Hardware Architecture of a Haar Classifier Based Face Detection System Using a Skip Scheme

Jongkil Hyun[1], Junghwan Kim[1], Cheol-Ho Choi[1], and Byungin Moon[1,2*]
[1]Graduate School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, Korea
[2]School of Electronics Engineering, Kyungpook National University, Daegu, Korea
[*]Email: bihmoon@knu.ac.kr

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401114

***Abstract*—Face recognition applications are being widely studied owing to their extensive usability in the field of computer vision. However, processing an entire image requires a considerable amount of time. To reduce the processing time, several algorithms that extract only the face from the image during pre-processing are studied. Haar classifiers are extensively used for the hardware implementation of face detection algorithms that improve the processing speed of face classification. This paper proposes a Haar classifier based face detection architecture that removes unnecessary iterations during classification to further improve the processing speed. The proposed architecture improves the processing speed by 4.46% compared to that of conventional Haar classifier based face detection architectures, for face detection using a VGA image with 30 faces. The proposed architecture tends to improve the processing speed as the number of faces in the image increases while matching the detection accuracy of conventional methods. Additionally, this architecture can be widely applied to classification algorithms that are based on iterations.**

Keywords—FPGA, hardware architecture, face detection, Haar classifier

I. Introduction

Computer vision and deep learning are becoming increasingly significant with the rapid development of computing technology. Accordingly, image processing of human faces is being studied for identification purposes and applications pertaining to augmented reality and internet of things [1–3]. However, performing image processing on an entire input image has the drawbacks of difficulties in real-time operation and implementation using small devices; this is because image processing consumes a considerable amount of time and power owing to the large amount of computation involved. To overcome these drawbacks, the process of cropping only the facial area from the entire image is necessary prior to implementation. Several studies use face detection algorithms to classify only the facial area within an entire image, which is subsequently used as an input for image processing to improve the efficiency of the entire image processing system [4–7]. Face detection algorithms are classified into feature-based [4], appearance-based [5], knowledge-based [6], and template-matching [7] methods. Among these algorithm-based methods, the feature-based method can detect the facial area in real-time because it utilizes low-level features, such as color and shape. The Haar classifier algorithm in the Viola–Jones Object Detection Framework is a feature-based method that detects human faces using the features trained by the AdaBoost machine learning technique [8]. The Viola–Jones algorithm has the advantage of being able to detect not only the frontal face, but also the side of the face or the accessories worn, depending on the learning data; further, this algorithm entails a low computational cost and ensures a high face detection rate. Moreover, because the classification of the Viola–Jones face detection algorithm is relatively simpler than the other algorithms, several studies are attempting to further reduce the power consumption and improve the processing speed by implementing the algorithm on hardware [9]. In one of the studies that implemented the Viola–Jones face detection algorithm on hardware, sub-window and an image-resizing method were used to detect multiple faces in one image; further, an architecture that considerably improves the face detection processing speed through the pipelining of classifiers was proposed [9]. The face detection hardware architecture in [9] exhibits a processing speed approximately 35 times faster than that of the software-based Viola–Jones algorithm. Despite several performance improvements, the processing speed is still limited because the detection operation needs to be performed repeatedly for all the sub-windows in the image. Therefore, we propose a face detection hardware architecture using a novel skip scheme that reduces the total number of iterations and promotes a faster processing speed. The classification of the sub-windows adjacent to the sub-window wherein the face is detected is skipped, and therefore, the total number of iterations is reduced. Consequently, the processing speed of the proposed architecture is improved.

II. Skip Scheme

Fig. 1 shows the process of a cascade classifier using Haar-like features that are learned via AdaBoost. Each stage of the process entails several Haar-like features. A sub-

