FPGA implementation of an embedded face detection system based on LEON3

L. Acasandrei[1] **and A. Barriga**[2]
[1] IMSE-CNM-CSIC, Seville, Spain
[2] IMSE-CNM-CSIC/University of Seville, Seville, Spain

Abstract - *This paper presents an FPGA face detection embedded system. In order achieve acceleration in the face detection process a hardware-software codesign technique is proposed. The paper describes the face detection acceleration mechanism. It also describes the implementation of an IP module that allows hardware acceleration.*

Keywords: Face detection, image processing, embedded systems, hardware implementation

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. A software detection realization implemented on a low speed, low resource, low power SoC (System on Chip) is not efficient. Instead a hardware-software codesign approach can be used to build hardware accelerators for most computational consuming parts of detection algorithms.

The main challenge in face detection design is finding an acceptable balance between detection accuracy (robustness) and operation efficiency (computation cost). Heuristics or knowledge-based face detectors, such as color-based and template-based detectors, use direct knowledge about faces and often give fast performance, but they are usually less robust with respect to large face variances and background interferences. On the contrary, statistical or learning-based face detectors, like the neural-network and SVM-based detectors make use of powerful pattern classification algorithms and provide better performance in discriminating face and non-face patterns. However, these learning-based algorithms often involve high processing complexity, which can be too costly for applications in embedded systems.

Recently there have been some proposals for face detection systems hardware implementations. Thus in [1] a dedicated system on FPGA is shown. The system receives the image from a camera and stored it in the internal FPGA memory. Other realizations are based on GPUs. In [2] they proposed face detection acceleration by distributing the computation between 4 GPUs. This communication presents the design of an embedded face detection system based on LEON3 SPARC V8 processor. The embedded face detection system implements the popular Viola-Jones object detection framework for face-like objects.

2 Viola-Jones Face Detection Algorithm

The face detection technique is based on the face detection framework proposed by Viola-Jones [3]. 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 which is build using the AdaBoost learning algorithm [4] 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.

The detection algorithm requires a preprocessing step that calculates the integral image. The integration of the image consists of adding for each pixel the values of the previous pixels.

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y') \qquad (1)$$

The advantage of the integral image is that it allows to calculate the sum of any rectangle in constant time as shown in Figure 1.

The Haar-like features used by the classifier are shown in Figure 2. They 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. The light region is interpreted as “add that area” and the dark region as “subtract that area”. 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 applies the AdaBoost learning algorithm [4]. 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.

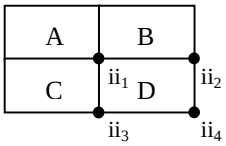


Fig. 1. The sum of the pixels of rectangle D is calculated as the following operation on the integral image: $ii_4+ii_1-(ii_2+ii_3)$

Fig. 2. Haar-like features

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 according to Figure 3. The first stages consist of simple detectors, very fast and low cost, that allows to eliminate 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.

Fig. 3. Architecture of cascade detectors

A threshold is applied to sums and differences of rectangular image regions. For the Viola-Jones rejection cascade, the weak classifiers that it boosts in each node are decision trees that often are only one level deep (i.e., "decision stumps"). A decision stump is allowed just one decision of the following form: "Is the value *v* of a particular feature *f* above or below some threshold *t*"; then, for example, a "yes" indicates face and a "no" indicates no face:

$$f_i = \begin{cases} +1 & v_i \geq t \\ -1 & v_i < t \end{cases} \quad (2)$$

The number of Haar-like features that the Viola-Jones classifier uses in each weak classifier can be set in training, but mostly we use a single feature (i.e., a tree with a single split) or at most about three features. Boosting then iteratively builds up a classifier as a weighted sum of these kinds of weak classifiers. The Viola-Jones classifier uses the classification function:

$$F = sign(w_1 \cdot f_1 + w_2 \cdot f_2 + ... + w_n \cdot f_n) \quad (3)$$

Here, the sign function returns –1 if the number is less than zero, 0 if the number equals zero, and 1 if the number is positive. On the first pass through the data set, we learn the threshold t_1 of f_1 that best classifies the input. Boosting then uses the resulting errors to calculate the weighted vote w_1. As in traditional AdaBoost, each feature vector (data point) is also re-weighted low or high according to whether it was classified correctly or not in that iteration of the classifier. Once a node is learned this way, the surviving data from higher up in the cascade is used to train the next node and so on.

