A Novel FPGA-based Hand Gesture Recognition System

Chao-Tang Li, Wen-Hui Chen

Abstract

There are many applications using hand gesture as a nature control interface, such as human-machine interaction and interactive entertainment. With the advancement of computer vision and machine learning, the development of vision-based hand gesture recognition systems has received more and more attention in recent years. However, much previous research was focused on software algorithms running on a PC-based platform, which makes it inapplicable to real-time applications. In this study, a hardware implemented gesture recognition system based on improved optical flow was proposed to accelerate the execution and achieve real-time operation. Experimental results showed that the proposed framework is feasible and can reach the speed of 30 frames per second.

Keywords: FPGA, Gesture Recognition, Image Processing, Optical Flow

1. Introduction

In recent years, a number of vision-based applications have been proposed for gesture recognition in [1]-[5][11]-[16]. The gesture information can provide a rich signal or message for home entertainment. Several studies developed various methods or solutions to meet the requirements of various applications. In [2], the authors built a powerful man-machine interface by using Laplacian of Gaussian (LOG) operators, motion energy, and finite state machines for feature extraction. However, a disadvantage in this system is that the motion energy requires large computation. In [3][4], a gesture recognition system was proposed for Human-Computer Interaction (HCI). The AdaBoost and Support Vector Machine (SVM) were used to detect hand location and to recognize hand type, respectively. However, a disadvantage in this system is that the pre-training process is required.

In [5], a remote control system by motion detection and open finger counting was built. To enhance the accuracy of object detection, the entire candidate region is assigned a threshold for selection of detected objects. In object detection [6]-[8], the color processing is based on skin color information. In recognition, the k-cosine in [9][10] was used; and two procedures of predefined and pre-training were also required. A disadvantage in the embedded system is that of mathematical operation, such as cosine calculation. The four modules used in [11][12] include real-time hand tracking and extraction, feature extraction and gesture recognition, which were integrated for gesture application. In general, the image difference operation in temporal domain, skin color detection, edge detection and region identification were used for feature extraction.

To enhance the accuracy, two fundamental concepts of region-of-interest and background subtraction were used in detection. Moreover, the Fourier operator was used in recognition. The accuracy of the final results was acceptable; however, a number of large stages are required in the embedded system. A pattern recognition model for dynamic hand gesture recognition was proposed in [13]. The neural network (NN) with a weighted fuzzy min-max type was used. A widely used concept is that a predefined pattern is a powerful engine for recognition when NN is applied. However, its disadvantages are pre-training and large computation in a real-time system.

Four articles, [14]-[17], proposed a suitable method to meet real-time embedded system requirement. Two concepts of Low complexity and FPGA-based architecture were considered. Based on the analysis, the embedded gesture system is a crucial trend for future application; therefore, this study proposes an FPGA-based gesture recognition system. To avoid the design of high complexity, the developed algorithm was improved for parallel processing.
2. Description of the proposed framework

The proposed system includes a CCD camera, display, and core DE2-70 evaluation board, as shown in Figure 1. From the CCD device, a real image was captured and processed by our proposed algorithm. The Set-Top Box was used as TV input. According to a predefined gesture type, the user can turn on the TV, turn off the TV, increase volume, decrease volume, switch to previous channel and switch to next channel.

![Figure 1. The proposed system hardware resources and related peripheral devices](image)

The detailed overall system description, in Figure 2, includes hardware and software co-design. First, the CCD camera captured the image and sent the composite signal to TV decoder device. Through the ITU-R 656 decoder, the YCbCr signal base on interlace mode was preformed. The preprocessing image flow was completed and the YCbCr signal was fit into FPGA core for algorithm operation. The median filter was first applied and the result was sent to the next algorithm of skin color detection before the entire candidates feature was selected. Optical flow provides accurate analysis and decision making to adjust the hand motion. The Kalman filter in feature extraction was used to reduce noise effect. From these processing techniques, the possible gesture was extracted concurrently by the FPGA system. After the object tracking, gesture adjustment and final command result were implemented in Nios II processor because the computation complexity was low and the calculation was dynamic. Finally, the command result was sent to the Set-Top Box for TV control.

