Intelligent Automation And Soft Computing, 2019
Copyright © 2019, TSI® Press
Vol. 25, no. 2, 375–384
https://doi.org/10.31209/2019.100000099

Hardware Design of Codebook-Based Moving Object Detecting Method for Dynamic Gesture Recognition

Ching-Han Chen[a], Ching-Yi Chen[b], and Nai-Yuan Liu[a]

[a]Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan, ROC
[b]Department of Information and Telecommunications Engineering, Ming Chuan University, Taoyuan, Taiwan, ROC

ABSTRACT
This study introduces a dynamic gesture recognition system applicable in IPTV remote control. In this system, we developed a hardware accelerator for real-time moving object detection. It is able to detect the position of hand block in each frame at high speed. After acquiring the information of hand block, the system can capture the robust dynamic gesture feature with the moving trail of hand block in the continuous images, and input to FNN classifier for starting recognition process. The experimental results show that our method has a good recognition performance, and more applicable to real gesture-controlled human-computer interactive environment.

KEY WORDS: dynamic gesture recognition, IPTV, hardware accelerator, FNN classifier.

1 INTRODUCTION

HAND gestures offer a natural and intuitive communication modality for human–computer interaction. The advantage of controlling remote devices using gestures is that users do not need to be in direct contact with the devices to interact with the relevant applications. The use of gesture recognition for remote control is an effective strategy for environments with hygiene and safety concerns, or with operational restrictions that do not allow users to be in direct contact with the devices. A vision-based gesture control system must detect hand position prior to the recognition of command gestures; meaning that the entered image sequence must be processed and analyzed to determine the correct hand position before gestures can be recognized. Among the various moving object detection methods, point detectors possess superior invariance for changes in light and camera angles, and are often used to locate useful data points in images that exhibit texture characteristics (Lowe, 2004). Apart from point detectors, mean-shift clustering (Comaniciu & Meer, 2002; Yin, Yang, Chai, & Yang, 2012), graph-cuts (Shi & Malik, 2000), Kalman filter (Yin, Peng, Chai, & Fan, 2013; Tsui & Basir, 2013), and supervised learning mechanisms (Rowley et al., 1998; Viola et al., 2005; Hsia et al., 2015) are also commonly applied in the processing of moving object detection and tracking.

The use of background models for object detection and tracking has also been a popular research topic (Brutzer et al., 2011). Since the late 1970s, researchers have been analyzing the color difference between multiple images to detect moving objects within image sequences (Jain & Nagel, 1979); other methods such as the Gaussian mixture model (GMM) (Oliver et al., 2000) and nonparameter kernel density estimation (Elgammal et al., 2002) also possess satisfactory effects for moving object detection. Most methods in the literature are only capable of processing static backgrounds, but Zhong et al. (2003) and Monnet et al. (2003) proposed background modeling methods with time-varying background capacities to surmount that limitation. These methods are capable of processing dynamic backgrounds such as water and clouds, further expanding the applicable scope of background modeling methods. Kim et al. (2005) proposed a real-time codebook model. Sample background values at each pixel are quantized into codebooks that represent a compressed form of the background model. Codewords not appearing for a long period of time in the video sequence are eliminated from the codebook model and new images that have appeared for some time are quantized into codebooks. This algorithm is also very effective for dynamic backgrounds (Lee & Park, 2012). Guo et al. proposed a multi-layer codebook model for background subtraction method. It can detect the

CONTACT Ching-Han Chen, pierre@csie.ncu.edu.tw

moving object efficiency and remove most of the nonstationary background (Guo et al., 2011; Guo et al., 2013).

After the removal of complicated backgrounds in image sequences and the correct segmentation of the portions of images depicting hands, processes such as feature extraction and sample classification are implemented. In general, the greatest challenge of dynamic gesture recognition is recognizing the same gestural movements performed by users at differing times; their speeds, trajectories, or durations differ, and result in spatial-temporal variability, a feature that makes the recognition of dynamic gestures more difficult than the recognition of static gestures. Therefore, extracting satisfactory gesture features and choosing suitable classifiers are extremely crucial for dynamic gesture recognition. During feature extraction, researchers often obtain useful trajectory features through analyzing the trajectories of image sequences, by details such as trajectory points, trajectory lengths, and orientation features, to ensure that the satisfactory features enable the system to accurately distinguish between differing gestures. Some classifiers that are often used in dynamic gesture recognition are finite state machines (Hong, Turk, & Huang, 2000), dynamic time warping (Corradini, 2001), hidden Markov models (HMMs) (Elmezain et al., 2008; Elmezain et al., 2009), and neural networks (Kim & Park, 2013, Kaluri & Reddy, 2018).

This study proposed a fuzzy neural network (FNN)-based real-time dynamic gesture recognition framework using trajectory features. The system uses the six basic dynamic gestures of the Internet Protocol Television (IPTV) gesture control interface as the target of recognition, to establish a gesture control system suitable for IPTV. To achieve real-time requirements, a codebook-based target object-detecting module was implemented as a hardware accelerator using field-programmable gate array (FPGA) technology. A pipeline design procedure was then adopted to segment the background modeling process into several stages, in order for all hardware modules to be fully applied and to avoid idle time, thereby effectively improving system performance. During the recognition stage, a coarse-to-fine classifier was established for gesture recognition; in the coarse classification stage, the FNN-based classifier divided the input gestures into linear and circular gestures, after which a detailed categorization was conducted based on information such as position changes in the adjacent data points of the trajectories, to accurately determine the category of the output gesture. Figure 1 shows the system architecture for the described method; the real-time object detection system was implemented on the DE2-115 platform using FPGA technology, and the gesture recognition module was implemented on the PC platform.