3 Hardware-Software Codesign

For the Viola-Jones object detection algorithm, an implementation in software or hardware or both combined still uses a large amount of computational resources and it needs a high memory bandwidth. Therefore, this constitutes an impediment for building real time object detection systems. In order to obtain an accelerated face detector hardware-software codesign techniques were used as shown in Figure 4. As a result some parts of the face detection algorithm that required flexibility were implemented in software while other parts that were time critical were implemented as hardware accelerators.

3.1 Accelerating Viola-Jones Face Detection

The Viola-Jones face detection framework was implemented in OpenCV (*Open Source Computer Vision*) [5] as a full-fledged face recognition application. OpenCV is a programming functions library for real time computer vision. Taking into consideration that the OpenCV library comes with a baseline application for video face detection it was decided to use the application source files as starting point in developing the video detection application for LEON3 embedded system.

The Haar-like features from OpenCV distribution are trained to be applied for a search rectangular window of 20x20 pixels. For other dimensions of the search window the Haar-like features must be scaled correspondingly. The face detection system consists of 22 cascade detectors also called stages, containing 2135 Haar-like features.

The first modification of the OpenCV implementation is based in 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.

The OpenCV face detection baseline application implements detection in two distinct modes.

Mode 1: Face detection by scaling the image. In this mode the image is scaled using 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.

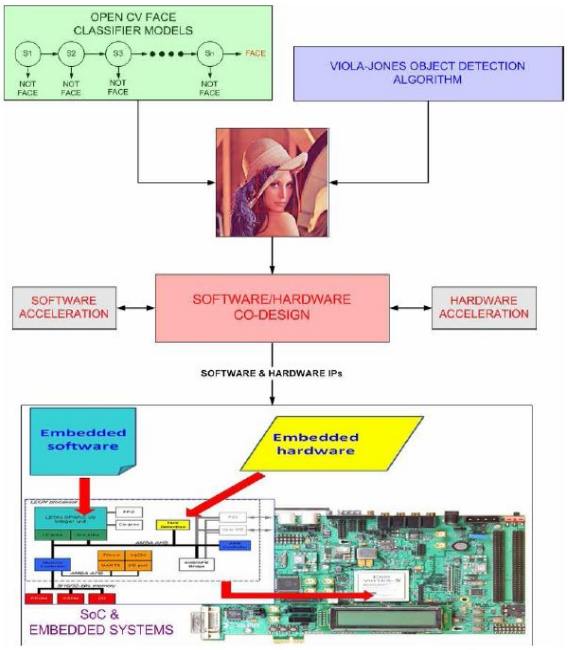


Fig. 4. Hardware-software codesign for face detection

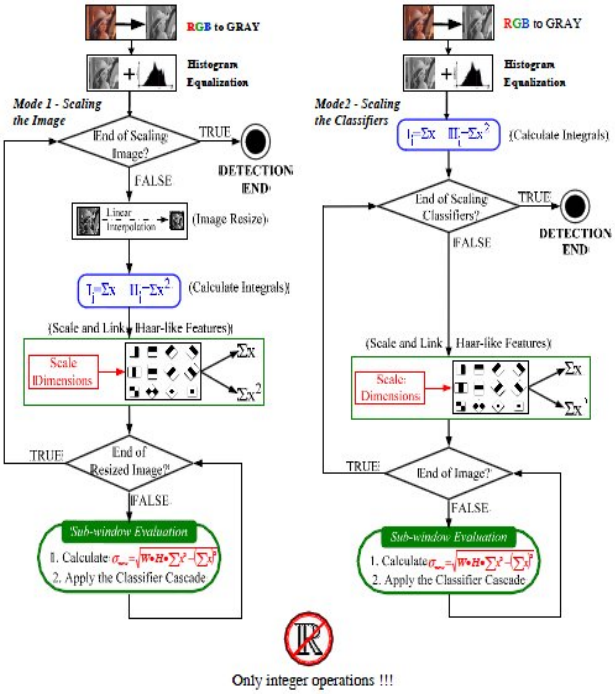


Fig. 5. Proposed face detection acceleration algorithm

Mode 2: Face detection by scaling the classifiers (see Figure 5). In this mode the integral images (normal=$\sum x$ and squared=$\sum x^2$), needed for variance normalization $\sigma^2 = \frac{\sum x^2}{W \times H} - \left(\frac{\sum x}{W \times H}\right)^2$, are calculated only once for the original image. However, the Harr-like features from 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.

