Fekih, H. B., Elhossini, A., & Juurlink, B.

An Efficient and Flexible FPGA Implementation of a Face Detection System.

An Efficient and Flexible FPGA Implementation of a Face Detection System

Hichem Ben Fekih, Ahmed Elhossini, and Ben Juurlink

Technical University of Berlin, Embedded Systems Architectures (AE)
Einsteinufer 17 , D-10587, Berlin, Germany,
hichem.f@live.de, {ahmed.elhossini,b.juurlink}@tu-berlin.de

Abstract. This paper proposes a hardware architecture based on the object detection system of Viola and Jones using Haar-like features. The proposed design is able to discover faces in real-time with high accuracy. Speed-up is achieved by exploiting the parallelism in the design, where multiple classifier cores can be added. To maintain a flexible design, classifier cores can be assigned to different images. Moreover using different training data, every core is able to detect a different object type. As development platform, the Zynq-7000 SoC from Xilinx is used, which features an ARM Cortex-A9 dual-core CPU and a programmable logic (FPGA). The current implementation focuses on the face detection and achieves a real-time detection at the rate of 16.53 FPS on image resolution of 640x480 pixels, which represents a speed-up of 6.46 times compared to the equivalent OpenCV software solution.

Keywords: Face detection, Computer Vision, Zynq, FPGA

1 Introduction

Several applications in different domains require a reliable fast detection system, where the location and scale of faces in the image are extracted. Face detection is widely used as a first stage for applications, such as face recognition, video surveillance, eyes detection to measure the driver drowsiness in modern cars or in advertisement industry to collect information like gender and age range for targeted advertisements. Face detection is a fundamental technique that enables a natural human-computer interaction (HCI). There were several approaches based on different feature sets and methods that have dealt with face detection. Many of them were too complex and time consuming, since the difficulty associated with the face detection can be attributed to various factors, such as variation in scale, location, pose, lighting condition, etc.

On 2001 Viola and Jones [14, 15] have proposed a new face detection framework based on Haar features that is able to process images rapidly. Since its publication, it has received considerable attention, because it can achieve very high detection rate. An open-source implementation of the detector made by Rainer Lienhart [9] exists already in the OpenCV library [1], which includes ready training data. Viola and Jones approach is primarily developed for face detection, but the algorithm is able to detect any object by using different training data.

This paper focuses on the face detection task based on Viola and Jones algorithm. A new software-hardware co-design approach is investigated,

which enhances the performance of the original approach and achieves real time face detection performance. A hardware accelerator is developed to perform most of the computationally intensive tasks and implements all necessary components participating in the face detection. To maintain a flexible design, this paper presents the architecture of an Evaluator core, which can be duplicated to improve the performance. Adding more cores will increase the frame-rate on the cost of more resources and possibly power consumption. The Evaluator core contains a Read Only Memory (ROM), which can be adapted to detect any object by using the respective training data. Moreover, the design can be accessed as a network service throuth the network so multiple sources and multiple face detection systems can be integrated togather to increase the overall throughput of the system and flexibility of the design.
The remainder of this paper is organized as follows. Section 2 covers the relevant background information needed in order to perform the work presented in this paper. Section 3 lists the related work that focuses on software as well as hardware solutions for the face detection task. The actual high-level design and implementation are presented in Section 4, which includes a discussion about crucial design decisions and some used techniques in order to accelerate the process. Section 5 presents the evaluation results of the system and performs a comparison with equivalent software and hardware implementations. Finally, Section 6 includes the conclusions of the paper.

2 Background

On 2010, Nikolay Degtyarev et al. [4] have presented some research comparing many different face detection algorithms based on the false face rejection rate, false acceptance rate and speed. It turned out that the extended realization of the Viola and Jones object detection algorithm [9] is the best open source available algorithm based on the performance and the detection rates. It is considered the first real-time object detection framework providing very good detection rates. A complete software realization of the algorithm already exists in the OpenCV library [1]. Basically the face detection algorithm tries to locate specific Haar features of human faces and consists of two phases, training and detection. This paper will only cover the implementation of the detection phase on the FPGA, since the training data can be generated off-line on a typical PC. The training data used for the detection are created by Rainer Lienhart and are located in the OpenCV library [1].

