Embedded Face Detection Implementation

Laurentiu Acasandrei, Angel Barriga

Instituto de Microelectronica de Sevilla
IMSE-CNM-CSIC
C/ Américo Vespucio, s/n (esquina Leonardo da Vinci)
Sevilla, Spain
laurentiu@imse-cnm.csic.es
barriga@imse-cnm.csic.es

Abstract: In this communication an embedded implementation of the Viola-Jones face detection algorithm targeting low frequency, low memory, and low power consumption, is presented. The design methodology, performance analysis and algorithm optimization in order to accelerate the face detection process, will be described. The resulted implementation is platform independent and achieves on average a 3 times detection speed up.

1 Introduction

Face detection is an important aspect for biometrics, video surveillance and human computer interaction. Detection systems require huge computational and memory resources due to the complexity of detection algorithms. Implementation difficulties appear when it is required to apply face detection techniques in embedded systems, where there are restrictions in size and power consumption.

Automatic face detection systems have experienced a remarkable progress in last decade. However, for large image dimensions, most of applications of these systems are based on software realizations running on computers. Usually on embedded systems like photo/video cameras, smartphones and tablets the detection is done on images scaled to lower resolutions, but this introduces the disadvantage of not being able to detect smaller to medium dimension faces that appear in the original image. To detect faces from full size images or video frames is necessary the adaptation of face detection algorithms to those embedded systems.

The purpose of this communication is to describe a face detection application with high performance that can be run smoothly on any embedded system. Taking into consideration that the OpenCV library comes with a baseline application for video face detection we have used the sources of that application as starting point in developing the video detection application for low resources embedded systems (with reduced memory resources, without floating point arithmetic unit, etc).

The structure of the paper is the following. Section 2 introduces the face detection technique. Section 3 presents the design methodology of the face detection embedded system. Section 4 exposes the embedded software face detection implementation. Section 5 focuses in the analysis of the proposed solution, while in section 6 some optimizations in order to accelerate the detection algorithm have been proposed. Finally, Section 7 summarizes the main conclusions of this research work.

2 Face Detection Technique

The face detection technique is based on the face detection framework proposed by Viola-Jones [VJ04]. The proposed framework is capable of processing images extremely rapidly while achieving high detection rates. The speed of the face detection framework relies on three important key components. Firstly, the image is transformed into "Integral Image" which allows the features, used by the detector, to be computed very quickly. Secondly, the used classifier is simple and efficient and is build using the AdaBoost learning algorithm [SFB98] to select a small number of critical visual features from a very large set of potential features. And thirdly, the classifier is formed by combining weak classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions. Viola-Jones technique is based on exploring the image by means of a window looking for features. This window is scaled to find faces of different sizes. The system architecture is based on a cascade of detectors. The first stages consist of simple detectors, very fast and low cost, that allow eliminating those windows that do not contain faces. In the successive stages the complexity of detectors are increased in order to make a more detailed analysis of features. A face is detected only if it makes it through the entire cascade. The Haar-like features used by the classifier consist of rectangular areas whose processing requires simple arithmetical operations. The calculation is based on the sum of the pixels of each rectangular region weighed by a weight. At all scales, these features form the "raw material" that will be used by the detector. The set of rectangle features in the image is quite large and overcomplete, so to reduce that number the AdaBoost learning algorithm [SFB98] is applied. The Viola-Jones classifier employs AdaBoost at each node in the cascade to learn a high detection rate at the cost of low rejection rate multitree (mostly multistump) classifier at each node of the cascade. To facilitate the processing of the features, the operations are not made on the original image but on an integral image. Therefore the detection algorithm requires a preprocessing step that calculates this integral image. The advantage of the integral image is that it allows calculating the sum of any rectangle in constant time. The integration of the image consists of adding for each pixel the values of the previous pixels.

