Hecto-Scale Frame Rate Face Detection System for SVGA Source on FPGA Board

Zheng Ding †1, Feng Zhao ‡2, Tinghui Wang ‡3, Wei Shu §4, Min-You Wu †5

† *Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China*
{1 dingzheng, 5 mwu}@sjtu.edu.cn

‡ *Digilent Electronic Technology Co. Ltd., Shanghai, China*
{2 frank.zhao,3 steven.wang}@digilentinc.com

§ *Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, USA*
4 *shu@ece.unm.edu*

***Abstract*—This paper proposes techniques for face detection and gives the implementation details for an FPGA development board. We analyze and discuss the relation between the system computation cost and selection of the image scaling factor. We give a new method to select the stop threshold for the image reduction process, which reduces the total computation by half. We also provide a color image output mode to let our system enjoy more human-oriented design. Test results show that the system achieves real-time face detection speed ($100fps$) and a high face detection rate (87.2%) for an SVGA (600×800) video source. The low power consumption ($3.5W$) is another advantage over previous work.**

***Keywords*-Face Detection System; Field Programmable Gate Array; Haar Feature;**

I. Introduction

Face detection is concerned with finding whether there are any faces in a given image or not, and if there are, returning the image locations and contents of all faces. Usually grayscale images are used; if the input is not grayscale, a pre-transform will be performed. Approaches to face detection can be divided into two categories: feature-based approaches and image-based approaches.

Systems using general purpose processors or graphics processing units (GPUs) have reached a very high detection rate for any image condition (such as low light intensity, profile, etc.). However, typical software implementations[1][2] give a detection speed lower than $15fps$ (frames per second), far less than realtime applications require. A realtime face detection system which can be directly attached to the video source (such as an industrial camera) is urgently needed.

There has been some work on implementing face detection using FPGAs. Junguk Cho [3] implemented a full FPGA-based face detection system, but the best performance was only $28fps$ at 320×240 resolution and $7.51fps$ at 640×480, well short of the realtime requirement ($> 60fps$). McCready's work [4] reaches a speed of $30fps$ with 320×240 input, but it uses a Transmogrifier-2 configurable hardware system which includes nine FPGA chips; this definitely cannot be used in a portable device. Paschalakis [5] presents an FPGA-based face detection and tracking system for audiovisual communications, with a particular focus on mobile video conferencing. However, the paper does not quantify the test cases and we cannot determine the exact performance of the design. Yu[6] implements the AdaBoost algorithm on an FPGA, focusing on how to implement an efficient face classification stage. Based on synthesis results, Yu's design can operate at 91 MHz, equivalent to 15 video frames (120×120) per second. Gao[7] gets a $1.05fps$ detection speed, but only the Haar feature classifiers are in the FPGA while the other parts of the system are still in a normal Intel CPU. The data transfer between the FPGA and the CPU through the PCIe interface.

Compared with previous work, our major contributions are: (1).Design and implement a feasible realtime ($100fps$) face detection system on a general-purpose FPGA development board to deal with a SVGA video stream. (2).Propose a new image reduction process stop threshold. (3).Give a theoretical derivation of the relation between image scaling factor and the computation cost, and a method of selecting the best scaling factor.

The rest of this paper is organized as follows: Section II provides an overview of the face detection system. Section III gives implementation details about the face detection circuit. Section IV gives a experimental results and resource utilization, and Section V concludes the paper.

II. Face Detection System Design

In this section, we give an overview of our face detection system. Also, the image reduction stop threshold and scaling factor selection method are analyzed.