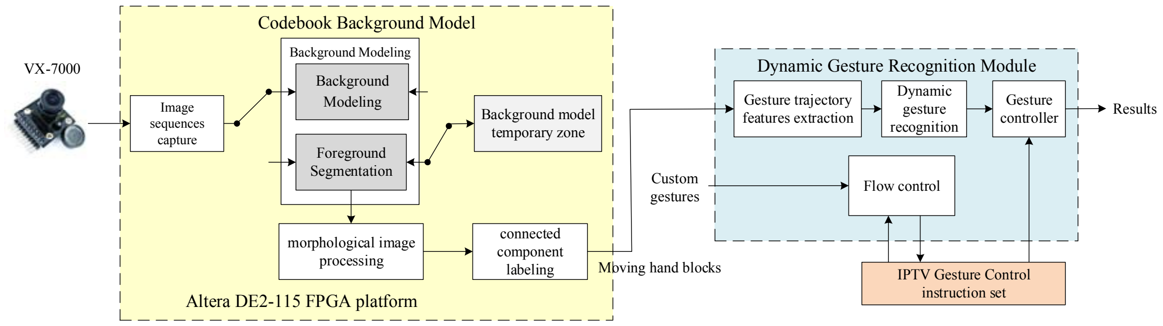

VX-7000
Codebook Background Model
Image sequences capture
Background Modeling
Background Modeling
Foreground Segmentation
Background model temporary zone
morphological image processing
connected component labeling
Altera DE2-115 FPGA platform
Moving hand blocks
Dynamic Gesture Recognition Module
Gesture trajectory features extraction
Dynamic gesture recognition
Gesture controller
Results
Custom gestures
Flow control
IPTV Gesture Control instruction set

Figure 1. System architecture of the real-time dynamic gesture recognition system.

2 HARDWARE ARCHITECTURE FOR MOVING OBJECT DETECTION

THE hardware architecture for the object detection system was divided into several circuit modules such as the codebook background modeling module, the morphological image processing module, and the connected component labeling (CCL) module. Figure 2 shows the Grafcet discrete-event model (David, 1995) for the object detection system. The first n input images were set as the background-training images; the n+1th image entered the foreground-segmentation stage, followed successively by morphological image processing and CCL.

2.1 *Codebook background model*

2.1.1 Kim et al.'s codebook method (Kim et al., 2005)

The basic codebook algorithm was proposed by Kim et al. (2005). It was based on the construction of a background model adopting a quantization and clustering technique from Kohonen (1990) and Ripley (1996). The algorithm in the present research is as follows:

Let X be a training sequence for a single pixel consisting of N red–green–blue (RGB) vectors: $X = \{x_1, x_2, \dots, x_N\}$, $C = \{c_1, c_2, \dots, c_L\}$ represent the codebook for the pixel consisting of L codebooks.

Each codeword c_i, $i = 1,\dots,L$ is represented by a RGB vector $v_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$ and a 6-tuple $aux_i = (\check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i)$ where $\check{I}_i$ and $\hat{I}_i$ are the minimum and maximum brightness of all pixels assigned to codeword c_i; f_i is the frequency with which the codeword has occurred; λ_i is the maximum negative run-length defined as the longest interval during the training period in which the codeword has not recurred; and p_i and q_i are the first and last access times, respectively, at which the codeword has occurred.

The codebook is created or updated using two criteria. The first criterion is based on color distortion δ whereas the second is based on brightness distortion. When we have an input x_t and a codeword c_i where $v_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$, the color distortion δ can be calculated by:

$$\delta = \sqrt{\|x_t\|^2 - \varphi^2} \qquad (1)$$

where φ^2 is given by (2)

$$\varphi^2 = \frac{(\bar{R}_i R + \bar{G}_i G + \bar{B}_i B)^2}{\bar{R}_i^2 + \bar{G}_i^2 + \bar{B}_i^2} \qquad (2)$$

According to Kim et al.'s method (2005), the brightness I is delimited by two bounds. The lower bound is $I_{low} = \alpha\hat{I}$ and the upper limit is $I_{hi} = min\left\{\beta\hat{I}, \frac{I}{\alpha}\right\}$. For an input pixel that has red, green, and blue colors, the formula of the brightness is given by (3).

$$brightness(I, \langle\check{I}, \hat{I}\rangle) = \begin{cases} true, & I_{low} \le \|x_t\| \le I_{hi} \\ false, & otherwise \end{cases} \qquad (3)$$

For each input pixel, if we find a codeword c_m that respects these two criteria (distortion criterion and brightness criterion) then we update this codeword by setting v_m to $\left(\frac{f_m\bar{R}_m + R}{f_m+1}, \frac{f_m\bar{G}_m + G}{f_m+1}, \frac{f_m\bar{B}_m + B}{f_m+1}\right)$ and aux_L to $\{min\{I, \check{I}_m\}, max\{I, \hat{I}_m\}, f_m + 1, max\{\lambda_m, t - q_m\}, p_m, t\}$. If we do not find a matched codeword, we create a new codeword c_L. In this case, v_L is equal to (R, G, B) and aux_L is equal to $\{I, I, 1, t-1, t, t\}$.