3 Design methodology

The starting point of the design methodology of the embedded system is the OpenCV face detection framework application. OpenCV (Open Source Computer Vision), started

by Intel in 1999, is a library of programming functions for real time computer vision [Op13]. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It is written in C/C++ and was designed for computational efficiency and with a strong focus on real-time applications. OpenCV comes with a face detection application which is the starting point of the proposed methodology. The target of the proposed face detection system is an embedded implementation using an AMBA bus processor. The embedded architecture of the face detection system is based on a software implementation within an operating environment running on the LEON3 processor. The LEON3 is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture [Leo13]. The model is highly configurable, and particularly suitable for *System-on-Chip* (SOC) designs.

The design flow is based on three stages, as shown in Figure 1. In the first stage an adaptation of the software application to execute on the embedded system has been made. In the next stage an analysis of the new embedded software application is performed, in order to detect "bottlenecks" and those tasks that are suitable to accelerate through hardware implementation. In the third phase, as result of the previous analysis, solutions to optimize and accelerate some of the tasks of the face detection process will be proposed. At each stage of the design process the performance of the face detection system is analyzed, in terms of speed and quality.

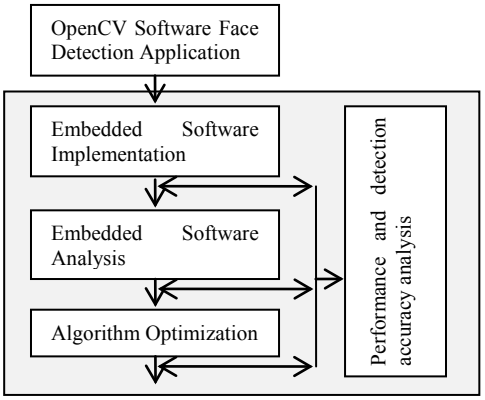

Figure 1: Design methodology

4 Embedded Software Face Detection Implementation

Viola and Jones organized each boosted classifier group into nodes of a rejection cascade. Each of the nodes contains an entire boosted cascade of groups of decision stumps (or trees) trained on the Haar-like features from faces and nonfaces (or other objects the user has chosen to train on). Typically, the nodes are ordered from least to most complex so that computations are minimized (simple nodes are tried first) when rejecting easy regions of the image. Typically, the boosting in each node is tuned to have

a very high detection rate (at the usual cost of many false positives). When training on faces, for example, almost all (99.9%) of the faces are found [BK08] but many (about 50%) of the nonfaces are erroneously "detected" at each node. But this is satisfactory because using (say) 20 nodes will still yield a face detection rate (through the whole cascade) of $0.999^{20} \approx 98\%$ with a false positive rate of only $0.5^{20} \approx 0.000001\%$. During the run mode, a search window of different sizes is swept over the original image. In practice, 70–80% of nonfaces are rejected in the first two nodes of the rejection cascade, where each node uses about ten decision stumps. This quick and early "attentional reject" vastly speeds up face detection. The Haar-like features are trained to be applied for evaluating rectangular window of 20x20 pixels. For other dimensions of the evaluating window the Haar-like features must be scaled correspondingly. The face detection system consists of 22 cascade detectors, containing 2135 Haar like features.

The first task of software/hardware codesign was adapting and optimizing the OpenCV baseline application for an embedded environment. We have considered that the majority of embedded environments are capable of running C/C++ applications with or without Operating System (OS) support. This means that the resulting application code has to be compatible for both C, C++ compilers and in the same time platform independent. Another consideration made is the fact that most of the SoC have no floating point support. For it, the resulting application uses integer operations instead of floating point operations in order to preserve the generality of the application for the embedded system world. An important moment in this step was finding an acceptable scaling coefficient of the floating point variables and data to integer variables and data. After trying different values and comparing the resulted integer application with the floating point application we found that by scaling with 20 bits (precision of 20 bits for the floating point decimals) the integer and floating point applications obtain identical results. Also the floating point squared root function necessary to calculate variance of the evaluating window was replaced with fast integer squared root function.