A. System Overview

The basic operating mechanism of our face detection system is: Let a sub-window slide on the original image from the upper left corner to the lower right corner following column major order with one pixel steps. The system will calculate the integral image of the sub-window and use the Haar feature classifier to detect whether the sub-window contains a face or not. If the whole image has been examined and no face has been detected, the image will be reduced by a constant scaling factor (f) and the sub-window slide

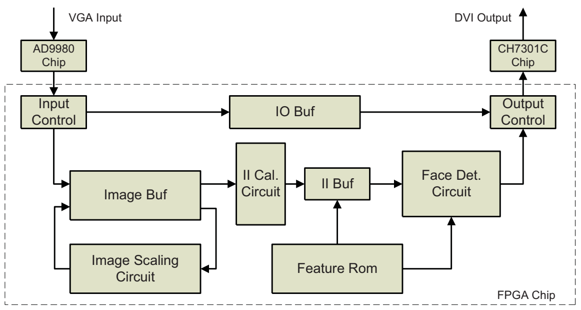

Figure 1. Face Detection System Overview

process redone until a face is detected or the image is smaller than a threshold. Figure 1 gives an overview of our system.

B. Image Reduction Stop Threshold and Factor Selection

In our previous face detection system[8] using a Virtex-II Pro board, when the system was tested against the Shanghai IsVision Sample Database which includes 5000 positive images and 11000 negative images, we got two observations. The first was that 89% of the faces had been detected before scaling down to smaller than 70% of the initial source image size. Table I gives the details and the scaling factor used is 1.2. The second was that if a face is detected, almost all other faces in the same image will be detected at the same scaling down level. The intuitive explanation is that all faces in the same image are about the same size so they are detected at about the same level (Statistical data show that: $P(0.8 \le \frac{ls}{ll} \le 1) = 96.4\%$, $P(0.8 \le \frac{ws}{wl} \le 1) = 97.2\%$. The P means probability, and ls, ws (or ll, wl) are the length and width of the smallest (or largest) face in an image. The smallest and largest faces in all images are 42×52 and 513×547 respectively after the images are normalized to 600×800 size). Based on these two observations, we can save time for the the integral image and Haar feature calculation by optimizing the image reduction stop threshold and factor selection.

The basic detection process in previous work is similar to our design. The difference is that in those designs the image reduction process stops only when the reduced image is smaller than a preset threshold (usually the size of the sub-window). In our design, the process stops either when a face has been detected or the image is smaller than the sub-window. As described in II-A, after every image scaling, the detection sub-window moves from the top left to the bottom right with a one column (the horizontal coordinate plus one) step. As each sub-window detection process attempts to detect the features, the amount of traveling the sub-window done on the image will determine the computation cost of the whole system. From the above description, we can infer that the computation load of the system is determined by the image reduction factor (f).

We assume the initial image is $P_h \times P_v$ (assume $P_h \le P_v$)

Table I
DETECTION RATE FOR DIFFERENT IMAGE SCALING LEVELS

Scale Times	Scaled Size	D. Num.	DR	Accum. DR
0	1	5982	0.374	0.374
1	0.833	6349	0.397	0.770
2	0.694	1987	0.124	0.894
more than 2	≤ 0.579	303	0.019	0.913

size, the sub-window size is $S \times S$ and the image scaling factor is f ($1 < f \le \frac{P_h}{S}$). If no face is detected in the initial image, the image is scaled down by f until a face is detected or the image becomes smaller than the sub-window size. So the scaling process is performed $log_f \frac{P_h}{S}$ times. After the ith ($0 \le i \le log_f \frac{P_h}{S}$) iteration, the sub-window needs $(P_h \cdot (\frac{1}{f})^i - S + 1) \cdot (P_v \cdot (\frac{1}{f})^i - S + 1)$ steps to traverse the image, that is:

$$T_i = P_h \cdot P_v \cdot (\frac{1}{f^2})^i - (S-1) \cdot (P_h + P_v) \cdot (\frac{1}{f})^i + (S-1)^2 \quad (1)$$

We suppose that the face can be detected after the initial image been scaled down to $\sigma_h \times \sigma_v$ size. The total steps which the sub-window needs to traverse are:

$$T = \sum_{0 \le i \le I} T_i \quad (2)$$

where $I = log_f \frac{P_h}{\sigma_h}$. We let $w = \frac{\sigma_h}{P_H}$ ($0 < w \le 1$) and can get T for our technique:

$$T = P_h \cdot P_v \cdot \frac{f^2 - w^2}{f^2 - 1} - (S-1) \cdot (P_h + P_v) \cdot \frac{f - w}{f - 1} - (S-1)^2 \cdot log_f w \quad (3)$$

