On the Acceleration of Viola-Jones Face Detection Algorithm on FPGAs

Curtis Bader, Peter Irgens, Theresa Le, Devansh Saxena, Cristinel Ababei
Department of Electrical and Computer Engineering
Marquette University, Milwaukee WI, USA
Email: {curtis.bader,peter.irgens,theresa.le,devansh.saxena,cristinel.ababei}@marquette.edu

***Abstract*—We present an FPGA based hardware accellerator for the popular Viola-Jones face detection algorithm. We start by first presenting a detailed discussion of the algorithm. The scope of this discussion is to understand the internal structure of the algorithm and thus identify its primary computational bottlenecks. These computational bottlenecks represent the main candidate portions for speed-up via efficient implementation and parallelization on FPGAs. Selected portions of the algorithm are then coded in VHDL. The VHDL implementation is verified on a DE2-115 board, which uses a Cyclone IV FPGA chip. Experimental results are compared against simulations performed using in-house both pure C++ as well as CUDA programmed implementations of the Viola-Jones algorithm. The FPGA accelerated version is $4\times$ faster.**

***Index Terms*—face detection; Viola-Jones algorithm; CUDA programming; FPGA accellerator**

I. Introduction

Field programmable gate arrays (FPGAs) have been gaining a lot in popularity because the gap between their performance and power consumption, compared to specialized ASICs, has been closing in recent years. FPGAs are currently used in virtually any application domain.

II. Related Work

Here, we describe previous attempts to speed-up the Viola-Jones algorithm via various hardware accelerators. The attempts to speed up the Viola-Jones algorithm can be categorized as follows: multithreading, GPU-based, and FPGA-based. By far, the most popular attempt was GPU-based. Hefenbrock et al. [5] were able to realize a 15.2 FPS (frames per second) CUDA-based implementation of running the Viola-Jones algorithm on multiple windows of the image simultaneously. Wang et al. [6] used OpenCL in parallel with the Viola-Jones algorithm and minimized the time cost on both AMD and NVidia GPU platforms. Li et al. [7] achieved a 3.07x speedup with more than a 50% power reduction on the Sandy Bridge processor, a processor that integrates GPU cores along with the CPU cores on the same die. Li et al. [8] used the Sandy Bridge processor again but instead used the SURF (speeded up robust features) algorithm to detect faces and achieved a similar speed-up as before with a speed-up of 1.42 as compared to the single thread implementation on a CPU. Oro et al. [9] used NVidia GPUs and sustained a throughput of 35 FPS under 1080p resolutions. Kong and Deng [11] achieved over 20x speedup using a CPU-GPU cooperative implementation for the Viola-Jones algorithm.

III. Background on Viola-Jones Face Detection Algorithm

In this section, we present a high level description of the Viola-Jones face detection algorithm. Such a description is necessary as it will help us better understand how to port some of its key tasks into FPGA based accellerators later on. The pseudocode description of the Viola-Jones algorithm is presented in Fig.1. In the following paragraphs we discuss several key ingredients that made this algorithm the first successful realtime face detection algorithm.

Algorithm: Viola-Jones Face Detection Algorithm

```
Input: original test image
Output: image with face indicators as rectangles
for i ← 1 to number of scales in image pyramid do
    Downsample image to create image_i
    Compute integral image, image_ii
    for j ← 1 to number of shift steps of sub-window do
        for k ← 1 to number of stages in cascade classifier do
            for l ← 1 to number of filters of stage k do
                Filter detection sub-window
                Accumulate filter outputs
            end for
            if accumulation fails per-stage threshold then
                Reject sub-window as face
                Break this k for loop
            end if
        end for
        if sub-window passed all per-stage threshold checks then
            Accept this window as a face
        end if
    end for
end for
```

Fig. 1. Pseudocode of the Viola-Jones face detection algorithm.

In a face detection algorithm, we must use an accurate numerical description such that it sets human faces apart from other objects in a given image. Such characteristics can be extracted with a committee algorithm called Adaboost [1]. Such a committee can be created with weak classifiers to form a strong classifier by employing a voting mechanism. The Viola-Jones algorithm [2] uses Haar-like *rectangle features* to construct classifiers. A Haar-like rectangle feature is a scalar product between the image and some Haar-like *pattern* or template. An example of a Haar-like pattern is shown in Fig.2.a.

A crucial element of the Viola-Jones algorithm is a technique to compute rectangle features very rapidly [2], [3].

(a) (b)

Fig. 2. (a) Haar-like rectangle feature is calculated only by using the pixels inside the Haar-like pattern (i.e., black and white rectangles). This example pattern illustrates that the area covering the eyes is usually darker than the area just above the cheecks. (b) The sum of the pixels inside rectangle D can be computed with four array references on values of the integral image: $4 + 1 - (2 + 3)$

This technique uses an intermediate representation for the image, the so called *integral image*. The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y). Instead of summing up all the pixels inside a rectangular window, this technique mirrors the use of cumulative distribution functions. Using the integral image any rectangular sum can be computed in four array references as shown in Fig.2.b.

The Viola-Jones algorithm uses so called *cascade classifiers*. A cascade classifier is constructed as a sequence of stages. At each stage a list of filters are applied to the area within the sliding sub-window. An example of such a multistage filter with 25 stages is shown in Fig.3. Each time the sliding sub-window shifts (typically pixel by pixel, but it can be more pixels at a time to further speed things up), the new region within the sliding sub-window is processed through the cascade classifier stage-by-stage. At each stage, the rectangle feature is evaluated and the weak classifier is computed. Then, a threshold check is used to see if the region is rejected as a face candidate or if it needs to continue to be processed in the next stage.