In the end it was obtained a face detection standalone application compatible with C/C++, using only integer type operations and data, where the cascade of classifiers and the image can be loaded from a desired memory location which is previously initialized with a raw RGB image. The accuracy of the resulted standalone application is the same as the OpenCV face detection application.

5 Embedded Software Face Detection Analysis

The next step in application development was analyzing different modes of detection in order to find the run time bottlenecks and optimize the detection. For the embedded target we have decided to use the detection mode where the detector (Haar-like feature) is scaled and the biggest regions containing faces are searched within an image. In this mode the detections starts with the biggest evaluation window and biggest evaluation step and progressively, the window together with the evaluation step are decreased until a region containing a face is detected. In the case that a region with face or multiple faces is detected, the attention of the detector concentrates in that region.

The trained classifier cascade (Haar-like feature) is provided by OpenCV in an XML file format. Using this XML format in an embedded system will produce memory and run time overhead. In order to avoid the overhead we have developed an application that receives a XML file, interprets the data and saves it in a simpler format to a C header file. The resulted embedded application can be compiled with the cascade of classifier or the data can be transmitted during the execution of the application via an appropriate interface. In order to obtain relevant insight about which parts (or functions) of the face detection program are taking most of the execution time GNU gprof tool was used. This tool permits one to learn where the program spends its time and the function calling tree during the execution. It can also tell which functions are being called more or less than are expected. Table I shows the obtained results.

Table I: Analysis of Detection System Bottlenecks

Time %	# calls	Function name
24.64	17	SetMatZero()
20.16	3201	RunHaarClassifierCascadeEmbedd()
14.81	16	SetImagesForHaarClassifierCascadeEmbedd()
13.39	1	Integral()
11.35	1	LinkDataToEmbeddClassifierCascade()
4.22	511	HResizeLinear()
3.11	262144	saturate_uchar()
2.62	512	VResizeLinear()
1.80	296384	sum_elem_ptr()
1.10	3201	isqrt64()

As we can see the function `SetMatZero()` uses 24.64% of the executing time even surpassing the time spent applying the cascade Haar like-features (20.16%) for the entire image. The function `SetMatZero()` is used to set to zero all the elements of a temporary matrix having the same dimension of the image. In this matrix the top left coordinates of a detected face are flagged with value 1. In this mode the detections starts with the biggest evaluation window and biggest step and progressively, the window and the step are decreased until a region containing a face is detected. The detection is done in two steps. In the first step the image is scanned with the evaluation window by applying only the first two Haar-like feature stages in order to rapidly detect regions containing potential faces. If a region is found to have a potential face then the coordinates are set to value one in the temporary matrix. In the second step each potential face (starting with their coordinates) from the temporary matrix is evaluated with the remaining Haar-like feature stages. If a true face is detected then the coordinates, width and height are stored in a list. At the end of the second step, the temporary matrix is set to zero in preparation for the next image where the evaluating window has smaller dimensions.

We can improve the speed by not using the function `SetMatZero()` at the end of the second step and instead each time after we have detected a face during the second step and that face is stored into a list, we set to zero the coordinates inside the temporary matrix. After applying this change, the detection time is improved with 24.64 %.

6 Embedded Face Detection Optimization

The OpenCv face detection baseline application implements detection in two distinct modes (see Figure 2). In mode 1 the image is scaled using linear interpolation until it reaches a predefined minimal dimension. Each time the image is scaled the two integral images (normal= $\sum x$ and squared= $\sum x^2$), needed for variance, are recalculated for the scaled image. The search window has fixed dimension during the detection process. In mode 2 the integral images(normal= $\sum x$ and squared= $\sum x^2$), needed for variance, are calculated only once for the original image but the Haar-Like features form the classifier are scaled progressively until their dimensions are close to the dimension of the original window. This mode lacks the interpolator used in mode 1. The search window has a variable dimension during detection process.