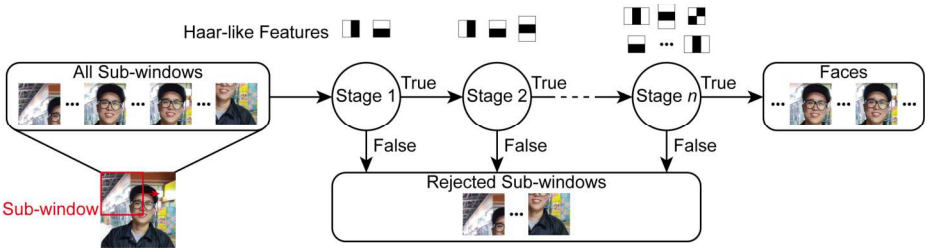

Fig. 1. Process of cascade classifier

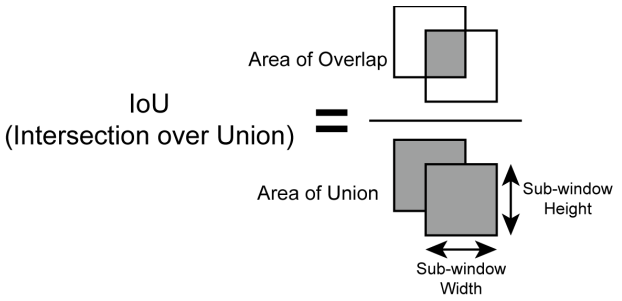

Fig. 2. Calculation of intersection over union (IoU)

window is inputted into the cascade classifier and is classified in each stage. Even if the classification in one stage is unsuccessful, the sub-window is determined not to be a face and the classification process is terminated immediately. The same face can be detected repeatedly in the region that is adjacent to a facial sub-window because the classification of the sub-windows progresses pixel by pixel. Intersection over union (IoU) is commonly used to merge the detection results of the same face. The degree of proximity is determined by calculating the IoU between the sub-windows, as shown in Fig. 2. When the calculated IoU is greater than the threshold value, the sub-windows are judged to be the same face and merged. Since the results of the same face are merged, the computation for the adjacent sub-windows can be skipped. Therefore, the skip scheme does not accommodate redundant computations of the sub-windows that are adjacent to a facial sub-window. The proposed method improves the processing speed because the number of unnecessary computations performed by the cascade classifier is reduced.

III. Hardware Architecture

Fig. 3 shows the proposed face detection architecture. The Frame Grabber receives the pixel data from the camera and stores the image of one frame in the Frame Buffer. The image is retained until the face detection is realized. Then, a new image is received and the Frame Buffer is updated accordingly. The Scaler resizes the image stored in the Frame Buffer and creates an image pyramid to detect faces of various sizes within the image. The Cascade Classifier that consists of the Integral Image Generator and Classifier, performs face classification using the Haar classifier. The Integral Image Generator generates an integral image for each sub-window for the resized image. Thereafter, the Classifier performs face classification using the integral image. The Face Merger receives classification results and coordinates for each sub-window from the Cascade Classifier and merges the sub-windows to detect the particular face. The conventional classifier must repeatedly perform classification for all the sub-windows in the image. In other words, because the number of iterations of the classifier is large, the conventional classifier takes a long time to classify the entire image. Reducing the number of iterations is directly linked to improving the processing speed of the entire face detection. Therefore, the proposed architecture reduces the number of iterations by skipping the classifications of the sub-windows that are adjacent to the sub-window wherein the face is detected. Since the information of the adjacent sub-windows are classified as the same face, the sub-windows are eventually merged by the Face Merger. The proposed architecture can improve the processing speed while maintaining the face detection rate without unnecessary iterations.

A. Scaler

The Scaler in the proposed architecture generates the address of the stored image in the Frame Buffer to perform image resizing. The Scaler resizes the image by applying the nearest-neighbor interpolation method with a scale factor of 1.2. The scale factor is updated by multiplying 1.2 as the downsizing step changes. The multiplication for scale factor is implemented using a number of logic-level shifters and adders. The row and column coordinates of the resized image are produced by continuously accumulating the scale factor. Subsequently, the Scaler generates an address for accessing the Frame Buffer. The address uses only the integer value of the coordinates. By using this address as the read address of the Frame Buffer, the image can be downsized without extensive computations. The Scaler produces a downscale image pyramid through a total of 18 resizing steps to facilitate the detection of faces of various sizes within the image.