In the same way, the steps needed for previous work are:

$$T_p = P_h \cdot P_v \cdot \frac{f^2 - (\frac{S}{P_H})^2}{f^2 - 1} - (S - 1) \cdot (P_h + P_v) \cdot \frac{f - \frac{S}{P_H}}{f - 1} - (S - 1)^2 \cdot log_f \frac{S}{P_H} \quad (4)$$

We know that T_p is related to the scaling factor f. In this implementation, our input image size is 600×800 and sub-window size is 24×24 so $T_h = 600$, $T_v = 800$ and $S = 24$. In figure 2, the subgraph (a) gives the number of steps needed in our proposal, the subgraph (b) gives the number of steps needed in previous work and the subgraph (c) shows the speedup of our proposal relative to previous work. In these three subgraphs, the variables are w and f. Our design is 50 times faster in the best case. For clarity, we give the variation diagram of T_p and T under $w = 0.83$ condition in subgraph (d) and (e). We also give the normalized speedup in subgraph (f) for different scaling factors (f), based on empirical values from our previous work. The normalization ratios of different w value are deduced from our previous experimental data shown in table I. We also found that the scaling factor should not be smaller than 1.1 since with small factors the amount of calculation increases dramatically.

There is a trade off between the amount of calculation and the detection rate; in our design, we choose a scaling factor of 1.2.

III. Face Detection Circuit Implementation

The face detection circuit is the most important component in our system. The basic function of this component is to use the Haar feature based cascade classifier to detect whether the sub-window includes a face or not. Figure 3 gives the details of our implementation. Since in our design, the integral image is generated in pipeline mode, we can carry out the detection work for more than one sub-window concurrently to accelerate the detection process. In figure 3, note that the two pieces of detection logic with dotted line frames are exactly the same. The reason we draw it into two parallel parts is to emphasize the concurrent pipeline detection process. Of course, if we have larger FPGA chips in the future, we can also implement it in two parts to get the parallelization speedup. When the first sub-window integral image data is finished the first stage detection and the second sub-window integral image data will feed to the detection pipeline and do the first stage detection work.

The data transferred between the feature ROM and detection logic include weight (W), feature threshold (FT) and stage threshold (ST) data. When the detection process begins, it will send the rectangular coordinate information of the first feature in the first stage to the integral image buffer. When the integral image buffer receives the point coordinate data, it will send the integral image value to the detection logic which will carry out the Haar feature calculation process. The Haar feature value will be multiplied by the feature weight and compared with the feature threshold. If it is larger than the threshold, the value will be added to the stage feature value buffer; otherwise the feature value is ignored. After all features are examined and the stage feature value accumulated, that value is compared with the stage threshold. If it is larger than the threshold, it means the sub-window may contain a face and will enable the second stage feature detection process. If the value is smaller, it means that there is no face in the sub-window and the Haar feature detection process for this sub-window is finished. If a sub-window passes all 25 detection stages, it will be tagged as having a face in it.

IV. Resource Usage and Performance Evaluation

The resource usage of different parts in our face detection system is shown in table II. The image buffer (Image Buf.) and integral image buffer (II Buf.) consume 10338 and 22168 slice registers respectively. This register usage accelerates the data access time since the whole or partial Image Buf. or II Buf. data need to be accessible simultaneously. The slice LUTs usage of a component can show its corresponding logical complexity. Since the image