The mode 1 and mode 2 have different scaling and search window control. In both detection mode (mode 1 and mode 2) the Haar Like-features components (weights and dimensions) are scaled proportionally with the dimensions of search window. If we do not scale the Haar-like feature weights and adjust the variance computation $\sigma = \sqrt{\frac{\sum x^2}{W \cdot H} - \left(\frac{\sum x}{W \cdot H}\right)^2}$ by using the formula $\sigma_{adjusted} = \sqrt{W \cdot H \cdot \sum x^2 - \left(\sum x\right)^2}$, it results that the number of arithmetic operations (like division and multiplication) and memory accesses are decreased substantially. This will make the algorithm perform faster due to the reduced number of computation for the adjusted variance of the search window [AB11]. Figure 2 shows the proposed optimization. In order to compare the performance of the OpenCV 2.2 baseline face detection with the accelerated Viola-Jones algorithm, and to analyze the influence of the configuration parameters, both implementations have been compiled and speed optimized for 64 bit Win7 OS using Visual Studio 2010 Professional edition. Both implementation (OpenCV's implementation and accelerated Viola-Jones version) have received the same test VGA(640x480) images and the same configuration parameters. In Table II we have different scale factor (sf) and minimum search window dimensions (swd): sf=1.1 and swd=20x20 (Conf1); sf=1.2 and swd=20x20 (Conf2); sf=1.1 and swd=30x30 (Conf3).

Table II: Performances of OpenCv and accelerated Viola-Jones implementation for different parameters

		Mode 1. Img scaling		Mode 2. Haar scaling	
		OpenCV Baseline	Optimized version	OpenCV Baseline	Optimized version
Conf1	speed	708.8 ms 1.41 FPS	185.7ms 5.38FPS	843.9 ms 1.18FPS	188.1ms 5.3FPS
	Search windows	602348	599816	697582	631343
Conf2	speed	409.5 ms 2.44FPS	164.8ms 6.06FPS	479.7ms 2.08FPS	140.1ms 7.1FPS
	Search windows	354321	353935	381474	364635
Conf3	speed	348.1ms 2.87FPS	99.4ms 10FPS	456.7 ms 2.18FPS	130.6ms 7.6FPS
	Search windows	352718	351184	402519	332917