After the training period, if an incoming pixel matches a codeword in the codebook, then this codeword is updated. If the pixel does not match, the pixel's information is put into a cache word and this pixel is treated as a foreground pixel. If a cache word is matched at other frequencies then that cache word is put into the codebook.

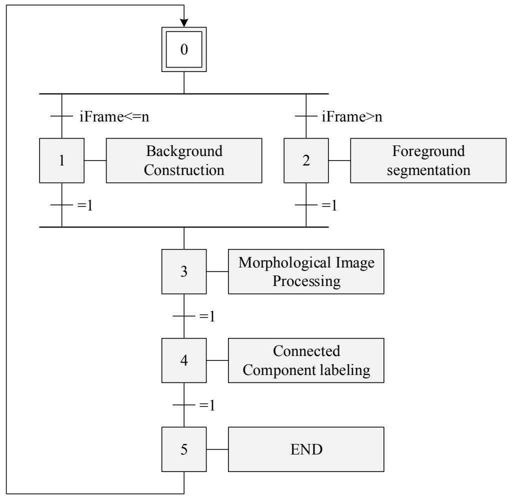
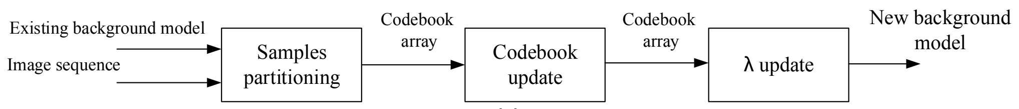


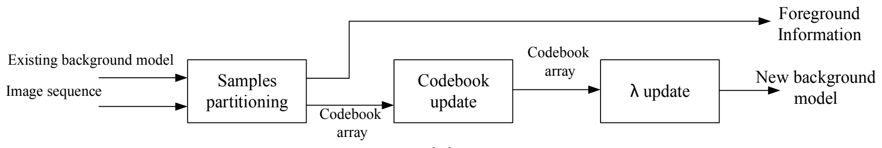
Figure 2. Grafcet discrete-event model of moving object detection system.

2.1.2 Hardware implementation of codebook background model

The codebook background modeling was mainly divided into the background-training stage and foreground-segmentation stage, thus the overall system was segmented into two modules; their architecture diagrams are shown in Figure 3(a) and 3(b) respectively. In addition, background modeling was further subdivided into three submodules, namely the sample partitioning module, the codebook update module, and the λ update module. When training the background model, the sample partitioning module inputs the color distortion and brightness of each pixel to determine whether that pixel belongs to the foreground or the background. The output results were binary data, with the result being 1 if the pixel was foreground, and 0 otherwise. The codebook update module and λ update module each updated the background codebook and the parameter λ according to the update rule in the aforementioned codebook background-training algorithm.



(a)



(b)

Figure 3. System architecture of codebook background modeling, (a) background model training stage, (b)foreground segmentation stage.

To enhance the operational efficiency of the system, the design work for the codebook background modeling was implemented by using a pipelined architecture. The pipelined design segmented the overall process of the hardware circuit into several stages. Each stage was capable of operating continuously in a cycle; this reduced the idle time of the hardware components and maximized their utilization; the system performance was improved without additional hardware resources. The pipeline controller designed in this study viewed each circuit module as a task and implemented conditional control on them. Figure 4 shows the Grafcet discrete-event model. In Figure 4, the sample partitioning module, codebook update module, and λ update module correspond to tasks controlled by the pipeline controller; P1–P3 were the main operating functional modules; T1–T3 and B1–B3 were in the idle state; their main function was to stabilize the transmission of control signals.

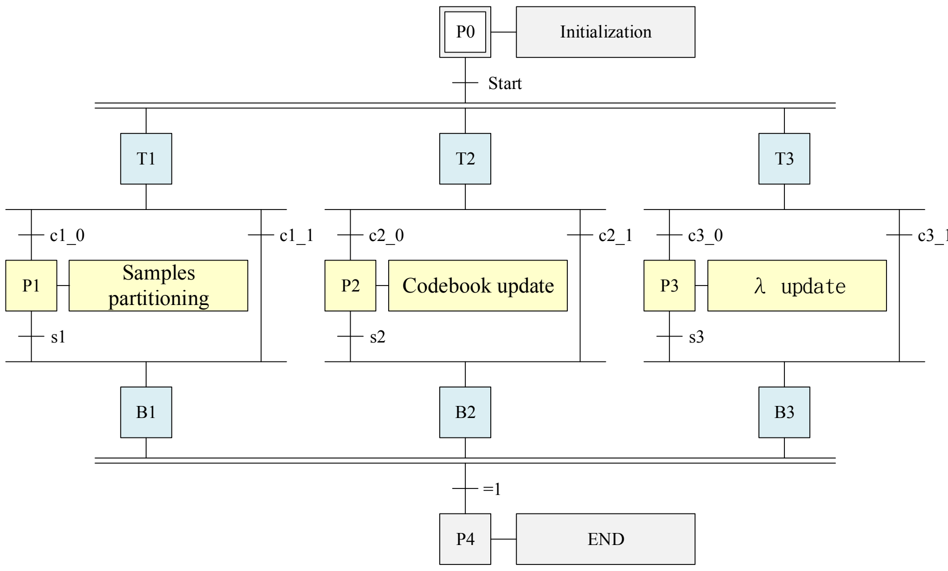


Figure 4. Grafcet discrete-event model of the pipeline controller.