In both detection mode the Haar-like features components (weights and dimensions) are scaled proportionally with the dimensions of search window. That means for a search window of dimension *WxH*, the weight of each rectangle forming the Haar-like features are scaled with *WxH*. In the search window, the sum of each applied Haar-like feature is calculated using:

$$HaarFeature^{Sum} = \sum_{I=1}^{3} Area_I \bullet Weight_I^{scaled} \qquad (4)$$

Area represents the sum of all pixels inside a component and *I*=1,2 or 3 represents the number of components for that Haar-like feature. In order to determine the next weight value for the stage sum, each $HaarFeature^{Sum}$ is compared with each normalized threshold of the respective Haar-like feature as:

$$StageSum = \begin{cases} StageSum + Stage_J^{Weight2}, if\ HaarFeature_J^{Sum} > Thres_J^{norm} \\ StageSum + Stage_J^{Weight1}, if\ HaarFeature_J^{Sum} < Thres_J^{norm} \end{cases} \qquad (5)$$

where *J*=[1…2135] represents the Haar-like feature indexes in a stage and $Thres_J^{norm} = \sigma Thres_J^{HaarFeature}$ (σ is the search window standard deviation).

If we do not scale the Haar-like feature weights and adjust the variance computation by using the formula $\sigma^2_{adjusted} = W \bullet H \bullet (\sum x^2 - (\sum x)^2)$, it results that the number of arithmetic operations (division, multiplication) and memory accesses are decreased substantially. This will make the algorithm perform faster due to a reduced number of operations needed for computation of the adjusted variance for the search window [6].

3.2 Hardware IP for detection acceleration

After a careful analysis of the face detection application it was found that the software bottleneck resided in the huge amount of memory read access, multiplication, and squared root operations done by all the search window evaluations. In order to detect a face from an image, hundreds of thousands of search windows are evaluated and this represents the most time consuming part of the application. Therefore it was decided to accelerate the evaluation of search windows by means an IP module [7].

In order to keep a high degree of flexibly and share the hardware resources with the rest of the LEON3 system it was decided for the IMSE_OBJECT_DETECTION IP to have two operating modes: the free mode in which LEON3 processor can use the IP core resources (multipliers, shared memory, etc) to implement other functionalities, and the face detection mode.

The IMSE_OBJECT_DETECTION and all internal modules components are clocked by the system clock (80 Mhz).

As it was previously mentioned the IP module implements the search window algorithm. The software application will load the compressed form of Haar-like features into the Shared Memory before any detection operation. Before starting any detection operation the configuration registers (scale, x-y coordinates, image dimension, etc.) must be configured with the desired configuration values.

When the start command is given the face detection procedure is fully controlled by the component *Imse_stage_evaluator_unit* (see figure 6). This unit is the core engine for accelerating face detection. At the end of the detection the component signalize if a face is present, the Status register is updated with the detection result and an interrupt is generated.

The *Imse_stage_evaluator_unit* has a multiple state machine control in order to deal with variable memory access latencies. Beside the multiple state machine control this unit contains the following specialized modules:

- *Haar_feature_rect_calc*: This module is used to calculate the area of integral rectangles using only the corners data.
- *Haar_feature_scaler*: This is a pipelined module for Haar-like feature scaling and search window address computation.
- *Sqrt64_array_pipe16*: A 64 bit pipelined SQRT unit that has data output latency of 16 clocks.
- *Mul41x33signed*: A 41x33 signed multiplier.

The Register Bank contains APB bus accessible registers that are used for the core configuration and control.

The APB Slave Interface connects the IP module to the APB bus and enables the LEON3 processor to access the registers from Register Bank. The AHB Master/Slave Interface is a simple DMA interface.

The IP module has a shared memory based on a dual-port RAM with AHB interface. The Shared Memory is used by the IP module to store the compressed Haar-like features. When the module works in 'Free Mode' LEON3 can use the Shared Memory as additional RAM memory.