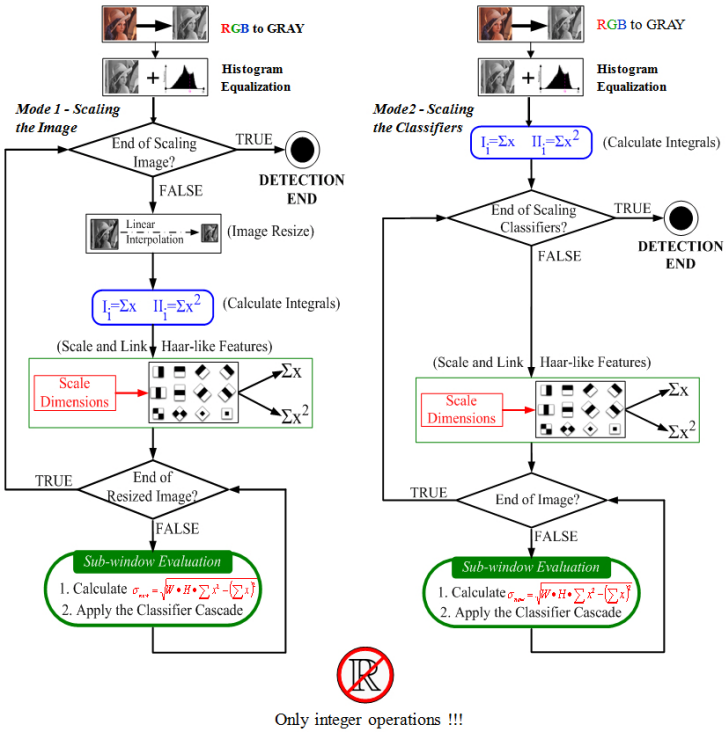

Figure 2: Proposed face detection acceleration algorithm

As the proposed implementation has kept the control mechanism for search windows identical with the one form the OpenCV baseline it is difficult to make a comparison with previous work done in accelerating detection due to the lack of setup information and how many search windows are evaluated. There is one exception Cho et all [CMO09] where they use mode1 with a scaling factor of 1.2, minimal search window dimension of 20x20 and the search window is applied with a vertical horizontal step of 1. For the Cho et all setup, we measure for the OpenCV implementation a execution time of 972.3 ms for a total of 881484 search windows and the accelerated versions of Viola-Jones has an execution time of 451.2 ms for 880585 search windows. As shown in Table II the number of search windows depends heavily on the configuration setup and also of the control mechanism. Measuring the number of searched windows performed by the detection system gives more realistic information about the detection performance. The speeds obtained by the accelerated Viola-Jones implementation in some configuration are faster that any single GPU acceleration [H09] of 4.2 FPS or FPGA implementation [CMO09] of 6.5 FPS, and for the lowest number of search windows (see Table II) it has closer performances to [HOT10] of 15.2 FPS. The accuracy of the accelerated Viola-Jones version is the same as the OpenCV face detection application. In order to analyze the detection accuracy a test software was developed. A PC based test bench software configures LEON3 based detection system and sends the test images. Then it receives the detection results for further analysis. The test setup is based on 2409 frontal face images from the color FERET database [PWH98].

7 Conclusions

This communication presents an embedded software implementation of the Viola-Jones face detection algorithm targeted for low frequency, low memory and low power embedded systems. The starting point was the trained classifier cascade (Haar-like feature) provided by OpenCV library resources for video face detection. For it some modifications has been make adapting and optimizing the OpenCV baseline application for an embedded environment. This modifications can be summarized in changing the floating point operations by integer ones, analyzing the performance in order to detect the system bottlenecks, and algorithmic speed-up by not scaling the Haar-Like feature weights and adjusting the computation of variance.

Acknowledgement

This work was supported in part by Spanish Ministerio de Ciencia y Tecnología under the Project TEC2011-24319, and by Junta de Andalucía under the Project P08-TIC-03674, and by Spanish Ministerio de Economía y Competitividad under the Project IPT-2012-0695-390000. Co-financed by FEDER.

References

[AB11] L. Acasandrei, A. Barriga: “Accelerating Viola-Jones Face Detection for Embedded and SoC Environments”, Fifth ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC’2011), Ghent, Belgium, Aug. 2011.

[BK08] G. Bradski, A. Kaehler, “Learning OpenCV”, O’Reilly Media, pp.506-516, September 2008.

[CMO09] J.Cho, S. Mirzaei, J. Oberg, R. Kastner, “FPGA-Based Face Detection System Using Haar Classifiers”, Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'09), pp. 103-112, 2009.

[H09] J. P. Harvey, “Gpu acceleration of object classification algorithms using nvidia cuda”, Master’s thesis, Rochester Institute of Technology, Rochester, NY, Sept. 2009.

[HOT10] D. Hefenbrock, J. Oberg, N.T.N. Thanh, R. Kastner, S.B. Baden, “Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs”, Proc. IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, 2010.

[Leo13] LEON3, link: http:// www.gaisler.com//

[Op13] OpenCV, link: http://sourceforge.net/projects/opencvlibrary/

[PWH98] P.J. Phillips, H. Wechsler, J. Huang, P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms," Image and Vision Computing J, Vol. 16, No. 5, pp. 295-306, 1998.

[SFB98] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, “Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods”, The Annals of Statistics, pp. 1651-1686, 1998.

[VJ04] P. Viola, M.J. Jones, “Robust Real-Time Face Detection”, International Journal of Computer Vision, v.57 n.2, pp.137-154, May 2004.