![Figure 2. The function diagram of the proposed system](image)

2.1. Algorithms architecture

The skin color detection was applied to enhance the accuracy of gesture recognition. Moreover, to avoid the influence of the outside environment and reduce the computational complexity, the optical flow was used in post-processing of skin color detection before the hand location was addressed.
According to real image verification, the speed and accuracy in gesture recognition are fast and acceptable, respectively. All of the related algorithms are presented as follows:

2.1.1 Gesture definition

To achieve easy usage, six different gestures were defined, as shown in Figure 3, such as beckon gesture in (a), goodbye gesture in (b), move left in (c), move right in (b), move up in (e) and move down in (f). For detailed expressions, the beckon gesture is defined as corresponding with ‘enter,’ the goodbye gesture is defined as corresponding with ‘leave,’ and the remaining gestures are defined for general control commands. Unlike previous studies that required the characteristic of a finger as the indicated feature, the advantage is that an accurate command control can be implemented when users only move their hand. Thus, this property is useful and distinct to compare with past methods.

2.1.2 Median filter

A widely used technology for noise reduction in image preprocessing is the median filter. The characteristic of parallel processing is also suitable for hardware implementation and selects the 3×3 filter basis, which was recommended in past studies.

2.1.3 Skin color detection

The skin color detection was selected to efficiently detect hand location. The main idea is that the characteristic of skin color varies with a general background environment in Cb and Cr color domains. To verify this idea, in [6]-[8], a useful approach of skin color segmentation was based on YCbCr color model and sigma control limits for variations in its color components was proposed. The segmentation was more efficient in terms of speed and accuracy. The YCbCr color model was more precise than the RGB color model, and the relative experimental results are shown in [6]-[8]. In the current embedded system, the YCbCr color model was supported as standard interface of image data capture. Thus, we selected the YCbCr color model for skin color detection. According to empirical threshold and the equation in (1), the result of skin color detection was decided when $P(x,y) = 1$.

$$P(x,y) = \begin{cases} 1, & \text{if } (lcl_{cr} \leq Cr(x,y) \leq ucl_{cr} \text{ and } lcl_{cb} \leq Cb(x,y) \leq ucl_{cb}) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Where $Cr(x,y)$ and $Cb(x,y)$ indicates Cr and Cb color intensity; the definition of four variables of $lcl_{cr}, ucl_{cr}, lcl_{cb}$ and $ucl_{cb}$ are shown in (2)

$$\begin{align*}
lcl_{cr} &= \mu_{cr} - 1.5\sigma_{cr} \\
ucl_{cr} &= \mu_{cr} + 1.5\sigma_{cr} \\
lcl_{cb} &= \mu_{cb} - 1.5\sigma_{cb} \\
ucl_{cb} &= \mu_{cb} + 1.5\sigma_{cb}
\end{align*} \quad (2)$$

Two variables of $\mu_{cr}$ and $\mu_{cb}$ indicate the image mean value in the Cr and Cb color domain, respectively; two variables of $\sigma_{cr}$ and $\sigma_{cb}$ indicate the image standard variation in the Cr and Cb color domain, respectively.
2.1.4 Optical flow

Our proposed optical flow, which based on the Horn and Schunck algorithm, was modified and applied to enhance the hand detection result from sections 2.1.2 and 2.1.3. First, as shown in Figure 4, we defined basic unit in 18×18 pixels and the minimal unit in 3×3 pixels.

![Figure 4. Definition of basic unit for each motion vector in each frame](image)

For each minimal unit, the optical energy presentation in x-domain, y-domain and time-domain were calculated by $E_b^x$, $E_b^y$ and $E_b^t$ respectively. Both $E_b^x$ and $E_b^y$ were used for motion vector presentation and $E_b^t$ was used for density of current motion vector. For time-domain operation, using $k$ and $k+1$ as indexing indicates the current block and previous block in co-location, respectively. A detailed expression is shown in (3).

$$
E_b^t = \frac{1}{4} \left\{ I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} - I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1} \right\}
$$

$$
E_b^x = \frac{1}{4} \left\{ I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} - I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1} \right\}
$$

$$
E_b^y = \frac{1}{4} \left\{ I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} - I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1} \right\}
$$

Finally, we obtained each basic unit of optical energy presentation in $t^{th}$ frame according to (4). These results were applied after Kalman filter processing.

$$
E_x(t) = \frac{\sum_{i=1}^{36} E_x^i}{36}, \quad E_y(t) = \frac{\sum_{i=1}^{36} E_y^i}{36}, \quad E_t(t) = \frac{\sum_{i=1}^{36} E_t^i}{36}
$$