2.1 Cascade Architecture

Viola and Jones algorithm consists of many cascaded stages, which are constructed during the training phase. A stage represents a strong classifier, which gets a sub-window as input and gives an output value indicating if the current window is a face candidate. Viola and Jones have proved in their work that only one stage containing 200 Haar features is able to detect faces successfully, but it will consumes a lot of time to process the complete image. Thus, stages are arranged in a cascaded form, where the first stages are less complex than the ones at the end of the cascade. In the case that any stage returns a negative result, the current sub-window will be immediately rejected otherwise next stage will

be activated. If the sub-window passes all stages then it will be considered as a face candidate. This process is illustrated in Figure 1. Because the number of non-face candidates in an image is much more than face candidates, the first few stages are constructed to reject most of negative examples before more complex stages are activated. The performance is increased by rejecting negative sub-windows as early as possible. For this reason the first few stages are designed to contain only a few number of features.

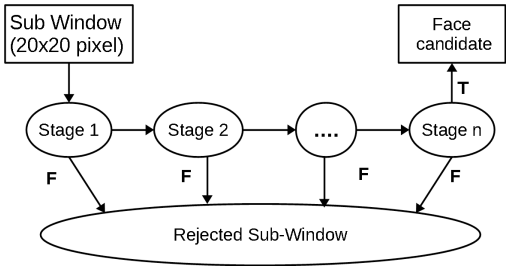


Fig. 1: Schematic depiction of the detection cascade

A single stage (also called a strong classifier), is responsible for classifying a sub-window as a face or a non face candidate. In this paper, the stage consists of many decision trees. Each tree evaluates the image and return an output value (h). These "h" values are summed up and compared to the stage threshold (θ) as shown in Equation 1.

$$H(\theta) = \begin{cases} 1 & if \sum h_i \geq \theta \\ 0 & otherwise \end{cases} \quad (1)$$

In Equation 1, H is the strong classifier function, an output value 1 means that the current sub-window may contain a face and should be further analysed. The output value 0 means that the current sub-window is a negative image and has to be immediately rejected.

3 Related Work

There exist several techniques to detect faces in still images. Some attempts are based on detecting faces by skin color [10, 13]. Such methods have the disadvantage, that they do not work with all types of skin color and are very vulnerable to different lighting conditions.

On 2007, S. Liao et al. have presented a face detection system based on Viola and Jones algorithm [8]. Instead of using Haar features, they have used Local Binary Patterns values (LBP), which are showing less accuracy compared to Haar features. Since LBP features are integer, the detection and training phase are much faster than with Haar features. Another attempt made by X. Zhi et al. [20] presents a unified model for face detection and pose estimation. An experimental result over faces larger than 150 px shows a good detection rate but no information are mentioned about the performance, as well as, the detection rate for faces smaller than 150 px.

The work presented in this paper is based on the work made by R. Lienhart et al. [9], which presents an extended Haar feature set for the face detection system of Viola and Jones. The work of R. Lienhart et

al. shows better detection rates than the algorithms presented in [10, 13, 8]. The trained classifiers are able to detect faces larger than 24 px. The software solution detects frontal faces in images (320x240) at 5 FPS on a Pentium-4 2 Ghz using a rescaling factor of 1.2. Since Viola and Jones approach is the most used and extended algorithm. A lot of work has been done in attempts to accelerate it. Depending on the host platform, software solutions that use optimized OpenCV implementations can reach 3 FPS on images consisting of 640x480 pixels. In order to accelerate the calculation of the Haar features, many researches have been dealt with hardware approaches.