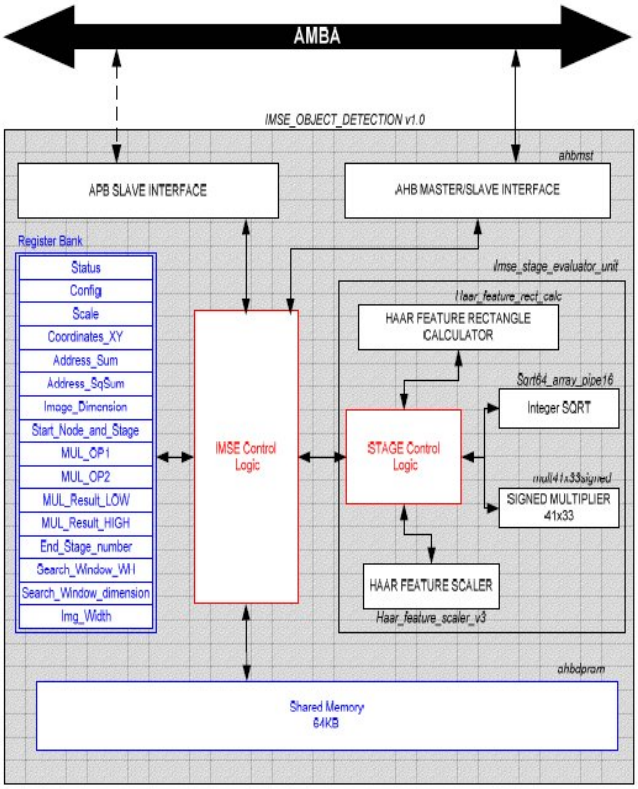


Fig. 6. IMSE_OBJECT_DETECTION IP block diagram

4 Results

The proposed LEON3 face detection system works with images (colored or grey) having a resolution smaller than 10242 pixels. It fully uses the OpenCV cascade of classifiers for frontal faces and it can store into the IP Shared Memory approximately 2730 Haar-like classifiers. It also works with a greater number of Haar-like classifiers but the extra classifiers must be store into the program memory and then loaded into the Shared Memory at the appropriate moment.

The system was implemented on a Xilinx XC5VLX50 FPGA. The entire LEON3 face detection system uses 6,435 slices (up to 89% of the device utilization) and 10,962 of flip-flops (up to 38% of the device utilization). The estimated static power consumption (measured with Xpower Analyzer from Xilinx) for the LEON3 core is 603 mW. The most consuming components are the DDR2 memory controller (216 mW), DVI interface (136.06 mW) and the clock generators. The LEON3 processor consumes 32.39 mW and even though the IMSE_OBJECT_DETECTION IP uses more flip flops and has approximately the same amount of logic, its power consumption is 6 times less (5.39 mW) than the LEON3 processor.

4.1 Performance

Number section and subsection headings consecutively in numbers and type them in bold. Use point size 14 for section headings and 12 for subsection headings and 10 for subsection within a subsection.

In order to measure the detection performances of the LEON3 embedded detection system three distinct software implementations for face detection were compared:

- Ported OpenCV software for embedded systems.
- Software accelerated version of the ported software.
- Hardware + Software accelerated version of the ported software.

The measured performances metrics are the execution time and the number of searched windows performed.

For the first two implementations the performances were measured for two distinct modes of detection (mode 1 and mode 2). In Hardware + Software accelerated version only the performance of mode 2 (Haar-like features are scaled) was measured. For each mode, four different set-up parameters were used (setup 1 to 4) for the minimum size search window (S) and the scale step (step): 1) S=30x30, step=1.2; 2) S=30x30, step=1.1; 3) S=20x20, step=1.2; 4) S=20x20, step=1.1.

From figure 7 it results that the accelerated face detection application is 3-4 times faster than the ported OpenCV application for both modes. Using the hardware acceleration IP the face detection is 7-9 times faster than the ported face detection application running exclusively on LEON3 processor core.

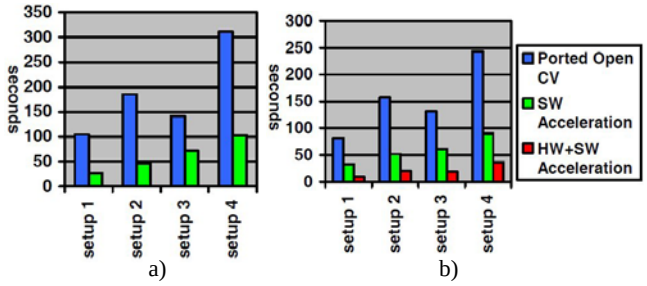


Fig. 7. Detection times of three distinct implementations for VGA image. a) Scale Image mode (Mode 1), b) Scale Haar-like features mode (Mode 2)

4.2 Detection accuracy

In order to analyze the detection accuracy an specific software has been developed. Figure 8 shows the test bench scheme. The PC based test bench software configures LEON3 based detection system and send 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 [8]