B. Integral Image Generator

The Integral Image Generator receives the resized image pixel from the Scaler and stores it in the Line Buffers to

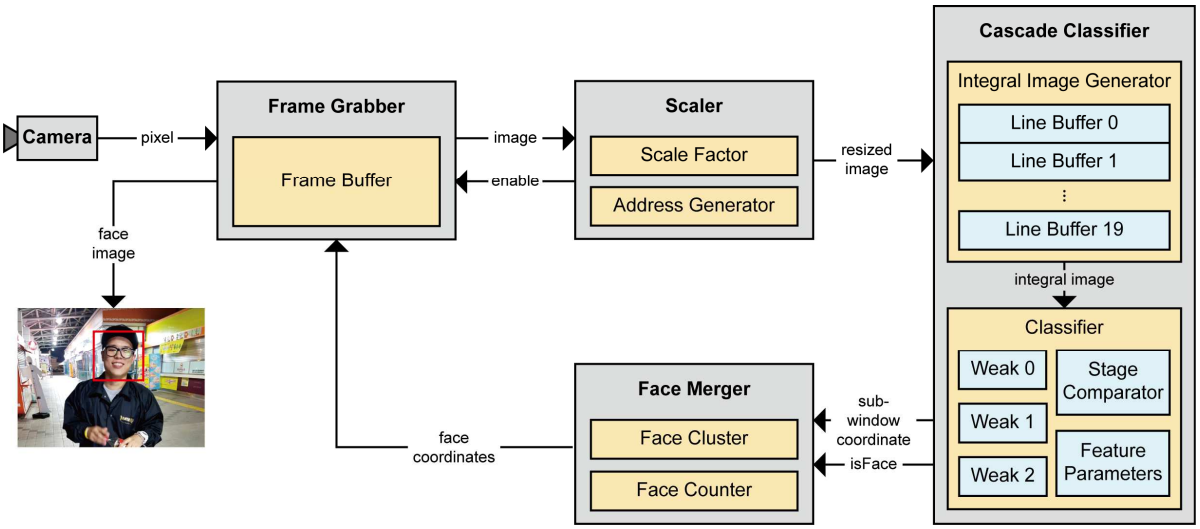

Fig. 3. Proposed face detection architecture

generate the integral image. Each Line Buffer stores one horizontal line of pixels of the resized image. When the sub-window slides, the Integral Image Generator reads 20 vertical pixels from the 20 Line Buffers and performs vertical cumulative summation operation through an adder tree [10]. Thereafter, the leftmost values of the integral image are added to the result of the vertical cumulative sum of each row to generate a new vertical integral image. The rightmost values of the integral image are subtracted from all the other elements of each row, and the new vertical integral image is included to update the 20 × 20 integral image. The Integral Image Generator is designed in a pipeline-structure such that the integral image can be updated for each clock cycle. The Integral Image Generator is paused while the classification for the current sub-window is in progress. Subsequently, the integral image is immediately updated for the next sub-window once the classification is completed.

C. Classifier

The Classifier performs face classification on the input integral image of the sub-window. The Classifier comprises three parallel weak classifiers. The weak classifiers communicate with the block memory to compute the values of each feature by utilizing the feature parameters that contain information about the face. Each calculated feature value is compared with the feature threshold of the human face; the leaf values obtained from the comparison results are accumulated for each stage. The stage comparator compares the accumulated leaf value with the stage threshold value of the human face to determine the success of each stage. Even if the classification in one stage is unsuccessful, the sub-window is determined not to be a face and the classification process is terminated immediately; thereafter, the classification for the next sub-window begins. On the contrary, if all the stages are successful, the sub-window is determined to be a face image, and a skip signal is generated. The skip signal is used to skip the unnecessary classification process of the sub-windows that are adjacent to the sub-window wherein the face is detected. The skip signal is on until the IoU is less than the threshold value in the horizontal direction. Through this skip scheme, the proposed architecture improves the overall face detection processing speed.