2.2 Morphological image processing module

The opening operator in the mathematical morphology first performed erosion on images with filtered backgrounds to eliminate the noisy or sharp parts of the images; next, the contour gaps in the foreground objects were filled using dilation to obtain a complete foreground object contour, as shown in Figure 5. Figure 6 shows the block diagrams of the hardware architectures for both erosion and dilation operations. When the start signal was at logic 1, each occurrence of a clock signal caused Data_In to shift a pixel signal into the line buffer; when the data temporarily stored in the line buffer were 3x3, the 3x3 mask data were then transferred to the compare module, wherein the maximum value for the 3x3 mask data was determined and sent to the output terminal.

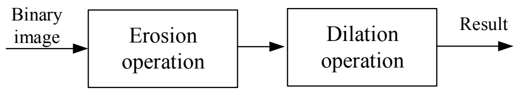


Figure 5. The morphological open operation.

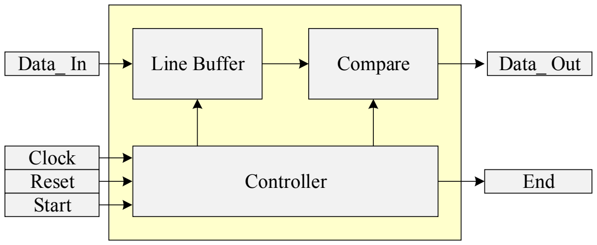


Figure 6. The hardware architecture for the dilation and erosion operation.

2.3 Connected component labeling module

The CCL module was able to determine whether a pixel point in the foreground shared a connected relationship with an adjacent foreground pixel point, that is, it determined whether the two foreground pixel points belonged to the same object. Binary images were scanned by the CCL using raster scanning from left to right and top to bottom; when each pixel was scanned, only the four points q, r, s, and t in its neighboring pixels needed to be considered, as shown in Figure 7. The CCL algorithm was divided into two stages: in the first stage, each pixel in the foreground was assigned a number based on the concept of the 8-connected component; the interconnected blocks with differing numbers were then merged in the second stage.

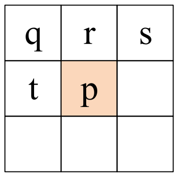


Figure 7. Adjacent points of the 8-connected component.

Figure 8 shows the hardware architecture for the CCL. The data for the four pixels of q, r, s, and t in the 8-connected component were stored using the line buffer, whereas the labeling information was obtained through the label assigner module and merge controller module, after which the labeling information was stored in the merge table and data table.

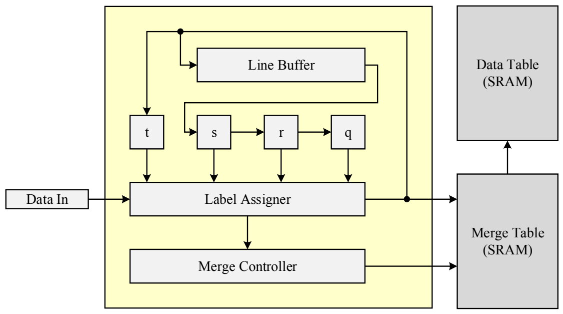


Figure 8. The hardware architecture for the connected component labeling module.

3 RECOGNITION SYSTEM STRUCTURE

3.1 Gesture feature extraction

(1) Ratio of major axis to minor axis

FOR a sequence of M images, the central location of the hand for each image was recorded, and the four boundary values of the gesture trajectories established by the M coordinate points were located: X_{max}, X_{min}, Y_{max}, Y_{min}.The major axis a and minor axis b for the gesture trajectories were then calculated, as shown in Figure 9(a). The first feature of the dynamic gesture t_1 was defined as:

$$t_1 = \frac{a}{b} = \frac{X_{max}-X_{min}}{Y_{max}-Y_{min}} \quad (4)$$

(2) Difference between major axis and minor axis

Similar to the approach for t_1, the difference between the major and minor axes (t_2) for the second feature of the dynamic gesture was defined as:

$$t_2 = a - b = (X_{max} - X_{min}) - (Y_{max} - Y_{min}) \quad (5)$$

(3) Occurring frequencies of the clockwise and counterclockwise elements

After X_{max}, X_{min}, Y_{max}, Y_{min} had been located, the intersection of a and b was viewed as the central point C of a virtual circle, with $r = \frac{a+b}{2}$ being the radius of the virtual circle, as shown in Figure 9(b). For the central coordinates of each block, the angle θ between the connecting line from the point to the center C and the horizontal line of the circle was determined, as shown in Figure 9(c) (Chen *et al.*, 2013). We can decide the gesture is clockwise or counterclockwise by M different angels (θ), and record the numbers of times that clockwise (f_{CW}) and counterclockwise (f_{CCW}) occur. We make the clockwise and counterclockwise frequency t_3 as $f_{CW} - f_{CCW}$.

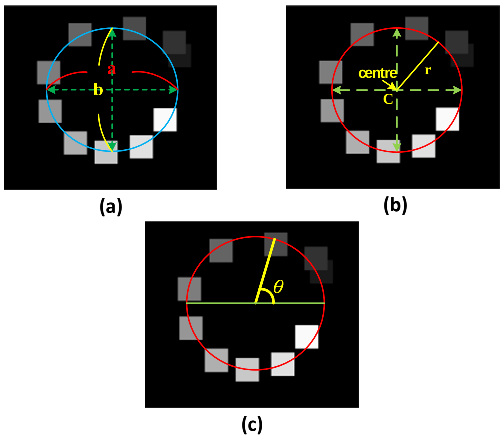


Figure 9. Definition of the three feature parameters of the gesture trajectory, (a) Ratio of major axis to minor axis, (b) Difference between major axis and minor axis, (c) Occurring frequencies of the clockwise and counterclockwise.

