downsizing the whole system and providing a high-speed and low-cost gesture interface.

With an AR module including the AR chip5 and novel gesture recognition algorithms, we have developed a gesture-oriented video game system. The algorithms we developed are based not only on normal image acquisition, but also on image projection. The AR chip can perform image projection by on-chip image processing, which allows the system to accelerate the algorithms. This is why we used the AR chip instead of a CCD as an image sensor.

The remainder of this paper is organized as follows. In Sec. 2, we overview the structural and operational principles of the AR chip and describe its use for on-chip image feature extraction. We also describe the use of the AR module as a versatile image-processing unit for multimedia use.
Section 3 deals with gesture recognition algorithms. In Sec. 4, we introduce a video game system using the AR module as a gesture interface, and describe two applications. Concluding remarks are made in Sec. 5.

2 AR Chip

2.1 Structural and Operational Principle of 1-D Filtering Performance

The AR chip is a versatile image sensor in applications ranging from normal image acquisition to on-chip image processing.\(^1\)\(^2\) Figure 1(a) shows a schematic structure of the AR chip with 1-D filtering. There are three parts: a 2-D array of variable-sensitivity photodetector (VSPD) cells, a random access scanner to control all the VSPDs’ sensitivity, and a multiplexer for image output. Figure 1(b) is a functional block diagram with a circuit diagram of each VSPD.\(^4\)

The scanner has three control electrodes: \(V_p\) (for positive output), \(V_n\) (for negative output), and \(V_r\) (for pixel reset). Each VSPD cell consists of a \(p\)n photodiode and a differential amplifier to achieve both high sensitivity and programmable polarity (positive or negative). This structure provides nondestructive image readout.\(^4\) The VSPDs aligned in the same column share the same control electrodes, and the VSPDs in the same row are connected to one current-summing line for output. By the combination of \(V_p\) and \(V_n\) electrodes, the VSPDs of the same column have the same polarity. However, some columns can be set to positive and others to negative. Each current-summing line produces the summation of the output currents from the VSPDs aligning in the same row. The final image-processing output is obtained by the multiplexer sequencing the columnwise results.

The on-chip image processing mentioned above can be regarded as vector-matrix multiplication. Let the matrix \(\mathbf{W}\) denote an input image projected onto the chip, and the vector \(\mathbf{s}\) a control pattern of the VSPDs’ sensitivity specified by the control electrodes at each column. The vector \(\mathbf{s}\) can specify the kind of image processing, and each element of \(\mathbf{s}\) can be set to a value 1, 0, or \(-1\). An output electrode is connected to each row, yielding an output photocurrent that is the vector product \(\mathbf{j} = \mathbf{W} \cdot \mathbf{s}\). By selecting each element of the vector \(\mathbf{s}\), the AR chip can execute linear operations such as edge detection or vertical image projection. This is an implementation of 1-D image filtering.

The chip design, testing, and optimization by CMOS fabrication are described in Ref. 4; one VSPD comprising six transistors is sized as 35\(\times\)26 \(\mu\)m\(^2\) (width\(\times\)height) with 25% fill factor, and the photosensitivity is 0.8 \(\mu\)A/\(l\)x for a 1-ms exposure time; the chip includes 256\(\times\)256 pixels in an 8.96\(\times\)6.66 mm\(^2\) image area, and the total chip size is 11.6\(\times\)11.6 mm\(^2\).

2.2 On-Chip Image Convolution as 2-D Filtering

In general, image convolution for feature extraction can be performed by a calculation involving a certain matrix (hereafter called the kernel) and its corresponding local image of the same size. For example, the Sobel operator, Kirsch operator, and Laplacian operator\(^6\) are commonly used as kernels for edge extraction. These operators can be represented as square matrices.

We have developed an AR chip that can realize on-chip 2-D image convolution. Figure 2 describes the schematic configuration of the chip, composed of a VSPD array (image area 128\(\times\)128 pixels) and four distinct scanners to realize on-chip convolution as follows:

1. \(K\) (for kernel) scanner: The \(K\) scanner holds the value of each element of a kernel, represented as a 5\(\times\)5 matrix. Every element can take the value 1, 0, or \(-1\) arbitrarily, and the origin of the kernel is at the top row and the leftmost column. The image area on the AR chip has an additional pixel area with four-pixel width and four-pixel height, padded to the full 128\(\times\)128 pixels, to obtain the image-processing result by applying a specific kernel.
2. \textit{R (for row select) scanner}: The \( R \) scanner specifies five consecutive rows to select the vertical position on the image area where the kernel should be applied.