D. Face Merger

The Face Merger stores the coordinates of the sub-windows where the face is detected. When a face is detected in a new sub-window, the coordinates of the sub-window are compared with the previously stored coordinates. If the result of comparison of the two sub-window coordinates is within a certain distance, the compared sub-windows are merged to minimize the duplication detection of the same face. In the proposed architecture that uses sub-window of 20 × 20 pixels, if the two sub-windows differ by less than 6 pixels horizontally, the IoU will be greater than 0.5. Therefore, the sub-windows that are within the horizontal distance of 5 pixels from the sub-window wherein the face is detected will be merged; therefore, the classification of the sub-windows is skipped. To ensure that the proposed face detection architecture can detect up to 30 faces in one image, the Face Merger is designed to store 30 different face coordinates.

IV. Experimental Results

The classifier of the proposed architecture is implemented using the haarcascade_frontalface_alt feature parameter of OpenCV [11]. The haarcascade_frontalface_alt was trained on the 20 × 20 frontal face sub-windows and contains a total of 22 stages and 2135 features. The number of features for each stage is summarized in Table I.

The proposed architecture has been designed using Verilog HDL, and the resource usage has been measured using the Xilinx Virtex-5 LX115 field programmable gate array (FPGA) to compare with the architecture of [9]. The resource usage is similar to that in [9] because the proposed architecture is an extension of the architecture proposed in [9]. Table II compares the resource usage of the architecture in [9] and the proposed architecture of the current study. The proposed architecture uses a greater number of BRAMs because the Frame Grabber of the proposed architecture stores color images, rather than gray scale images. If the Frame Grabber of the proposed architecture would store solely gray scale images, the resource usage would almost be the same as that in [9]. In addition, the number of slice registers and slice look-up tables (LUTs) used in the proposed architecture is less than that used in [9] because this architecture uses a greater number of digital signal processers (DSPs). If the number of DSPs used is equal to that in [9], the functionalities of the Slice Register and Slice LUTs will become similar.

Further, Table II also summarizes the resource usage of the face detection architecture based on whether the skip scheme is applied. The number of Slice Registers and Slice LUTs used are increased slightly, 0.1% and 0.5%, respectively, when the skip scheme is applied. The increase in resource usage occurs due to the additional control signals required for the Classifier and the Integral Image Generator to apply the skip scheme. However, the slight increase in resource usage due to the application of the skip scheme has a minor effect relative to the improvement in the processing speed.

TABLE I. Number of Features for Each Stage

Stage	Number of features	Stage	Number of features	Stage	Number of features
0	3	8	56	16	140
1	16	9	71	17	160
2	21	10	80	18	177
3	39	11	103	19	182
4	33	12	111	20	211
5	44	13	102	21	213
6	50	14	135	**Total**	2135
7	51	15	137		

TABLE II. Hardware Utilization for Face Detection System

	Architecture of [9]	Proposed architecture (without skip scheme)	Proposed architecture (with skip scheme)
Slice Registers	21,902	12,931	12,944
Slice LUTs	84,232	67,590	67,934
BRAMs	97	176	176
DSP 48E	7	125	125

Table III compares the performance of the proposed face detection architecture and the architecture of [9] for 640 × 480 (VGA) resolution images. Both the architectures use three parallel weak classifiers. The skip scheme decreases the processing time by 0.01 ms for 1 face, 0.75 ms for 6 faces, 1.74 ms for 11 faces, and 7.18 ms for 30 faces. For 30 faces, the proposed skip scheme decreases the processing time by approximately 4.46%. In addition, the detection accuracy is similar to that of conventional methods because the skip scheme is used only when faces are detected. The proposed architecture is more effective when there are multiple faces in the image because the more the number of faces detected, the more classifications are skipped.

The proposed architecture is implemented using Xilinx ZC706 and the test configuration is shown in Fig. 4. The camera interface board receives the video from the camera and transmits the video to FPGA every clock cycle. The proposed face detection architecture is implemented on the FPGA board. The FPGA board stores one image frame that is input from the camera interface board and outputs the final detection result back to the camera interface board when the face detection is completed. Thereafter, the camera interface board transmits the face detection result to the PC that visualizes and verifies the detection result through a USB interface. The face detection result obtained by implementing the proposed architecture using FPGA is shown in Fig. 5.