2.1.5 Kalman filter and gesture recognition

The gesture tracking based on Kalman filter was considered and implemented for accurate adjustment. The large degree of gradient value in all of the candidates of basic unit of optical flow was selected for Kalman filter. The gesture trajectory can be obtained for final recognition processing from the continuous images. Five statuses are used to define a state transition model, as shown in Figure 5. In four statuses, that is, R, U, L and D, only one of X-direction or Y-direction is trigged. S status is the stop command. In detailed state presentation, the U status indicates hand movement up from the middle of the chest location in X-direction and the Y-direction remains motionless; the D status indicates hand movement down from the middle of the chest location in X-direction and Y-direction remains motionless; the R status indicates hand movement right from the middle of the chest location in Y-direction and X-direction remains motionless; the L status indicates hand movement left from the middle of the chest location in Y-direction and X-direction remains motionless.
All six states are presented in Figure 5. The two states transition of beckon and goodbye gestures defines two arbitrary states transition. The main purpose is to reduce noise effect and to improve algorithm performance in the decision rule.

2.2. System software and hardware architecture

The evaluation board of ALTERA Cyclone II EP2C70 FPGA, version number DE2-70, was selected for system hardware. To implement real-time operating system (RTOS), the Nios II processor [19] was pre-built in this evaluation board. The advantages include large capacity and multi-type in memory operation. The Tick value was short because the large computation was conducted on the FPAG gesture system and the frame rate was approximately 30 frames-per-second (fps). Moreover, by providing a powerful FPAG-based system, the multi-task mechanism to control system I/O and algorithm results was suitable in this hardware/software co-designed system.

The proposed gesture recognition system can achieve 30 fps real-time 640×480 pixels image processing by utilizing only adequate or limited hardware resource. As a result, all data buses are designed by 8 bit for integer data calculation, which is shown in Figure 2.

A RTOS should be selected carefully for a multi-tasking system. According to experimental results in [18], the µC/OS II RTOS was considered and implemented in this study. Consequently, another crucial issue is that the widely developed resources can be obtained easily in internet and research articles. In this study, the µC/OS II RTOS processed three main tasks, including hand tracking, gesture recognition and peripheral devices. The overall flow of system software is shown in Figure 6. First, the initialization part constructed all necessary drivers, such as system clock, I/O and Timer. Three tasks for algorithm execution were built. Finally, parallel execution of three tasks ran on RTOS for gesture recognition. To avoid deadlock in task overhand, the time-sharing protocol and 10ms for tick assignment were implemented and assigned, respectively.

Figure 6. Nios II processor core integrated with peripherals and System software flow
### 2.3. Algorithm and hardware comparisons

In Table 1 show comparisons of three related works with the proposed design in both hardware and algorithm perspectives.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Optical flow gradient</td>
<td>2. Motion</td>
<td>2. Chain code</td>
<td>2. Region</td>
</tr>
<tr>
<td>Recognition method</td>
<td>State transition</td>
<td>State transition</td>
<td>Trajectory-based neural network</td>
<td>Hidden Markov Models</td>
</tr>
<tr>
<td>System platform</td>
<td>FPGA</td>
<td>PC</td>
<td>FPGA</td>
<td>FPGA</td>
</tr>
<tr>
<td>System chip</td>
<td>Altera Cyclone II EP2C70</td>
<td>Pentium PC (Core2 Duo, 2.40GHz)</td>
<td>Xilinx Virtex II Pro 2VP30</td>
<td>Xilinx Virtex II XC2V2000</td>
</tr>
<tr>
<td>Resolution</td>
<td>640×480 pixels</td>
<td>320×240 pixels</td>
<td>320×240 pixels</td>
<td>384×240 pixels</td>
</tr>
<tr>
<td>Process speed</td>
<td>30 fps</td>
<td>real-time</td>
<td>30 fps</td>
<td>none</td>
</tr>
<tr>
<td>Cost</td>
<td>Low</td>
<td>High</td>
<td>Middle</td>
<td>Middle</td>
</tr>
</tbody>
</table>

### 3. Experimental setup

A CCD camera is adopted with the Altera DE2-70 FPGA hardware platform in our implementation and the Altera’s Quartus II version 11.0 and Nios II EDS version 11.0 are utilized as development tools. This platform owns a video interface, two 16 bit 32MB SDRAM modules and large FPGA resources, which is suitable for video processing. In order to operate gesture recognition accurately, we suggest that a user should be 1.5~3.0 meters away from the CCD camera so that the size of a user's hand can be fitted reliably to operate recognition.