3. \textit{X (for multiplexer) scanner}: The \( X \) scanner specifies five consecutive columns to select the horizontal position on the image area where the kernel should be applied. With the \( R \) scanner shifting from the top to the bottom and the \( X \) scanner shifting from the left to the right, any area of \( 5 \times 5 \) pixels in the VSPD array is scanned with overlapping. In a specific area of \( 5 \times 5 \) pixels, the kernel represented in the \( K \) scanner is applied, and the \( X \) scanner produces five columnwise outputs as currents. Each output current reflects the summation of products between elements of two respective columns of the above-specified area and the \( K \) scanner. The multiplication is actually an addition or subtraction, because each element of the \( K \) scanner is represented as 1, 0, or -1. Thus, the summed current equivalent to the total output amounts to the result of image convolution at the top and leftmost pixel in the above specified \( 5 \times 5 \) area.

4. \textit{C (for charge) scanner}: The \( C \) scanner charges all VSPDs by scanning the whole image area and controls the amount of photocarriers accumulated at each VSPD. This scanner is not directly involved in calculating the image convolution, but it is necessary for adjusting the pixel intensity.

Figure 3 shows the results of on-chip image convolution by three different kernels. Figure 3(a) shows the normal image acquisition of the AR chip. Figure 3(b) and Fig. 3(c) exemplify the cases of vertical and slanted edges, respectively. The kernel specified in Fig. 3(d) is an approximate Laplacian operator, which can detect the difference of pixel values in every direction at any pixel. The result of the on-chip image processing in Fig. 3(d) shows that the contour of a box is enhanced, whereas vertical edges are enhanced in the result of Fig. 3(b) and slanted edges in Fig. 3(c).

2.3 \textit{AR Module}

We have developed an AR module,\(^5\) a compact image recognition unit of size \( 8 \times 4 \times 3 \) cm. Figure 4(a) is a photograph of the module, including the AR chip of \( 32 \times 32 \) pixels with 1-D filtering and a 16-bit microprocessor with a clock speed of 10 MHz. The functional diagram inside the module is shown in Fig. 4(b). The AR chip performs several image-processing functions such as image acquisition, edge extraction, and 1-D projection. The microprocessor has two roles: providing the AR chip with the

---

![Diagram](image)

**Fig. 2** Schematic structure of on-chip image convolution of the AR chip.

![Diagram](image)

**Fig. 3** Examples of on-chip image convolution of the AR chip (128×128 pixels). (a) Normal image acquisition. Top: specified kernels corresponding to (b) enhanced vertical contrast, (c) enhanced slanting contrast, and (c) an approximate Laplacian. Bottom: corresponding results of on-chip image convolution.
sensitivity vector \( s \), as described in Sec. 2.1, and processing the AR chip output to analyze the gestures. The processing speed of the module is about 10 ms per frame. This speed is limited by the built-in analog-to-digital converter (ADC) of the microprocessor, but can be reduced to 1 ms if a faster ADC is used instead of the built-in one.

In Ref. 5, we described the use of the AR module to recognize hand gestures; the size, the center-of-mass position, and the orientation of a hand are recognized by the module, and these results were used to control the direction of a toy robot. Thus the module can be regarded as a self-contained gesture-input interface. In a similar manner, the AR module can be used to recognize body action. Examples of the AR module applied to video games are described in Sec. 4.

3 Gesture Recognition

3.1 Background

Recently video games have been very popular and various game contents have been available for game users. In spite of the progress of commercially available video game machines in terms of computational power, available game interfaces such as joysticks or keypads have remained the same, except that specific game interfaces such as a toy steering wheel have been attached to certain games. We believe that a gesture-input interface can make video games more satisfying.

For example, let us consider a track-and-field game, as described in Sec. 4.3. In such a game, the players usually press buttons repeatedly to make a game character run or jump. It may make the game more exciting if a gesture-oriented interface can be provided.

To realize such an interface, we decided on hand motion and body motion as inputs for gesture recognition. We developed two algorithms: the center-of-mass algorithm to detect the position of a moving hand, and the optical-flow algorithm to detect the direction and evaluate the amplitude of body motion. These algorithms were used in two different video games described in Sec. 4.3. We have based these algorithms on normal image acquisition, but both of them can be accelerated by using the outputs of the AR chip with on-chip image projection.