In the work of D. Hefenbrock [5] a GPU approach is proposed, which is programmed using CUDA [12] and tested on NVIDIAs Tesla processors. The final GPU implementation reached 15.2 FPS and is running on a desktop server containing 4 Tesla GPUs. The reported performance is considered as sufficient and can be used in real-time applications. However using 4 GPU processors will consume certainly much power and typical quad-Tesla desktop supercomputers are expensive.

Another work made by Hung-Chih Lai et al. [7] describes a face detection implementation on FPGA, where the classification process is done in only one clock cycle and the integral image is stored in registers. Such approaches can reach an enormous high speed detection but the system is only using 52 Haar features for the classification. Each one has a direct connection to the integral image. Using such a little number of features may affect the accuracy of the detection. Unfortunately, no information are mentioned about the accuracy of the system. Their hardware design may become infeasible when the number of features increases, because the number of classifiers will increase, which have a direct connection to the size of the integral image registers.

Cho et al. [3, 2] have proposed an FPGA design, where images are stored in BRAMs and the integral image is stored in registers. Their design supports multiple classifiers, which enables processing many Haar features in parallel. The reported performance reached 16.08 FPS using 8 classifiers, which is the best reported frame rate with good accuracy. Nevertheless, the work presented in [3] is showing a decreased frame rate when the number of faces in the image is growing. Their design consumes relatively much FPGA resources and can not be adopted for lower end FPGAs.

A lot of promising work has been presented in the literature for face detection. The majority of them meets the real-time constraints. However, this paper will presents a complete new hardware solution, where the processing time is reduced and a flexible design is provided, which minimizes the reserved hardware resources and enables low end FPGAs to be used to extend their functionality.

4 Design and Implementation

This section discusses some design decisions and presents an FPGA implementation of the described face detection system. The first sub-section introduces the complete detection system. The remaining sub-sections will deal in details with the design of all required modules.

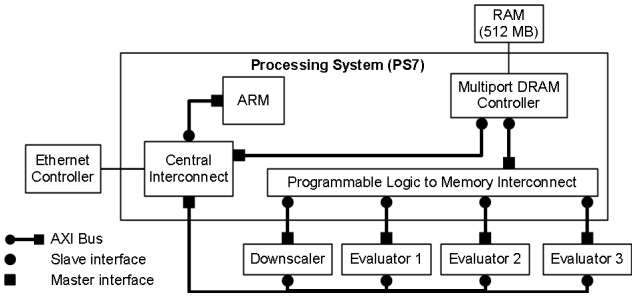


Fig. 2: Complete System Overview

4.1 System Overview

In the proposed system, images are captured on the PC and transferred via Ethernet to the system using the UDP protocol. An application running on the PC developed using C++ is responsible for sending images and presenting the results. The ARM-CPU on the Zynq Chip handles receiving the pixel data and ensures storing them in an external memory (DDRAM). The RAM acts as shared memory between all components. The Processing System (PS7) [17] includes four high performance AXI [16] slave ports, which allow accessing the shared memory from four different modules using an AXI master interface. The Evaluator classifies a 20x20 pixel image as a face or non face candidate. It implements the task of the cascade architecture presented in Section 2. In Figure 2 three instances of the "Evaluator" are included to improve the performance of the face detection system. Each one will be assigned to a part of the image and thus the processing time will be reduced. On the other hand, a module is required to accelerate the image down-scaling, which is called Downscaler and it includes an AXI master port to read/write image pixels directly from/to the memory. The current version of the module resizes the image using the nearest neighbor interpolation algorithm and a scale factor of 1.2. The Downscaler and the 3 instances of the Evaluator are connected to the ARM CPU via AXI Bus. The CPU takes the mission to command and synchronize the modules. It also assigns image regions to the Evaluator instances for classification. More Evaluator cores can be connected to the AXI bus system to increase the performance if required.