V. Conclusion

This paper proposes a face detection architecture that does not require classification of the sub-windows that are horizontally adjacent to the sub-window wherein the face is detected. Because the proposed architecture skips the sub-windows that are eventually merged in the face merge step, the detection accuracy is the same as that of conventional methods. The proposed face detection architecture improves the overall processing speed by removing unnecessary iterations, while ensuring that the resource usage is similar to that of conventional methods. The processing speed of the proposed architecture exhibits an improvement of 4.46% compared to that of conventional methods, for face detection using 640 × 480 images (produced by the camera at 60 fps) containing 30 faces. Although the resource usage is slightly increased, it is insignificant compared to the improvement of the processing speed. The skip scheme is widely utilized because it can be applied not only to the proposed architecture, but also to all the architectures that entail classifications based on the iteration method. In the future, we intend to study the face detection architecture in depth to improve the performance and reduce the number of iterations further.

TABLE III. Performance for Face Detection System

Number of faces	Architecture of [9]	Proposed architecture (without skip scheme)	Proposed architecture (with skip scheme)
1	133.14 ms (7.51 fps)	138.48 ms (7.22 fps)	138.47 ms (7.22 fps)
6	146.745 ms (6.81 fps)	145.85 ms (6.86 fps)	145.10 ms (6.89 fps)
11	152.66 ms (6.55 fps)	152.08 ms (6.58 fps)	150.34 ms (6.65 fps)
30	-	161.03 ms (6.21 fps)	153.85 ms (6.50 fps)

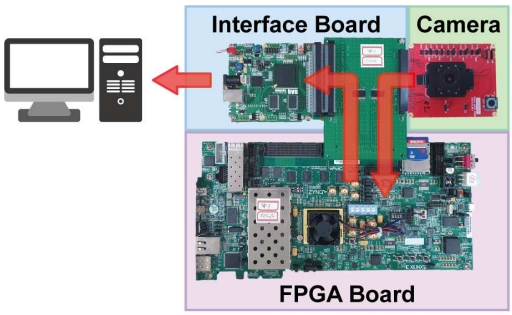

Fig. 4. Test configuration of proposed architecture

Fig. 5. Face detection result

Acknowledgment

This research was supported by Multi-Ministry Collaborative R&D program (R&D program for complex cognitive technology) through the National Research Foundation of Korea (NRF) funded by MOTIE (2018M3E3A1057248).

References

[1] I. Aydin and N. A. Othman, "A new IoT combined face detection of people by using computer vision for security application," in *Int. Artif. Intell. Data Process. Symp.*, Sept. 2017, pp. 1–6.

[2] H. Peng, "Application research on face detection technology based on OpenCV in mobile augmented reality," *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 8, no. 4, pp. 249–256, Apr. 2015.

[3] I. L. K. Beli and C. Guo, "Enhancing face identification using local binary patterns and k-nearest neighbors," *J. Imaging*, vol. 3, no. 3, pp. 1–12, Sept. 2017.

[4] W-C. Hu, C-Y. Yang, D-Y. Huang, and C-H. Huang, "Feature-based face detection against skin-color like backgrounds with varying illumination," *J. Inf. Hiding Multimedia Signal Process.*, vol. 2, no. 2, pp. 123–132, Apr. 2011.

[5] I-O. Stathopoulou and G. A. Tsihrintzis, "Appearance-based face detection with artificial neural networks," *Intell. Decis. Technol.*, vol. 5, no. 2, pp. 101–111. Feb. 2011.

[6] A. Z. Kouzani, F. He, and K. Sammut, "Commonsense knowledge-based face detection," in *IEEE Int. Conf. Intell. Eng. Syst.*, Sept. 1997, pp. 215–220.

[7] Z. Jin, Z. Lou, J. Yang, and Q. Sun, "Face detection using template matching and skin-color information," *Neurocomputing*, vol. 70, no. 4-6, pp. 794–800, Jan. 2007.

[8] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.

[9] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, "FPGA-based face detection system using Haar classifiers," in *ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2009, pp. 103–112.

[10] D. Kim, J. Hyun, and B. Moon, "Memory-efficient architecture for contrast enhancement and integral image computation," in *Int. Conf. Electron. Inf. Commun.*, Apr. 2020, pp. 1–4.

[11] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, 2008.