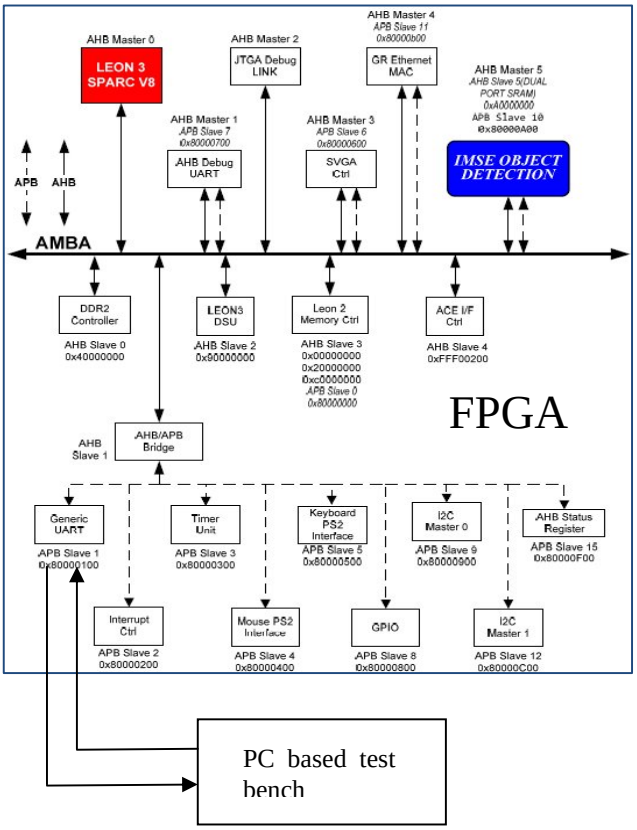
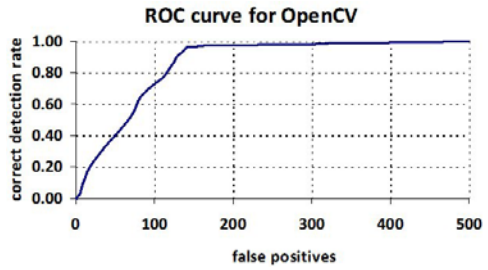


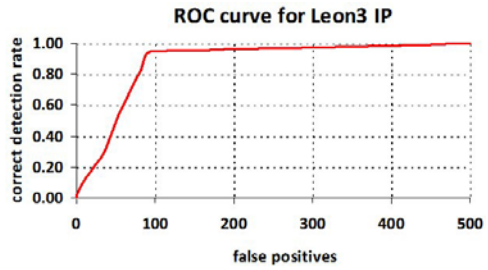
Fig. 8. Testbench system block diagram

The accuracy of face detection can be described using receiver operating characteristic (ROC), which is a curve widely adopted in signal-detection theory. An ROC is essentially a scatterplot that shows the relationship between the false acceptance rate and the true acceptance rate. The false acceptance rate measures the likelihood of a face detector to incorrectly accept a background image as a face. The true acceptance rate measures the likelihood of a face detector to correctly identify a face. In the literature [9] true acceptance rate is also referred as face detection rate.

The ROC curve of a given face detector shows its performance as a trade-off between the false acceptance rate and the face detection rate by varying its discrimination criterion (e.g. a threshold parameter). Figure 9 shows the ROC curves for OpenCV software and the IP module. Both have very similar results. There is a small difference between the two ROC curves because the data result (i.e detected faces) aggregation mechanism is slightly different in the analysis tools.



a)



b)

Fig. 9. ROC curves: a) OpenCV face detection software, b) IMSE_OBJECT_DETECTION IP based system

5 Conclusions

The LEON3 face detection system is a flexible embedded SoC containing sufficient hardware resources to make this system capable to run a wide variety of applications. One important feature of this system is that it contains a dedicated face detection hardware accelerator IP. The face detection hardware accelerator IP is built on the principle of resource sharing, i.e when the system is performing different task tan face detection, the hardware IP resources can be used freely by the system.

6 Aknowledgments

This work was supported in part by the European Community under the MOBY-DIC Project FP7-IST-248858, by Spanish Ministerio de Ciencia y Tecnología under the Project TEC2011-24319, and by Junta de Andalucía under the Project P08-TIC-03674.Co-financed by FEDER

7 References

[1] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, “Fgpa-based face detection system using haar classifiers,” in FPGA ’09: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays. New York, NY, USA: ACM, pp. 103–112, 2009.

[2] 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.

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

[4] 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.

[5] OpenCV: http://sourceforge.net/projects/opencvlibrary/

[6] 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.

[7] L. Acasandrei: “Embedded Face Detection System Implemented on LEON3 Microprocessor”, Master Thesis, Univ. Seville, 2011.

[8] 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.

[9] M. Yang, D. Kriegman, and N. Ahuja. “Detecting faces in images: a survey”, IEEE Trans. on PAMI, 24(1), p. 34-58, Jan 2002.