#### 3.1. Hardware resource

The hardware specification of the Altera DE2-70 FPGA platform as Figure 1 adopted in our design is as follows:
- Altera Cyclone® II 2C70 FPGA device with 68,416 logic elements and 250 M4K RAM blocks
- Altera Serial Configuration device - EPICS16
- 2-Mbyte SSRAM
- 8-Mbyte Flash memory
- 50-MHz oscillator and 28.63-MHz oscillator for clock sources
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- Two 40-pin Expansion Headers with diode protection

### 4. Experimental results analysis

To verify our proposed algorithm, a referenced performance of processing frame rate in INTEL 3.0G Hz PC was 14 fps. The developed environment was VC++ 2008 and openCV2.0 library was applied for initial algorithm construction. The image resolution was 640×480 pixels. To verify that our proposed system was workable, four real video sequences, as shown in Figure 8, and 640×480 pixels resolution were selected. For detailed analysis, six continuous images and three valuable results, including optical flow, detection and tracking are presented in Figure 9. In these results, the hand
location and gesture recognition are boxed. The smooth tracking is depicted from (a) to (f). The detailed resource layout and our implemented real FPGA-based gesture system are depicted in Figure 7.

In general, the processing frame rate ran on 30 fps under 50 MHz crystal oscillators. The logic elements were limited to 25% of total logic elements and the final result of the maximal frame rate was up to 75 fps under 125 MHz crystal oscillator. To verify the system performance, each case and function were tested 25 times. All experimental results are shown in Table 2. The data in this table indicate that the recognition rate for cases (a) and (b) in Figure 8 was 100%. The recognition rate for the beckon and goodbye gestures of case (c) was 96%; the recognition rate for the other four gestures of case (c) was 100%; and the recognition rates for the six gestures in case (d) were from 84% to 96%.

Because the algorithms that have been used can greatly decrease interference originating in the static background, the test results of the two indoor environments shown by cases (a) and (b) in Figure 8 can reach a recognition rate of 100%. However, the recognition rates of the beckon and goodbye gestures in case (c) have failed to reach 100%. This is primarily because the outdoor background was excessively bright and a large area of the buildings has been recognized as skin color. This has caused the contrast ratio between the hand in motion and the background to decrease, and made the features of the moving hand less clear. Thus, the tracking target has been lost, and recognition errors result. The reason that the recognition rate of case (d) is lacking is because some of the moving vehicle regions in the background have been judged as skin color, interfering with hand tracking. The interference of this horizontal movement has a greater influence on the gestures indicating goodbye, move left, and move right, which move horizontally in the same direction. This leads to tracking target errors or a loss of the tracking target, reducing the successful recognition rate.

As our proposed system is based on the combination of skin color detection and optical flow gradient features, the user should assure the palm part is not blocked, but the length of sleeves is not limited. The speed of gesture is assumed as normal speed, waving too fast or too slow would affect recognition result, and the lack of environment luminous would limit the system from recognition.

**Figure 7.** The hardware resource consumption and real FPGA-based gesture system

**Figure 8.** Four different cases for gesture verification, (a) simple indoor background, (b) complex and static indoor background, (c) complex and static outdoor background and (d) complex and dynamic outdoor background
Figure 9. An example of real gesture tracking from (a) to (f)

Table 2. The experimental results include detected counts and accurateness in difference case

<table>
<thead>
<tr>
<th>Hand Gesture</th>
<th>Detected counts</th>
<th>Accurateness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Case (a)</td>
<td>Case (b)</td>
</tr>
<tr>
<td>Beckon</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Goodbye</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Move left</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Move right</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Move up</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Move down</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

5. Conclusions

A robust FPGA-based hand gesture recognition system has been realized in this study. The main contribution of this paper is the combination and modification of several algorithms according to the hardware specification. The computational complexity of optical flow was improved to perform...
parallel processing in hardware architecture. Based on RTOS, system I/O and detection as well as recognition algorithms can be efficiently scheduled in real-time applications. For resource saving, all of the logical elements in implementation were limited to below 25% capacity in the hardware system. The speed of gesture detection and recognition can achieve 30 fps, and the system software can subsequently schedule all tasks in processing.

6. References