4.2 The Evaluator Core

The Evaluator is the main module of the proposed architecture. It is responsible for classifying a 20x20 sub-window and it implements the cascaded strong classifiers described in Section 2. The Evaluator module is composed of several components. Figure 3 shows an overview of the Evaluator structure. The Preprocessing Engine module reads the pixels from memory and computes the integral image. Then, the Evaluator core starts reading the training data from the ROM module and it selects the required pixels for the classification from the integral image buffer. The tree receives the node threshold values, three node inputs and the rectangle data required for the computation of the Haar features. Then, the tree selects a value from the inputs, which represents the output vote

indicating if the current window contains a face or not. Next sections deal in detail with the individual sub-modules of the Evaluator.

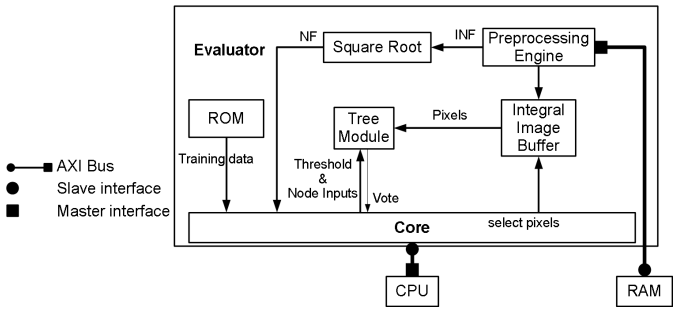


Fig. 3: Evaluator Overview

The ROM module consists of Block RAMs and contains the training data needed for the face detection. Overall, the training data includes 20 stages, 1047 trees, 2094 nodes (Haar features) and 4535 rectangles. These data are converted to a compressed binary format and are accessed by the evaluator core at each stage of the processing.

Preprocessing Engine To perform the classification, 4535 rectangles have to be read from the ROM. Each rectangle needs four memory accesses to be computed. Hence, it is essential to copy a part of the original image to an internal cache to reduce the memory accesses and increase the performance. The Preprocessing Engine module uses the AXI Master Burst component [16] from Xilinx to communicate with the memory and to copy 24x22 window to the internal cache.

To minimize the effect of different lighting conditions when detecting faces, the implemented Evaluator uses training data, that have been generated using variance normalized images. It is therefore necessary during the detection to normalize images before processing them. The normalization is simply performed by multiplying the node thresholds with a normalization factor (NF), which is computed using Equation 2 and 3.

$$NF = \sqrt{INF} \qquad (2) \qquad INF = N.\sum i_s^2 - \sum i_s \qquad (3)$$

In Equations 2 and 3, N is the total number of pixels, i_s is the pixel value within the sub-window and INF is the intermediate normalization factor. A square root module is required to compute the final equation. While reading the pixel data, the integral image and the INF values are computed.

Integral Image Buffer The integral image facilitates the Haar feature calculation. The face detection system processes one tree every cycle (four rectangles in total) to ensure real-time performance. Since each rectangle requires four accesses, we need to access the integral image 16 times simultaneously (four rectangles times four accesses). This component consists mainly of dual-port block RAMs, each one permits maximal two simultaneous accesses and consequently 8 instances are required, that contain the same data. Figure 5 shows the structure of the Integral Image Buffer module. It contains two buffers, each one consists of 8 BRAMs.

Using the Buffer Select signal one buffer can be chosen for read accesses and the other one for write operations. Data can be read and written in the same time. This feature approximately doubles the final frame rate.