To build up gesture recognition algorithms mentioned above, we were based on motion analysis. Here we clarify the advantages of motion analysis over shape analysis as follows:

1. **Hand motion**: The position of a moving hand can be easily recognized by motion analysis. Provided that there is no motion except the hand motion within the scope of the image, its center-of-mass position can be calculated by frame subtraction over time. In contrast, when trying to identify the position of the hand by shape analysis, exhaustive template matching over the entire image must be done. In addition, it becomes much more difficult when any similar object exists within the visual scope even if it is stationary.

2. **Body motion**: The result of optical flow calculation consists of pixelwise motion direction and amplitude. When a game user jumps in front of an image sensor, for example, the resulting optical flow pattern reflecting the user’s jump can be obtained. By taking a vertical summation of all vectors and monitoring its amplitude over time with a threshold, the timing of the jump can be easily detected. It would be much more difficult to determine the timing of jumps by shape analysis than by motion analysis. This is because it is not easy to determine which body part represents the whole body motion even if each position of body parts is successfully identified.

3.2 Center-of-Mass Algorithm

We built a center-of-mass algorithm to recognize the position of a moving hand. It is based on ordinary image acquisition, but can be accelerated by taking advantage of on-chip image projection in the AR chip.

The algorithm comprises four steps:

- **Step 1. Frame subtraction**: In this step, the absolute value of the pixelwise difference is calculated between successive images over time.
- **Step 2. Frame projection**: Using the result of step 1, frame projection along horizontal and vertical directions is obtained. Frame projection means horizontal or vertical summation of the pixel values obtained by frame subtraction.
- **Step 3. Center-of-mass calculation**: The center-of-mass coordinate of pixels representing motion can be calculated by using the result of step 2. The horizontal component can be calculated by using the horizontal frame projection, and the vertical one by using the vertical frame projection. The center-of-mass calculation using frame projection is described in Ref. 7.
- **Step 4. Construction of a motion vector**: A motion vector, indicating the position of a moving hand, can
be constructed by defining the center of the image as a start point and the center-of-mass position calculated by step 3 as an end point.

Figure 5 illustrates the algorithm. Figure 5(a) is an image taken by the AR chip, including the hand motion. Figure 5(b) shows an example of the result of step 1, where gray pixels represent hand motion. In Fig. 5(c), the results of steps 2 and 3 are superimposed on the result of frame subtraction. Black bars placed at the right and the bottom are the results of horizontal and vertical frame projection, respectively. To eliminate other motion than that of the hand, we set a threshold on the horizontal and on the vertical frame projection. The horizontal and vertical solid lines indicate the result of the center-of-mass calculation with thresholding. The crossed point of the dashed lines indicates the actual position of the center of mass of the pixels in motion over the entire visual scope, and that of the solid lines yields a stronger signal about the hand position than the actual center-of-mass position. With this thresholding, we can construct a motion vector as shown in Fig. 5(d), reflecting the approximate center of the moving hand as a result of step 4.

Here we describe how the above algorithm can be accelerated by on-chip image projection of the AR chip. In the case of ordinary image acquisition by conventional image sensors such as the CCD, steps 1 and 2 are needed, whereas neither is necessary when using image projection in the AR chip. If the image acquisition is done with a resolution of 32×32 pixels, the total calculation by conventional image sensors is estimated as 3264 operations (including arithmetic additions, subtractions, and multiplications), while the AR chip needs only 256 operations. The acceleration by the AR chip amounts to 12.75 times. We have already developed an AR chip with 256×256 pixels, and this chip will yield an acceleration of 96.75 over conventional image sensors of the same resolution.

3.3 Optical-Flow Algorithm

We built a fast and simple optical flow algorithm to recognize body motion. Most of other optical flow algorithms are iterative, and they cannot perform fast calculation except with dedicated hardware. Our algorithm is noniterative, approximating the gradient and integration steps of a more precise calculation. It was suggested by considering a simple case of an image patch moving by one pixel in some direction over two frames. Then we discovered a fundamental rule related to pixelwise estimation of motion direction described below, which became the underpinning of our four-step algorithm.

In the first step, it is determined whether or not a motion vector can be calculated at each pixel. This is done simply by evaluating the change in each pixel value between two successive frames; if a pixel value has no change, the calculation of its motion vector is omitted.