3.2 Dynamic gesture classifier design

The gestures identified in this study were the six most commonly used dynamic gestures in the IPTV gesture instruction set, namely left, right, up, down, clockwise, and counterclockwise, as shown in Figure 10.

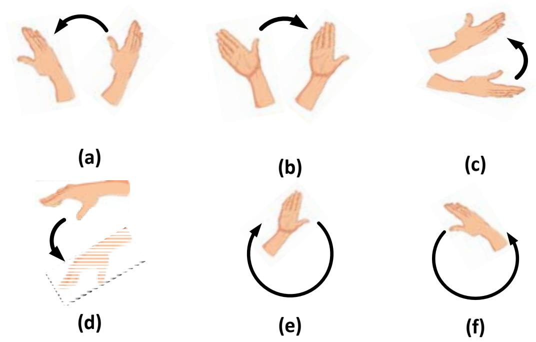


Figure 10. Six basic gestures in the IPTV gesture instruction set: (a)left, (b)right, (c)up, (d)down, (e)clockwise, (f)counterclockwise

The classifier design for the dynamic gestures was implemented using a FNN classifier (Chen et al., 2013), and its architecture is shown in Figure 11. The classifier was constituted of multiple Mamdani-type fuzzy rules and a single neuron. A classical fuzzy system consists of 3 stage: input stage, processing stage, and output stage. The input stage converts the crisp input into fuzzified variables using the membership functions. The processing stage invokes each appropriate rule and generates a result for each, then combines the results of the rules. Finally, the output stage converts the combined result back into a specific control output value. In this paper, a fuzzy rule has the form shown in (6):

$$R^j : \text{If } x_1 \text{ is } A_1^j, \ldots, \text{and } x_n \text{ is } A_n^j,$$
$$\text{then } X \text{ belong class } y_j \text{ with } CF = CF_j \qquad (6)$$

where x_1, …, x_n are input variables of the fuzzy rule, $y_j \in \{1,2,\ldots,S\}$ is the output variable of the jth fuzzy rule, which is one for the S class. A_k^j is the fuzzy set of the kth input variable, CF_j represents the reliability of the fuzzy rule R^j, j=1,2,…,r. We apply the generalized modus ponens and max-min composition operation, and the fuzzy inference outputs expressed as follows:

$$\mu_A^j(X) = min\left\{\mu_{A_1^j}(x_1), \ldots, \mu_{A_n^j}(x_n)\right\} \qquad (7)$$

$$u_y^j = \mu_A^j(X) \cdot CF_j \qquad (8)$$

The output values u_y^j of each fuzzy rule were linked to the input terminal of a single neuron, which executed the defuzzification process to obtain a crisp output. The r inputs transmitted to the single neuron were first calculated with the weight values of the neuron, after which the result was input into the transfer function φ to obtain the final output, v, as follows:

$$v = \sum_{j=0}^{r} W_j \cdot u^j \qquad (9)$$

$$y = \varphi(v) = \frac{\alpha_2}{1+e^{-\alpha_1 \cdot v}} \qquad (10)$$

where α_1 controls the shape of the transfer function, α_2 adjusts the size of scale. Figure12 shows the combination of the fuzzy system and the single neuron.

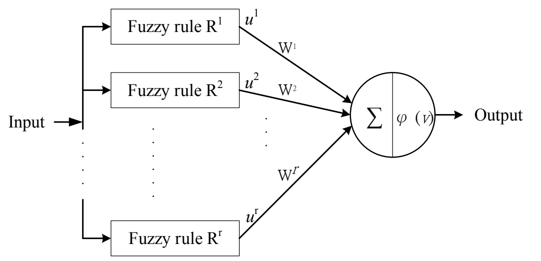
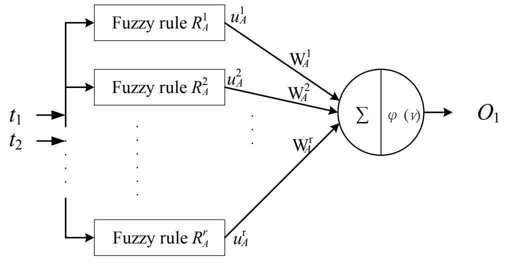


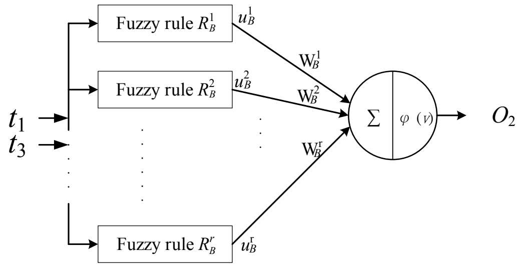
Figure 11. **The architecture of the proposed fuzzy neural network classifier.**

3.2.1 Coarse classification of dynamic gestures

Based on the classifier architecture in Figure 11, the two classifiers used to determine the circular gestures and linear gestures were established separately, as shown in Figure 12(a) and 12(b). The input features of the linear gesture classifier were t_1 and t_2, and its output value O_1 was the inferred probability of the input gesture belonging to the set of linear gestures; the input features of the circular gesture classifier were t_1 and t_3, and its output value O_2 was the inferred probability of the input gesture belonging to the set of circular gestures.