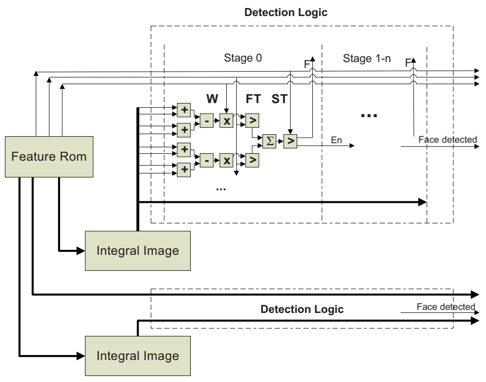

Figure 3. Face Detection Circuit

Table II
FACE DETECTION SYSTEM RESOURCE USAGE

Component	Slices Reg.	Slice LUTs	BRAM	DSP48Es
Input Control	567	384	0	0
Output Control	2063	1682	0	2
Image Buf.	10338	2086	16	0
Image Reduce Circuit	4903	10068	0	0
II Cal. Circuit	10917	8439	0	2
II Buf.	22168	2746	18	0
Feature Rom	249	376	33	2
Face Det. Circuit	1964	4982	0	6
IO Buf.	604	760	20	0
Whole System	56107	33053	87	12
AVAILABLE	69120	69120	148	64

reduction algorithm enjoys a high degree of parallelism, we have implemented 10 identical image reduction circuits to speed up the image scaling process; this uses 10068 LUTs. The feature ROM including all the 2913 weak classifiers is constructed with BRAM. The capacity of each BRAM is $36Kb$. In the output control module, the DSP48Es are used to quicken the detected face coordination calculation process by the coordination of the reduced image and the reduction count number. In summary, the system takes up 81.1%, 47.8%, 58.7% and 18.7% of on chip registers, LUTs, BRAM and DSP48Es respectively.

Since Hiromoto's work[9] has a lot of common ground with ours, we give the comparison in table III. The general detection rate between the two is roughly equal. We win in the input image resolution, output image mode, detection rate, resource usage and price aspects. The low cost with high performance character of our design gives us a good chance to enter the practical industrial market. The color image output also gives us a strong advantage compared to grayscale in real applications. One of the most important contributions in [9] is the relation between classifier bit and false positive rate, and in that area, his system outperforms ours though ours outperforms many others.

A state-of-the-art GPU-based (GPUs and FPGAs are widely considered to be the most promising solutions to the next generation HPC) face detection solution is [2],

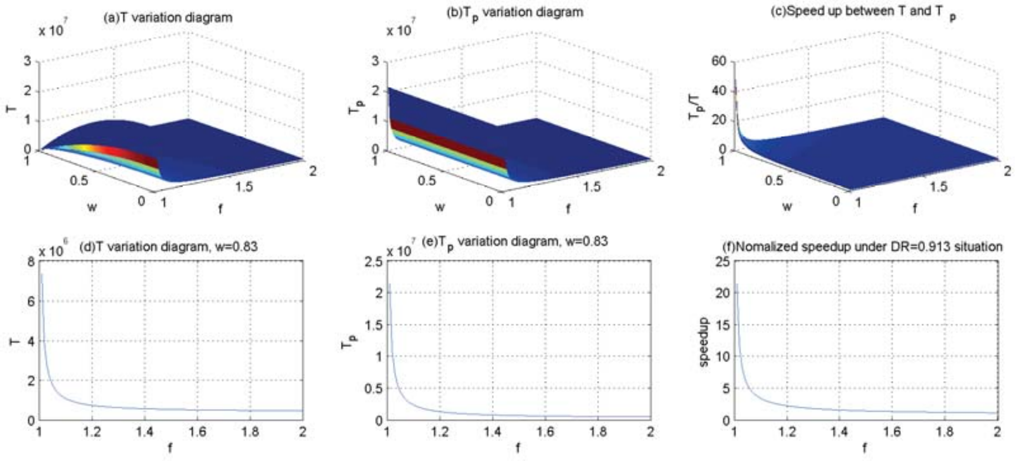

Figure 2. The Relation Among T, T_p, w and f

Table III
FACE DETECTION SYSTEM PERFORMANCE SUMMARY

	Our Work	Hiromoto [9]
Platform	XUPV5LX110T	N.A.
FPGA Chip	XC5VLX110T	XC5VLX330
FPGA Chip Price	\$1,613	\$9,313
Clock Frequency	150MHz	140MHz - 160MHz
Input Image Resolution	600×800	640×480
Output Image Mode	color image	grayscale
Sub-window Size	24×24	24×24
Detection Rate	87.2%	$\approx 90\%$
False Positive Rate	1.7%	$\approx 0.002\%$
Detection Speed	$100fps$	$30fps$
Registers	56107	63443
LUTs	33053	55515