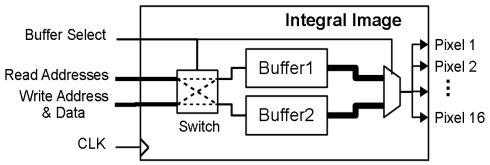


Fig. 5: Block Diagram of the Integral Image Buffer

Decision Tree The tree contains two nodes, as shown in Figure 6, which predict an output value starting from two Haar features. Each node receives the Haar feature data from the integral image buffer and then it computes the feature value and compares it to the threshold received from the ROM module. Depending on the result a vote is selected.
Figure 6 shows the internal architecture of the proposed node design. It receives as inputs 3 rectangle definitions (a, b, c and d), weight of the second rectangle, a threshold value and a polarity (p).

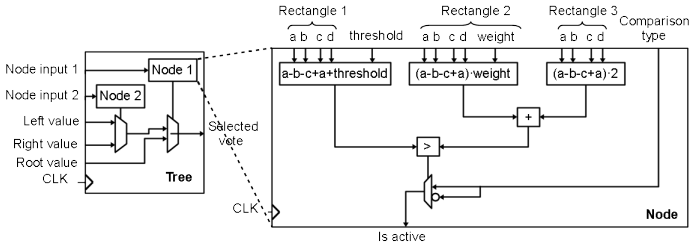


Fig. 6: Tree Architecture

4.3 Summary

The system presented in the prvious sections receives the image data from the video source using the network interface. The software running on the ARM core is responsible for receiving the image through the network interface, storing it in the memory, and transmitting sub-windows to the evaluator core(s) for evaluation. It also controls the scaling module to scale sub-windows before processing them by the evaluator. Finally, the software transferes the evaluation results (locations of faces) back to the image source. This configuration allows the system to receive images from different sources and also multiple versions of the system can be used in parallel to process images from the same source. Higher resolutions as well as higher framerates can be achieved using this approach. Changing the training data stored in the ROM module allows the use of the same architecture to detect other types of objects such as animals, humans, cars and others.

5 Evaluation and Results

Several experiments are presented in this section to evaluate the performance and accuracy of the proposed architecture. In our evaluations we use two development boards equipped with two different chips of Xilinx Zynq-7000 All Programmable SoCs. The two boards are The ZedBoard [19] and Xilinx ZC706 Evaluation Kit [17]. Individual components were

modelled in VHDL and validated using RTL simulation and by integration with the ARM Based SoC using Xilinx EDK 14.7 [18].

5.1 Resource Utilization

Xilinx EDK [18] platform is used to synthesis the proposed architecture. Based on the synthesis result, the FPGA can operate at a maximum clock speed of 144.32 MHz for the Z706 board and 100MHz for the ZedBoard. A short analysis showed that the critical path is located in the AXI Master Burst interface, which is used to accelerate reading image pixels from the memory. Table 1 shows the resources utilized by the face detection system on the ZC706 and ZedBoard when it includes 1, 2 or 3 Evaluator cores (running in parallel). The most consumed resources by the design are the BRAMs, which contain the training data and the integral image. The required resources increase linearly with the number of the used Evaluator cores. Depending on the available FPGA resources, the number of cores can be chosen. From these results, it is clear that the proposed design is portable and can be implemented on different FGPA architecture.

Table 1: Device Utilization Characteristics for the Face Detection System

Board Name	ZedBoard						ZC706	
Target Device	xc7z020-1-clg484						xc7z045-ffg900	
Number of Evaluator Cores	1		2		3		3	
Number of Slice Registers	4113	3%	7011	6%	9908	9%	9908	9%
Number of Slice LUTs	4596	8%	7852	14%	11004	20%	11047	5%
Number of used BRAMs	28	20%	56	40%	84	60%	84	15%
Number of DSP48E1s	11	4%	22	8%	33	15%	33	3%