(a)



(b)

Figure 12. **The FNN classifier used for dynamic gesture recognition, (a) linear gesture classifier, (b) circular gesture classifier.**

The FNN classifier can roughly determine the dynamic gestures to be either linear gestures or circular gesture. Further, in accordance with the positional relationship of the gesture trajectory coordinates, it determines the dynamic gesture to be up, down, left, right, clockwise, or counterclockwise.

3.2.2 Fine classification of dynamic gestures

The position changes between all adjacent points (x_i, y_i) and (x_{i+1}, y_{i+1}) were determined using the M coordinate points that constituted the dynamic gesture trajectory. To reflect the fact that the gesture movements captured by the camera moved in directions opposite to the directions perceived by the users, the determined formula definitions are shown in (11)：

$$\begin{cases} C_{left} = C_{left} + 1 & , if\ x_{i+1} > x_i \\ C_{right} = C_{right} + 1 & , if\ x_i > x_{i+1} \\ C_{up} = C_{up} + 1 & , if\ y_i > y_{i+1} \\ C_{down} = C_{down} + 1 & , if\ y_{i+1} > y_i \end{cases} \qquad (11)$$

where C_{left}, C_{right}, C_{up}, and C_{down} are the times of waving in different directions.

Combining with the times of waving clockwise f_{CW} (C_{CW}) and counterclockwise f_{CCW} (C_{CCW}), we define the probabilities of the six dynamic gestures and the formula is as following:

$$\begin{cases} p_1 = \frac{C_{left}}{C_{left}+C_{right}+C_{up}+C_{down}} \times O_1 \\ p_2 = \frac{C_{right}}{C_{left}+C_{right}+C_{up}+C_{down}} \times O_1 \\ p_3 = \frac{C_{up}}{C_{left}+C_{right}+C_{up}+C_{down}} \times O_1 \\ p_4 = \frac{C_{down}}{C_{left}+C_{right}+C_{up}+C_{down}} \times O_1 \\ p_5 = \frac{C_{CW}}{C_{CW}+C_{CCW}} \times O_2 \\ p_6 = \frac{C_{CCW}}{C_{CW}+C_{CCW}} \times O_2 \end{cases} \tag{12}$$

where $p_{1,\dots,}p_6$ are the probabilities of different dynamic gestures. O_1 and O_2 are the outputs of the linear classifier and the circular classifier.

4 EXPERIMENTAL RESULTS AND DISCUSSION

THE program was tested on Intel Core (TM)2 Quad CPU Q6660 with 4GB DDRIII RAM, and the spec of CMOS sensor was 2 million pixels. The designed hardware accelerator were described in VHDL and synthesized for Altera Cyclone Ⅳ FPGA, with the aid of the tool Quartus II 12.1.

4.1 Real-time object detection system

The results for the real-time object detection system under differing color temperatures are shown in this section. The color temperature level varied according to the viewers' various external conditions, such as weather conditions, environments, and objects; its unit was Kelvin (K); a lower value indicated a stronger red, whereas a higher value indicated a stronger blue. The experiment was conducted using four image sequences of differing color temperatures, which were an outdoor environment in the early morning (1600 K), an indoor environment under a fluorescent lamp (4000 K), and outdoor environment at noon (5200 K), and a cloudy day (7000 K). Figure 13 shows the results of the hand detection segmentation. The measured results indicated that even under distinct color temperatures, the real-time object detection system was capable of successfully segmenting the areas of the users' hands.

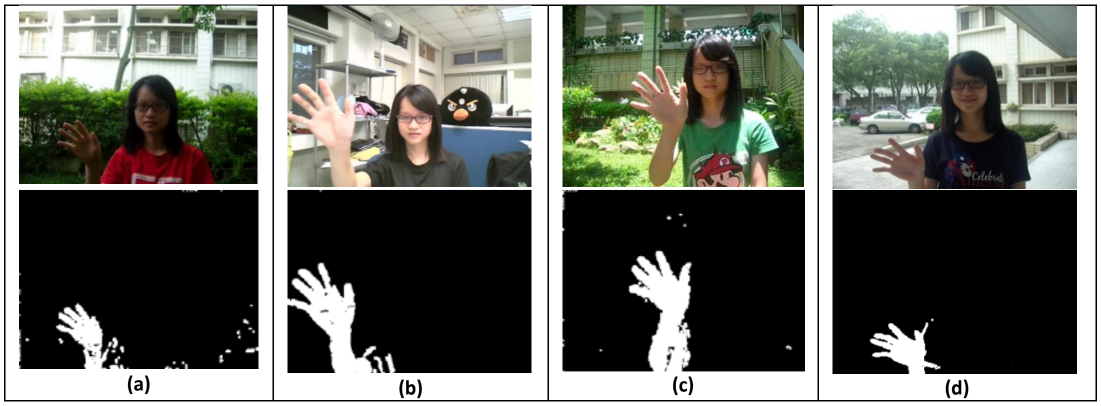


Figure 13. Results of image sequence and its segmented hand areas captured at different time periods (a) outdoor environment in the early morning, (b) indoor environment under a fluorescent lamp, (c) outdoor environment at noon, and, (d) outdoor environment on a cloudy day.