In the second step, motion direction is estimated by evaluating the sign of each pixel-value difference. We use the classification table for moving contrast edges shown in Fig. 6. This table illustrates two cases of a 1-D edge with dark and light luminance moving along a specific direction.
By comparing the patterns in Figs. 7(d) and 7(e), it is observed that the latter flow pattern is much closer to the supposed motion in Fig. 7(a). In particular, some vectors of Fig. 7(e), such as the ones at the upper right and lower left corners in the previous frame, have more plausible directions than in Fig. 7(d).

We applied the above algorithm to two successive images of 8×8 pixels obtained by local averaging (within non-overlapping areas of 4×4 pixels) and subsampling from the original 32×32-pixel data to capture the horizontal or vertical body motion [see Fig. 10(a) in Sec. 4.2 for the AR image (32×32 pixels), and Fig. 10(b) for the resulting optical flow pattern]. Such local averaging before applying the optical-flow algorithm to the images is necessary to capture the global direction of body motion. If no local averaging is done, the resulting optical flow includes a lot of local motion, and the summation of the whole motion vectors will not necessarily reflect the direction of body motion.

We assumed in the above explanation that the algorithm is based on normal image acquisition. This means that the algorithm can also be applied when conventional image sensors such as CCDs are used. But in detecting the average horizontal or vertical motion over the image, we can obtain an acceleration by on-chip image projection on the AR chip.

In the case of ordinary image acquisition, the resulting optical flow is obtained as a two-dimensional pattern. The horizontal and vertical components of the whole motion amplitude can be extracted by taking the sum of the whole motion vectors. But both components can be calculated via image projection by applying the optical-flow algorithm to the horizontal and vertical image projections. The output of on-chip horizontal projection of the AR chip can be regarded as an image composed of 32×1 pixels, and a one-dimensional optical-flow pattern can be obtained by applying the algorithm. In this case, the resulting optical flow pattern reflects horizontal motion, and the sum of the constituent motion vectors amounts to the horizontal compo-
component of the whole motion amplitude. The vertical component can be calculated in the same way from the on-chip vertical projection.

Here we estimate how the algorithm can be accelerated by on-chip image projection on the AR chip. We have three choices for extracting the horizontal and vertical directions of body motion, as follows:

1. Based on normal image acquisition: The procedure has four steps: the averaging and subsampling over two successive frames, frame subtraction, optical-flow calculation, and 2-D vector summation.

2. Based on image projection to be calculated on a separate computer (not on chip): The procedure has five steps: the calculation of the horizontal and the vertical projection over two successive frames, subtraction of those projections, averaging and subsampling of the results, 1-D optical-flow calculation, and 1-D vector summation.

3. Based on on-chip image processing with the AR chip: The procedure has four steps: subtraction of horizontal and vertical projections between two frames, averaging and subsampling of the results, 1-D optical-flow calculation, and 1-D vector summation.

In case 1, the total amount of calculation is estimated as 3536 arithmetic operations including addition, subtraction, division, and comparison. The estimates in cases 2 and 3 are 4358 and 262 calculations, respectively. Both cases 1 and 2 are feasible with any conventional image sensor, and the former procedure is faster in view of the above estimation. Comparing case 1 implemented by a conventional image sensor with 32×32 pixels and case 3 implemented by the AR chip with on-chip image projection, the acceleration amounts to 13.5 times. But it will be much higher if a CCD with high resolution is used, because the averaging then takes much more calculation.

4 Game Applications

4.1 Game System

We have developed a game system using the AR module serving as a gesture-oriented interface described in Sec. 2.3. As shown in Fig. 8, the system has three components: the AR module, a PC, and a video game machine. Two displays are connected to the PC and the game machine. No additional hardware was attached to the PC for accelerating the algorithm. We used a commercially available video game machine, the SEGA Saturn, which is one of the major video game machines for home use. A conventional game controller, usually attached to the SEGA Saturn, was replaced by the AR module connected to the PC. This means that a signal released from the PC is equivalent to that released by pressing a button on the conventional game controller. The PC display is used to monitor the current image taken by the AR chip and observe the results of gesture-sensing algorithms.

While playing a game, a player stands in front of the AR module, watching the game display. Figure 9 shows an example view of a gameplay.