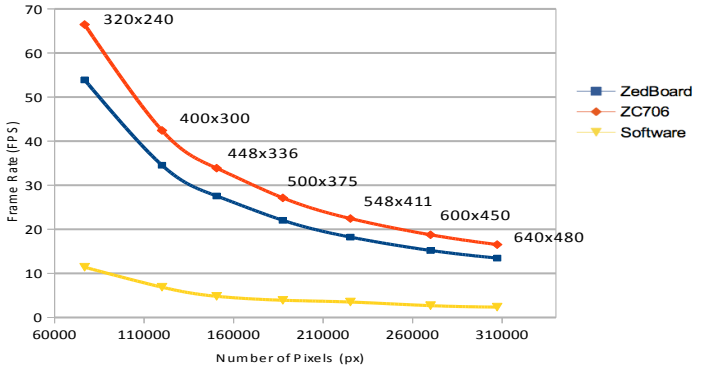


Fig. 7: Performance Measurement of the Face Detector Dystems using Different Image Sizes

5.2 Performance Comparison

The proposed face detector is based on the OpenCV implementation of the algorithm[9] and uses the same training file. In this section we introduce a comparison between the proposed implementation and the software implementation using OpenCV. The performance of the OpenCV implementation is determined by measuring the computation time required for analysing images on a PC having an Intel Core i5-4200U CPU (2.30 GHz), 8 GB DDR3L RAM (1600 MHz), Microsoft Windows 8.1. All of the software programs are developed using MinGW [11]. To make the

Table 2: Performance Comparison of the Proposed Detector and the Original Implementation

(a) The ZedBoard

Number of Faces		1	8	21
Software Detector		3.03 FPS (1.00x)	2.75 FPS (1.00x)	2.56 FPS (1.00x)
Hardware	1 core	6.19 FPS (2.04x)	6.17 FPS (2.24x)	6.14 FPS (2.40x)
	2 core	10.54 FPS (3.47x)	10.51 FPS (3.82x)	10.46 FPS (4.09x)
	3 core	13.48 FPS (4.45x)	13.47 FPS (4.90x)	13.43 FPS (5.25x)

(b) The ZC706 Evaluation Board

Number of Faces		1	8	21
Software Detector		3.03 FPS (1.00x)	2.75 FPS (1.00x)	2.56 FPS (1.00x)
Hardware	1 core	7.69 FPS (2.54x)	7.68 FPS (2.79x)	7.64 FPS (2.98x)
	2 core	13.08 FPS (4.32x)	13.05 FPS (4.75x)	13.00 FPS (5.08x)
	3 core	16.58 FPS (5.47x)	16.57 FPS (6.03x)	16.53 FPS (6.46x)

comparison possible both implementations of the hardware and software use exactly the same parameters. To study the influence of the image size over the performance, an experiment was conducted, which consists of measuring the frame rate of the face detector using different image sizes. The result is depicted in Figure 7. It shows a decreasing frame rate when the size of the image is increased. Higher resoultions are supported as well, with the cost of more processing time.

Since the classifier spends more time on face sub-windows, images containing a lot of faces will require more time to process and hence will reduce the frame rate. Thus, the second measurement covers a performance comparison by processing 640x480 images containing different number of faces and using variant number of Evaluator cores. The results of various configurations and number of faces are shown in Tables 2a and 2b for the ZedBoard and ZC706 evaluation board respectivly.

On the ZedBoard the 1 core face detection system is capable of processing images at an average speed of 6.19 FPS. The 2 cores face detection system is capable of processing images at an average speed of 10.54 FPS, which represents a performance improvement of 1.70 times over the 1 core version. The 3 core face detection system has the best speed results and it is able to process the image at an average speed of 13.48 FPS. The ZC706 evaluation board shows better performance compared to the ZedBoard, since it has better speed grade and the programmable logic is operating at higher frequency. The highest frame rate of an average of 16.5 FPS is achieved using 3 Evaluator cores. This represents an improvement of 6.46 times compared to the software version and 1.23 times relative to the 3 core detector on the ZedBoard. Using 2 cores on the ZC706 board provides almost the same performance of the ZedBoard when using 3 Evaluator cores. By increasing the number of faces we notice that the frame rate of the software solution decreases. The classifier needs more time to evaluate positive windows than negative ones, which are discarded at the beginning of the cascaded stages. The same applies to the hardware system. However, the detection rate is not affected by increasing the number of faces. This is due to the fact that latency of the classification process in the hardware solution is hidden by the double buffering in the Integral Image Buffer Module, which allows transferring image data at the same time it is being processed.