4.2 Dynamic gesture recognition results

For the gesture recognition experiment, the six dynamic gestures were captured as image sequences. Each gesture was recorded 10 times for each of 10 users, thus a total of 600 gesture data were stored in the gesture database, with each dynamic gesture having an image sequence length of 10 frames. At the training stage of the recognition architecture, 70 data were randomly selected from each dynamic gesture for FNN classifier training; the initial values for W_j of the FNN classifier were all preset to 1, and particle swarm optimization was employed during the training stage to optimize the parameters to improve the recognition rate of the classifier.

After obtaining a sequence of data showing moving hands, the real-time object detection system implemented on the FPGA board then sent the sequence to the gesture recognition system at the PC end to infer the probability that the sequence might belong to a known gesture category. Table 1 shows the experimental results for the recognition rate of dynamic gestures.

4.3 Computation efficiency analysis of the system

This study used Quartus II software, version 12.1, to implement and verify the hardware design of the real-time object detection system, which consisted of three circuit modules. Table 2 shows the analysis results for the FPGA resources consumed by the final implemented system.

Table 1. The recognition rate of dynamic gesture.

	1	2	3	4	5	6	7	8	9	10	Avg
Left	100%	100%	100%	100%	80%	90%	90%	90%	100%	80%	93%
Right	90%	100%	90%	100%	100%	100%	100%	90%	100%	100%	97%
Up	90%	100%	100%	100%	70%	80%	100%	90%	100%	100%	93%
Down	100%	100%	50%	100%	80%	70%	80%	70%	60%	60%	77%
Clockwise	100%	80%	100%	70%	100%	100%	90%	100%	100%	90%	93%
C-clockwise	100%	100%	100%	80%	100%	80%	100%	100%	90%	100%	95%
Avg.	97%	97%	90%	92%	92%	87%	93%	90%	92%	88%	92%

Table 2. Resource usage analysis of the object detection system on the DE2-115 development board

	Pins	Memory Bits	Embedded Multiplier 9-bit Elements	Logic Elements
Total Number of DE2-115 Resources	529	3,981,312	532	114,480
System Resource Usage	149	148,160	39	4,788
Resource Use Ratio	28%	3%	7%	4%

Next, the operating clock frequencies of the three hardware modules were analyzed using the Quartus II 12.1 synthesis software. The operating speed of the background modeling module was affected by two factors; the first factor was the operating clock frequency of image processing by the FPGA internal execution; the second factor was the operating clock frequency of the external synchronous dynamic random access memory (SDRAM) for storing information. If that SDRAM had frequently been required to switch between reading and writing, that would have affected the operational performance; to avoid inefficiency, the circuit was designed to use two SDRAM units, of which one was responsible for writing whereas the other was for reading. Read–write exchange was performed by the two SDRAM units after an image had been processed, thereby ensuring that the external SDRAM was capable of operating under the default operating clock frequency. Table 3 shows the operating clock frequencies for the three hardware modules in the real-time object detection system, with the operating clock frequency of the overall system reaching 107.63 MHz.

Table 3. Performance index table

Module	clock frequency
codebook background model	107.63 MHz
opening module	254.52 MHz
CCL module	149.50 MHz
system clock frequency	107.63 MHz

Table 4 compares the performance levels of the codebook model hardware architecture designed and implemented in this study and the embedded architectures described in other articles. Table 4 indicates that when the codebook model was implemented as a hardware circuit, its operating speed was indeed capable of achieving the purpose of real-time processing. Compared to the architectures proposed in other papers, the research method described in this study possessed the advantage of operating speed.

Table 4. Performance comparison with existing background model.

Method	Time cost (ms/frame)	Speed (fps)
Codebook model (FPGA hardware)	3.8	262.76
Codebook model (Intel CPU)	45.2	22.12
Frame differencing (Apewokin *et al.*, 2009)	7.6	131.96
Approximate median (Apewokin *et al.*, 2009)	8.5	117.33
Weighted mean (Apewokin *et al.*, 2009)	26.8	37.28
MoG (Apewokin *et al.*, 2009)	273.6	3.65

5 CONCLUSION

THE construction of background modeling in the foreground-segmentation stage can achieve a satisfactory effect; however, because of the long computation times required for background modeling, object detection systems based on background modeling are incapable of real-time processing. This study introduced a remote gesture recognition system applicable for IPTV; the overall system functionality comprised real-time object detection and gesture recognition. To achieve the requirement for real-time processing, FPGA technology was used to implement the real-time object detection system. An FNN-based real-time dynamic gesture recognition architecture using gesture trajectory features was proposed, thereby enabling the system to deliver satisfactory performance regardless of its operating speed or the gesture recognition accuracy. In addition, to provide users with more versatile gesture operations and enhance the expandability of the system, the Grafcet model was also used to establish a hybrid gesture command system model, enabling users to create new gestures according to their own requirements apart from the six basic gestures.

REFERENCES

S. Apewokin, B. Valentine, D. Forsthoefel, L. Wills, S. Wills, and A. Gentile, (2009). Embedded real-time surveillance using multimodal mean background modeling. In *Embedded Computer Vision*, B. Kisacanin, S. Bhattacharyya, & S. Chai, Eds., 163-175, Springer, New York, USA.

S. Brutzer, B. Hoferlin, and G. Heidemann, (2011). Evaluation of Background Subtraction Techniques for Video Surveillance. *2011 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 1937-1944.