Fig. 3. Illustration of a cascade classier with 25 stages as a decision tree, where at each node a threshold check is done to decide if the sub-window is rejected as no face (False, F) or if it is passed for further processing to the next stage (True, T), which means that the sub-windows still has chances to contain a face.

To be able to detect faces of different sizes, the algorithm works with a pyramid of scaled images (see Fig.4). This effectively allows sweeping using the same set of Haar-like patterns different scaled versions of the initial image. Thus, sliding sub-windows will sweep each of the images from the pyramid as illustrated in Fig.5.

Fig. 4. Pyramid of scaled images. The bottom image is the original image. The others are scaled images obtained via downsampling of the original image using neighboring pixels.

Fig. 5. Illustration of the process of sliding a sub-window. At each step during the sweeping process, the region within the sub-window has the Haar-like feature computed and processed through a cascade classifier.

When the outer-most for loop in the pseudocode description from Fig.1 finishes its execution, the Viola-Jones algorithm would have found and marked with rectangle indicators all faces present in the original test image as well as in the scaled versions of the image. For example, Fig.6 shows the result of running the Viola-Jones algorithm on the test image from Fig.5.

% Time	Cumulative seconds	Self seconds	Calls	Name
54.49	0.67	0.67	35889623	evalWeakClassifier
21.14	0.93	0.26	2360055	int-sqrt
21.14	1.19	0.26	2360055	runCascadeClassifier
1.63	1.21	0.02	18	integralImages
0.81	1.22	0.01	18	nearestNeighbor
0.81	1.23	0.01	18	ScaleImageInvoker

TABLE I
GPROF ANALYSIS OF C++ SEQUENTIAL IMPLEMENTATION

Fig. 6. The output of running the Viola-Jones algorithm shows detected faces via rectangle indicators superimposed on the original test image.

IV. COMPLEXITY ANALYSIS OF VIOLA-JONES ALGORITHM

The GNU profiler, *gprof* was used to determine the computational complexity of an in-house C++ implementation of the Viola-Jones algorithm. It helped to determine which parts of the program were taking most of the execution time. The sequential C++ implementation was run on a Linux machine with an Intel Quad-core i7-2600 processor with a processing speed of 3.40 GHz. The gprof flat profile revealed the functions with the highest computational time. These functions are depicted in Table 1. This sequential C++ implementation will be parallelized using p threads which will be used to make multiple simultaneous calls to these functions. The quad-core processor will ensure a fast concurrent computation of these threads.

(This section will be updated with the data after the C++ parallelization)

V. VHDL IMPLEMENTATION

Once portions of the VJ algorithm have been identified as the main candidates for implementation on FPGAs, we code their functionality in VHDL.

VI. EXPERIMENTAL AND SIMULATIONS RESULTS

Here, we perform report results of the comparison between the FPGA based accelerated VJ algorithm and the C++ implementation runing on a Linux machine.

VII. DISCUSSION

In our experiments, we found that.

VIII. CONCLUSION

We proposed an FPGA based implementation of the first face detection algorithm that was designed for real time applications. Our entire simulation framework is publicly available at [4].

REFERENCES

[1] Y. Freund and R.E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting," *J. of Computer and System Sciences*, vol. 55, no. SS971504, pp. 119-139, 1997.

[2] P.A. Viola and M.J. Jones, “Rapid object detection using a boosted cascade of simple features," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[3] Y.-Q. Wang, “An analysis of the Viola-Jones face detection Algorithm," *Image Processing On Line (IPOL)*, ISSN 2105-1232, 2014.

[4] Software downloads at MESS Lab, Marquette University, 2014. [Online]. Available: http://dejazzer.com/software.html.

[5] D. Hefenbrock, J. Oberg, N. Thanh, R. Kastner and S. Baden, “Accelerating Viola-Jones Face Detection to FPGA-Level Using GPUs," *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 2010.

[6] W. Wang, Y. Zhang, S. Yan, Y. Zhang and H. Jia, “Parallelization and performance optimization on face detection algorithm with OpenCL: A case study," *Tinshhua Sci. Technol.*, vol. 17, no. 3, pp. 287-295, 2012.

[7] E. Li, B. Wang, L. Yang, Y. Peng, Y. Du, Y. Zhang and Y. Chiu, “GPU and CPU Cooperative Accelaration for Face Detection on Modern Processors", *2012 IEEE International Conference on Multimedia and Expo*, 2012.

[8] E. Li, L. Yang, B. Wang, J. Li and Y. Peng, “SURF cascade face detection acceleration on Sandy Bridge processor", *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012.

[9] D. Oro, C. Fernandez, J. Saeta, X. Martorell and J. Hernando, “Real-time GPU-based face detection in HD video sequences", *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011.

[10] L. Acasandrei and A. Barriga, “Design methodology for face detection acceleration", *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 2013.

[11] J. Kong and Y. Deng, “GPU accelerated face detection", *2010 International Conference on Intelligent Control and Information Processing*, 2010.

[12] M. Che and Y. Chang, “A Hardware/Software Co-design of a Face Detection Algorithm Based on FPGA", *2010 International Conference on Measuring Technology and Mechatronics Automation*, 2010.

[13] C. Kumar and S. Agarwal, “A novel architecture for dynamic integral image generation for Haar-based face detection on FPGA", *TENCON 2014 - 2014 IEEE Region 10 Conference*, 2014.