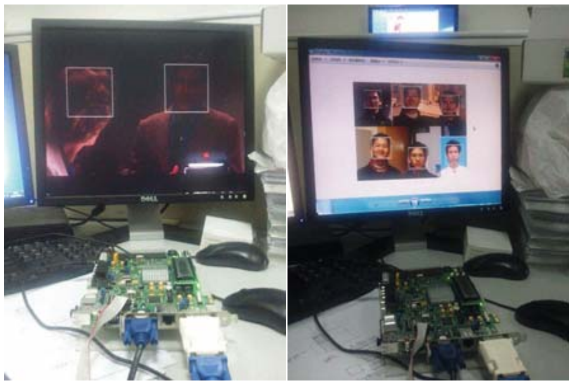

Figure 4. Face Detection Result

using four Geforce GT 220 GPUs. Our design gives better performance ($100fps$ vs. $15.2fps$) at lower cost (\$1613 vs. \$($500 \times 4 = 2000$)) using less power ($3.5W$[10] vs. $200W$[11]). We cannot compare the detection rate and false positive rate since these are not given in [2]. The run results of our face detection system are shown in figure 4.

V. CONCLUSION

In this research, we have designed and implemented an FPGA-based face detection system. Through the design of a new image reduction stop threshold and the optimization of the image reduction factor selection, our system achieves real-time $100fps$ face detection on SVGA (600×800) video. On other performance indicators, we also outperform previous work. In our implementation, we also introduce the color image output mode, which increases the competitiveness of our design when entering the industrial market.

ACKNOWLEDGMENT

This work is supported by Shanghai Jiao Tong University Innovation Fund For Postgraduates.

REFERENCES

[1] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.
[2] D. Hefenbrock, J. Oberg, T. Nhat, R. Kastner, and S. B. Baden, "Accelerating viola-jones face detection to fpga-level using gpus," in *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines,*, 2010, pp. 11–18.
[3] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, "Fpga-based face detection system using haar classifiers," in *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*. 2009, pp. 103–112.
[4] R. McCready, "Real-time face detection on a configurable hardware system," in *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*. Springer Berlin / Heidelberg, 2000, vol. 1896, pp. 157–162.
[5] S. Paschalakis and M. Bober, "A low cost fpga system for high speed face detection and tracking," in *2003 IEEE International Conference on Field-Programmable Technology (FPT), 2003. Proceedings.*, 2003, pp. 214–221.
[6] W. Yu, B. Xiong, and C. Chareonsak, "Fpga implementation of adaboost algorithm for detection of face biometrics," in *IEEE International Workshop on Biomedical Circuits and Systems,*, 2004, pp. S1/6–17–20.
[7] G. Changjian and L. Shih-Lien, "Novel fpga based haar classifier face detection algorithm acceleration," in *International Conference on Field Programmable Logic and Applications, 2008. FPL 2008.*, 2008, pp. 373–378.
[8] T. Wang, F. Zhao, J. Wan, and Y. Zhu, "A novel hardware architecture for rapid object detection based on adaboost algorithm," in *Advances in Visual Computing*. Springer Berlin / Heidelberg, 2010, vol. 6455, pp. 397–406.
[9] M. Hiromoto, H. Sugano, and R. Miyamoto, "Partially parallel architecture for adaboost-based detection with haar-like features," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 19, no. 1, pp. 41–52, 2009.
[10] Xilinx, "Xpower estimator," 2010. [Online]. Available: http://www.xilinx.com/ise/power_tools/license_virtex5.htm
[11] Nvidia, "Nvidia products," 2009. [Online]. Available: http://www.nvidia.com/page/products.html