C.H. Chen, K. Chang, N.Y. Liu, and G. Su, (2013). Dynamic Gesture Recognition Based on Fuzzy Neural Network Classifier. *International Conference on Advances in Computer-Human Interactions*, 57–61.

D. Comaniciu and P. Meer, (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 24(5), 603-619.

A. Corradini, (2001). Dynamic TimeWarping for Off-line Recognition of a Small Gesture Vocabulary. *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time System,* 82-89.

R. David, (1995). Grafcet: a powerful tool for specification of logic controllers. *IEEE Transactions on Control Systems Technology*. 3(3), 253-268.

A. Elgammal, R. Duraiswami, D. Harwood, and L.S. Davis, (2002). Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE,* 90(7), 1151-1163.

M. Elmezain, A. Al-Hamadi, and B. Michaelis, (2008). Real-time capable system for hand gesture recognition using hidden markov models in stereo color image sequences. *Journal of WSCG*, 16(1-3), 65–72.

M. Elmezain, A. Al-Hamadi, and B. Michaelis, (2009). Hand Trajectory-based Gesture Spotting and Recognition Using HMM. *IEEE International Conference on Image,* 3541-3544.

J.M. Guo, C.H. Hsia, Y.F. Liu, M.H. Shih, C.H. Chang, and J.Y. Wu, (2013). Fast background subtraction based on a multi-layer codebook model for moving object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(10), 1809–1821.

J.M. Guo, Y.F. Liu, C.H. Hsia, M.H. Shih, and C.S. Hsu, (2011). Hierarchical method for foreground detection using codebook model. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(6), 804–815.

P. Hong, M. Turk, and T.S. Huang, (2000). Gesture modeling and recognition using finite state machines, *IEEE International Conference on Automatic Face and Gesture Recognition*, 410–415.

C.H. Hsia, J.S. Chiang, and C.Y. Lin, (2015). A face detection method for illumination variant condition. *Scientia Iranica*, 22(6), 2081–2091.

R. Jain and H. Nagel, (1979). On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1(2), 206-214.

R. Kaluri and C.H.P. Reddy, (2018). Optimized feature extraction for precise sign gesture recognition using self-improved genetic algorithm. *International Journal of Engineering and Technology Innovation*, 8(1), 25-37.

H.J. Kim and S.J. Park, (2013). Rule Extraction for Dynamic Hand Gesture Recognition using a Modified FMM Neural Network. *International Journal of Software Engineering and Its Applications*, 7(6), 367- 374.

K. Kim, T.H. Chalidabhongse, D. Harwood, and L. Davis, (2005). Real-time foreground-background segmentation using codebook model. *Real-Time Imaging,* 11, 172-185.

T. Kohonen, (1990). Improved versions of learning vector quantization, *International Joint Conference on Neural Networks*, 1, 545–550.

J. Lee and M. Park, (2012). An Adaptive Background Subtraction Method Based on Kernel Density Estimation. Sensors, 12(9), 12279–12300.

D.G. Lowe, (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision,* 60(2), 91-110.

A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. (2003). Background modeling and subtraction of dynamic scenes. *International conference on computer vision* (*ICCV 2003*), 1305-1312.

N.M. Oliver, B. Rosario, and A.P. Pentland, (2000). A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 22(8), 831-843.

B.D. Ripley, (1996). *Pattern recognition and neural networks*. Cambridge University Press, Cambridge.

H.A. Rowley, S. Baluja, and T. Kanade, (1998). Neural network-based face detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(1), 23-38.

J. Shi and J. Malik, (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 22(8), 888-905.

P. C. P. Tsui and O. A. Basir, (2013). An evolutionary model for optimizing sensor pose in object motion estimation applications, *Intelligent Automation & Soft Computing*, 12(2), 127-149.

P. Viola, M. J. Jones, and D. Snow, (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision,* 63(2), 153-161.

H. Yin, C. Peng, Y. Chai, and Q. Fan, (2013). A robust object tracking algorithm based on surf and Kalman filter. *Intelligent Automation & Soft Computing*, 19(4), 567-579.

H. Yin, J. Yang, Y. Chai, and S.X. Yang, (2012). An improved mean-shift tracking algorithm using PSO-based adaptive feature selection. *Intelligent Automation & Soft Computing*, 18(6), 647-657.

J. Zhong and S. Sclaroff, (2003). Segmenting foreground objects from a dynamic textured background via a robust kalman filter. *IEEE International Conference on Computer Vision*(*ICCV*)*,* 44-50.

NOTES ON CONTRIBUTORS

Ching-Han Chen received the D.E.A degree in Informatique, Automatique et Productique in 1992 and Ph.D. degree in 1995 from the Franche-Comte University, France. He was an associate professor in the department of electrical engineering, I-Shou University from 1995. Since 2006, he is now an associate professor in the Department of Computer Science and Information Engineering, NCU, Taiwan. His research interests include embedded system design, multimedia signal processing, and robotics.

Ching-Yi Chen received his Ph.D. degree in Electrical Engineering from Tamkang University, New Taipei City, Taiwan, R.O.C., in 2006. He joined the Department of Information and Telecommunications Engineering, Ming Chuan University in 2007 and is now an Associate Professor. His main research interests include swarm intelligence, pattern recognition, and embedded systems.

Nai-Yuan Liu received her M.S. degree in the Department of Computer Science and Information Engineering from National Central University, Taiwan, R.O.C., in 2013. Her research interests include image processing and fuzzy system.