4.2 Game Signal

To replace a game controller by our gesture-oriented interface, we had to construct a signal equivalent to the data the game controller sends to the game machine. We used the two algorithms for gesture analysis explained in Sec. 3: the center-of-mass and the optical-flow algorithm. In the former algorithm, it is simple to create an equivalent game signal because the output of the center-of-mass method is a vector and can be identified with the signal emitted from the directional controller on the game pad. But in the latter algorithm, it needs some reduction to create an equivalent signal from an optical flow pattern, because of the richness of its motion information. Figure 10 explains how to reduce the optical flow pattern to simple commands such as jumping or running.

Figure 10(a) is an image taken by the AR chip with a resolution of 32×32 pixels. When the player jumps with both arms spreading, the optical flow pattern composed of 36 vectors was derived as shown in Fig. 10(b). It was calculated from two successive images at 8×8-pixel resolution, obtained by averaging and subsampling from the 32×32 pixel data. There are no vectors around the marginal area of the optical flow pattern where there are no adjacent pixels and a motion vector cannot be computed.

One of the simplest ways to reduce the optical flow pattern to a game signal is to take an average of the constituent motion vectors. The vertical component of the averaged vector is in proportion to the amplitude of the player’s jump. The zigzag trace in Fig. 10(c) reflects vertical ampli-
tude of the player’s frequent jumps. If a threshold is set at some level and the motion amplitude goes over the threshold at a certain time, the command for jumping will be signaled to the game machine. In addition to the vertical motion amplitude, the horizontal component of the averaged vector can also be used for a game signal. When the player runs, for example, the resulting optical flow pattern will reflect the motion of the arms. In this case, the resulting pattern will produce a zigzag trace similar to Fig. 10(c). When the player stands still, the trace will become a vertical line (hereafter called a neutral line), and we can create a game command for running by reading out a time for the trace going across the neutral line from the left side to the right side or the other way round. As described in Sec. 4.3, the game signal for running in Decathlete can be generated by repeated pressing of a particular button on a game pad. By using the above method of optical flow analysis, we can replace the tedious button pressing by our own gestures.

4.3 Game Examples

4.3.1 NiGHTS

NiGHTS is a so-called flight-action game, where a game character flies through a pseudo-3-D space. The direction of the game character can be controlled by a directional control button on a game pad connected to the SEGA Saturn. To gain a high score, the direction of flight of the character must be elaborately controlled to capture various objects floating in the pseudo-3-D space. Some objects can be captured only when surrounded by a loop orbit of the character. A game user can proceed to the higher stage only after all objects are captured. The basic game operation is a skillful use of the directional control button operated by the game user’s thumb. This button can be operated not only in every 2-D direction, but also along the loop orbit.

Using the center-of-mass algorithm described in Sec. 3.2, we replaced the above game operation of the directional button by a motion of the player’s hand. If the player moves the hand within the visual scope of the AR chip, the position of the hand is converted to a vector directed from the center of the AR image to the position corresponding to the hand [Fig. 11(a)]. The game control for any direction can be generated by pointing the hand in that direction. To make the game character fly along a loop orbit, the player has only to shift the position of the hand along the loop orbit in the space. The resulting sequence of the vectors generated by the algorithm also makes a loop. Thus the player can make the game character fly in any desired direction [Fig. 11(b)].

4.3.2 Decathlete

Decathlete is a track-and-field game composed of ten events: 110-m hurdles, 100-m dash, 400-m race, long jump, discus throw, javelin throw, shot put, 1500-m race, high jump, and pole vault. We omitted three of these events: the 1500-m race, high jump, and pole vault. The reason for omitting the 1500-m race was simply the length of time it took. The remaining two events were omitted because it seemed difficult to select appropriate gestures for the player corresponding to the feats of a game character in the air.

The basic operation for the game user is the use of two buttons, referred to as a running button and a timing button. The running button is used to speed up the game character’s running pace by repeated presses. The timing button works for each event as follows:

1. **110-m hurdles**: The button press determines when to make the game character jump.
2. **Javelin throw**: The button press and release give the first and second timings for hurling a javelin, respec-

Fig. 10 The reduction of the optical flow pattern to a game signal. (a) An AR image including a player jumping with both arms spreading. (b) The resulting optical flow. (c) A plot of horizontal motion amplitude against time.