5.3 Accuracy Comparison

In this section, the detection rate and the false positive rate of the hardware and software solutions are measured. A set of images (484 images

containing 641 faces) from the database presented in [6] are used to perform the required measurements.

Table 3: Accuracy comparison

	Detected Faces	Detection Rate	False Alarms	False Positive Rate
Hardware	590	92%	1055	0.034%
OpenCv	619	97%	1551	0.050%

The result shown in Table 3 indicates that the software implementation has a better detection rate compared to the hardware version. There exists two possible reasons for the accuracy degradation. The first is that the hardware detector uses only 7 bits from the 8 bit pixel of the original image to compute the integral image. Using less bits reduces the amount of memory needed to store the integral image, but it reduces the accuracy of the system as well. The second reason encompasses the different downscaling algorithms applied in both implementations. While the linear interpolation is applied for downscaling images in the software, the hardware uses the nearest neighbour interpolation, which may result in a substantial information loss when resizing.

5.4 Comparison With Similar Works

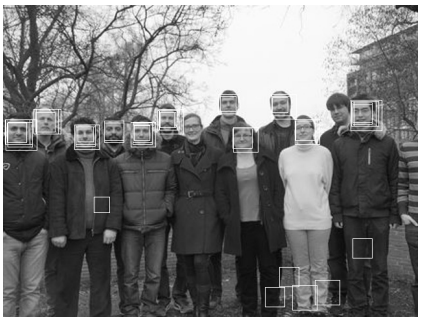
The proposed face detection system is compared to two similar face detection systems presented in [2, 5] and are discussed in Section 3. The comparison is made based on the results presented in both publications. Table 4 illustrates the reached frame rate of the proposed face detector compared to the work presented in [2, 5]. All algorithms use the same scale factor 1.2. On image resolution 640x480, the GPU implementation [5] operates at 15.2 FPS utilizing 4 GPUs. The FPGA implementation [2] reached 16.08 FPS using 8 classifiers (where a classifier is equivalent to a node in this paper). The detection rate and the false positive rate of the referenced works are not mentioned, but in the best case, they are comparable to the rates of the OpenCV implementation.

Table 4: A Comparison of Different Accelerated Versions of Viola and Jone's Algorithm

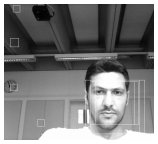
Approach	Number of Features	Resolution	Frame Rate
Proposed System	2094	320x240	66.45
		640x480	16.53
FPGA [2]	2135	320x240	61.02
		640x480	16.08
GPU [5]	x	640x480	15.2

5.5 Image Results

A sample output of the proposed hardware face detection system is shown in Figure 8a. The white squares present the detected face on the image. The result shows that in most cases, faces are successfully detected. Some of the undetected faces has a small rotation angle. This can be explained by the actual training data, which are generated using frontal faces. Some white rectangles point to non-face sub-windows. A simple filter can be implemented to remove such wrong rectangles. An existing face in an image is usually classified in many overlapped rectangles. This property can be exploited to search for overlapped rectangles and rejects single rectangles, which does not overlap with others as illustrated in Figure 8.