Fig. 11 (a) An image taken by the AR chip, including a hand motion and the resulting game command as a vector (white arrow). (b) A scene from NiGHTS: a game character and its flying direction [black arrow, equivalent to the white arrow in (a)].
Motion

3.3, we replaced the button operation by the player's body motion. The optical-flow algorithm extracts the vertical and horizontal amplitudes of the body motion. The operation of the timing button is replaced by monitoring the horizontal amplitude over time; the operation of the running button is replaced by monitoring the vertical amplitude against time is a zigzag line. As soon as the player swings both arms sideways, the plot of the horizontal amplitude is monitored to determine the running status. If the player swings both arms sideways, the plot of the horizontal amplitude against time is a zigzag line. As described in Sec. 4.2, the crossing of the trace over the neutral line is converted to the game command equivalent to the press and release of the running button.

In the javelin throw, only monitoring the vertical amplitude is necessary to obtain the time to hurl a javelin. The two timings for when to hurl a javelin and how far the javelin flies are determined by the upper and lower thresholds; if the vertical amplitude goes under the lower threshold, the first timing is triggered, and the second timing follows when the horizontal amplitude goes over the upper threshold. Actually, we can trigger these timings by upward and downward body motion successively. Thus the player pantomiming the hurl of the javelin can generate the desired game signals.

The merits of the gesture-oriented interface for any game user are summarized as follows:

1. With a conventional game pad, the running speed can be controlled only by pressing the running button. To attain a high score in a track event like the 100-m dash, the game user has no choice but to press the button fast. This will become tedious. In contrast, if our gesture-oriented interface is available, the game user can control the speed of the game character by sideways motion of the arms; this will provide the game user with a feeling of actual running in the track event.

2. The operation of the timing button is less tiring than that of the running button, but the game player may feel a lack of resemblance to the desired result. The downward and upward body motion, replacing the operation of the timing button, is more suggestive of the game's content. Although the player's gestures will vary among the events of javelin throw, long jump, shot put, and discus throw, they will be effective as long as they are reduced to downward or upward motion at an appropriate time. The player does not have to have any prior knowledge about how to play each event; only the downward or upward motion will be necessary.

4.4 Discussion

4.4.1 Limitations of the system as a vision-oriented gesture interface

In this sub-subsection, we discuss some limitations of our game system from an algorithmic viewpoint. Here we refer to two examples, described in Sec. 4.3: NiGHTS and Decathlete. A fundamental limitation in both systems is that the player must have a minimum contrast against the background for motion detection.

The NiGHTS system has the following additional limitations:

- The algorithm only detects the position of a single hand in motion, and cannot exclude other motion in the background. Therefore motion generated by both hands simultaneously cannot work properly to play the game, and two players cannot play at the same time.
- To generate the same vectors over successive frames, the player must keep on moving a single hand at a fixed position.
- The player must keep the hand within the visual field of the AR chip.
- The algorithm cannot discriminate hand motion from a change of luminance in the background (e.g., due to blinking lights or TV monitors).

The Decathlete system has similar limitations, but the player does not have to be careful about where to stand or
how much to jump. As long as the player moves within the visual field of the AR chip and jumps at the right times, the resulting gesture will be effective.

Although our game system has several limitations, it does not need any attached markers to obtain information on body motion. This merit enables us to realize a gesture-oriented interface on a small scale and at low cost.

4.4.2 Comparison of the performance using the standard game pad and the gesture interface

In the NiGHTS system, we replaced the operation of the directional control button on the game pad by hand motion. We think that such a gesture interface will make any game user feel much closer to the game content than using the conventional game pad, although the performance will be nearly the same, especially for first-time users. In the Decathlete system, we think that our gesture interface improves on the standard game pad in two ways. One is that tediously frequent button pressing is clearly inferior to sideways arm motion as a way to make the game character run. The other is that our gesture interface allows a game player to enjoy a much closer analog of the game content of Decathlete than with the standard game pad, which calls only for the simple operation of button pressing.

4.4.3 Originality of our game system

Regarding the originality of our work as a video game-oriented gesture interface, we compared our system with another gesture-oriented game pad. We discussed several limitations of our system, and compared our system to two SEGA Saturn games, NiGHTS and Decathlete, and described how these two algorithms extracted signals equivalent to those emitted from the conventional game pad. We discussed several limitations of our system, and compared our system with another gesture-oriented game system to show our originality.

In the future our research will be directed to more elaborate gesture recognition involving the identification of motions of each part of the body.

Acknowledgment

We thank SEGA Enterprises for their helpful cooperation.

References


Biographies of the authors not available.