(a) Output of the Face Detector on a Test Images



(b) Without Filter

(c) With Filter

Fig. 8: Sample System Output

6 Conclusions

In this paper a software-hardware co-design approach is presented, that enables the detection of frontal faces in real time. This work is based on the object detection framework of Viola and Jones, which makes use of a cascade of classifiers to reduce the computation time and identifies particular Haar features to detect faces. The proposed architecture is flexible, as it allows the use of multiple instances of the face detector. This makes developers free to choose the speed range and reserved resources for this task. For small applications, that requires a face detection speed of 6 FPS or less, only one core will be sufficient, so that the design can fit into low capacity FPGAs. The current implementation runs on the Zynq SoC and receives images over IP network which allows exposing the face detection task as a remote service, that can be consumed from any device connected to the network. Using three Evaluator cores, the ZedBoard system achieves a maximal average frame rate of 13.4 FPS when analysing an image containing 640x480 pixels. This stands for an improvement of 5.25 times compared to the software solution and represents an acceptable results for most real-time systems. On the ZC706 evaluation board, a higher frame rate of 16.58 FPS is achieved. The proposed hardware solution achieved %92 accuracy which is low compared to the software solution (%97) due to the low precision used in the arithmetic operations and different scaling algorithm. The proposed solution achieved higher frame rate compared to other solutions found in the literature.

References

1. Bradski, G.: Opencv library. Dr. Dobb's Journal of Software Tools (2000)
2. Cho, J., Benson, B., Mirzaei, S., Kastner, R.: Parallelized architecture of multiple classifiers for face detection. In: Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on. pp. 75–82 (July 2009)
3. Cho, J., Mirzaei, S., Oberg, J., Kastner, R.: Fpga-based face detection system using haar classifiers. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. pp. 103–112. FPGA '09, ACM, New York, NY, USA (2009)

4. Degtyarev, N., Seredin, O.: Comparative testing of face detection algorithms. In: Proceedings of the 4th International Conference on Image and Signal Processing. pp. 200–209. ICISP'10, Springer-Verlag, Berlin, Heidelberg (2010)
5. Hefenbrock, D., Oberg, J., Thanh, N., Kastner, R., Baden, S.: Accelerating viola-jones face detection to fpga-level using gpus. In: Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on. pp. 11–18 (May 2010)
6. Jain, V., Learned-miller, E.: Fddb: A benchmark for face detection in unconstrained settings. Tech. rep., FDDB (2010)
7. Lai, H.C., Savvides, M., Chen, T.: "proposed fpga hardware architecture for high frame rate (<<100 fps) face detection using feature cascade classifiers". In: Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on. pp. 1–6 (Sept 2007)
8. Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.: Learning multi-scale block local binary patterns for face recognition. In: Lee, S.W., Li, S. (eds.) Advances in Biometrics. Lecture Notes in Computer Science, vol. 4642, pp. 828–837. Springer Berlin Heidelberg (2007)
9. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Image Processing. 2002. Proceedings. 2002 International Conference on. vol. 1, pp. I–900–I–903 vol.1 (2002)
10. Liu, Q., zheng Peng, G.: A robust skin color based face detection algorithm. In: Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on. vol. 2, pp. 525–528 (March 2010)
11. MinGW: Mingw homepage. "http://www.mingw.org" (2014)
12. NVIDIA: Cuda developer zone. "https://developer.nvidia.com/about-cuda" (2014)
13. Störring, M.: Computer Vision and Human Skin Colour: A Ph.D. Dissertation. Computer Vision & Media Technology Laboratory, Aalborg University (2004)
14. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: International Conference on Computer Vision and Pattern Recognition (CVPR). pp. 511–518. IEEE, IEEE (2001)
15. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137154 (2004)
16. Xilinx: Axi reference guide. "http://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/v13_4/ug761_axi_reference_guide.pdf" (2012)
17. Xilinx: Zynq-7000 soc zc706 evaluation kit. "http://www.xilinx.com/publications/prod_mktg/Zynq_ZC706_Prod_Brief.pdf" (2013)
18. Xilinx: Embedded devlopment kit 14.7. http://www.xilinx.com/tools/platform.htm (2014)
19. ZedBoard.org: Zedboard hardware users guide. "http://www.zedboard.org" (2013)
20. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 2879–2886 (June 2012)