International Journal of Scientific Research & Engineering Technology (IJSET)
Vol.1, pp.44-47, 2014
Copyright IPCO-2014

Implementation Software and Hardware for Face Detection

Marwa Chouchene[#1], Haytham Bahri[#1], Fatma Ezahra Sayadi[#1], Mohamed Atri [#1] , Rached Tourki [#1]

[#] *Laboratory of Electronics and Microelectronics (EμE)*
Faculty of Sciences Monastir Monastir, Tunisia
[1]ch.marwa.84@gmail.com
[1]bahri.haythem@hotmail.com
[1]sayadi_fatma@yahoo.fr
[1]mohamed.atri@fsm.rnu.tn
[1]rached.tourki@fsm.rnu.tn

Abstract—**The analysis of the video represents a significant evolution in the field of computer • vision, especially in the field of video surveillance. This area seeks to detect the presence of some form, some edge ... this is the problem of detection. Among the objects that can be detected in this area face detection.**
Face detection in a fixed image without special hypothesis is a difficult problem due to the high variability of the shape to detect. Many techniques of detection and face recognition have been developed in recent years and many of which are very efficient. Among these methods, we find the method of Viola and Jones were studied in this work. The aim of our work is to study this method to implement on the CPU with C / C + +, then acceleration was presented with OpenCV. Subsequently, a second implementation in FPGA was presented

Keywords— **Face detection, method of Viola/Jones, C/C++, OpenCV, FPGA.**

I. Introduction

Face detection is a new computer technology that determines the locations and sizes of human faces in images. It detects facial features and ignores anything else, such as buildings, trees, bodies and any device other than the face and so on [4] .This technology is used in many fields such as biometrics for identification and recognition face, it is also used in video surveillance systems and many digital cameras use the latest face detection for autofocus.

Our work is divided into three parts: In the first part we present the method of Viola and Jones face detection.

Subsequently a second part deals with the face detection on C/C++. The acceleration with OpenCV is presented later, a second implementation in FPGA was presented. Finally, we conclude the paper.

II. Face Detection

A. Generality

The methods of face detection can be classified into four categories, [1]:

- Methods for a priori. These methods based on the rules used to model the knowledge of what makes a face. Typically, these rules represent relationships in facial eatures.
- Approaches invariant features. These approaches are based on structural features that exist even when the pose, the view or the illumination conditions vary, and use them to locate faces.
- Methods based models. Several standard models of faces are used to define a face model or models of facial features separately.
- The correlation between the image and the models is evaluated for the presence of face.
- Methods for learning. In contrast to methods based models, the models are learned from a set of training images which should allow characterizing the variability of the appearance of a face. These models are then used to learn the detection.

B. Method of Viola Jones

Viola et al. [2] presented a fast object detection algorithm based on a cascade simulated by simple descriptors called "Haar-like" descriptors. They can be calculated more efficiently by using an intermediate representation of the image called integral image. They also proposed a method for processing multi-stage classification which significantly reduces the duration of the audit while achieving almost the same accuracy compared to a classification algorithm single-phase which is certainly much slower and more complex.

Several references show the effectiveness of this system in terms of good detection, false alarms and speed, which is why we have chosen to implement it. We will detail later the detection method.

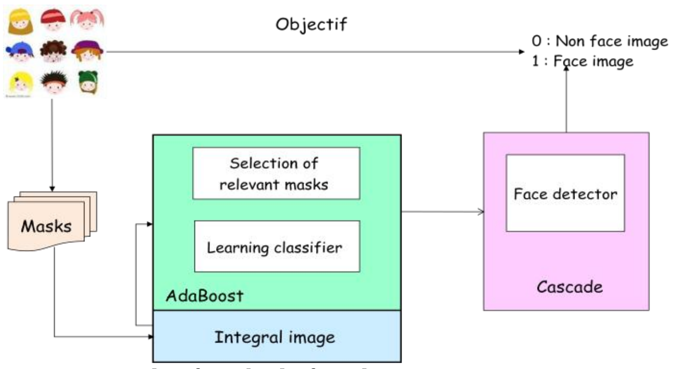

Fig. 1 Principle of method of Viola / Jones

1) Integral image: The integra *Integral image:* The integral image is defined as the summation of the pixel values of the original image. The value at any location (x, y) of the integral image is the sum of the image's pixels above and to the left of location (x, y).

2) Haar feature: Haar features are composed of either two or three rectangles. Face candidates are scanned and searched for Haar features of the current stage. The weight and size of each feature and the features themselves are generated using a machine learning algorithm from AdaBoost. The weights are constants generated by the learning algorithm. There are a variety of forms of features. . Each Haar feature has a value that is calculated by taking the area of each rectangle, multiplying each by their respective weights, and then summing the results. The area of each rectangle is easily found using the integral image. The coordinate of the any corner of a rectangle can be used to get the sum of all the pixels above and to the left of that location using the integral image. By using each corner of a rectangle, the area can be computed quickly.

3) *Classifier:* A Haar classifier uses the rectangle integral to calculate the value of a Haar feature. The Haar classifier multiplies the weight of each rectangle by its area and the results are added together. Several Haar classifiers compose a stage. A stage accumulator sums all the Haar classifier results in a stage and a stage comparator compares this summation with a stage threshold. The threshold is also a constant obtained from the AdaBoost algorithm. Each stage does not have a set number of Haar features. Depending on the parameters of the training data individual stages can have a varying number of Haar features.

4) Cascade: The cascade eliminates candidates by making stricter requirements in each stage with later stages being much more difficult for a candidate to pass. Candidates exit the cascade if they pass all stages or fail any stage. A face is detected if a candidate passes all stages.

III. Method of Viola Jones on C/C++

The Viola-Jones Object Detection Framework is a generic framework for object detection, which is particularly successful for face detection. In this assignment, we provide a simplified version of Viola-Jones face detection algorithm (figure 2). The simplified implementation does not include the training part of the framework. The cascade classifier in simplified implementation use pre-trained parameters for the cascade classifier. Although the provided code is not meant to be an optimal implementation, yet it provides reasonable detection rate for a wide range of input images.

So following the steps of the flowchart in figure 2, we find the following results (figure 3):

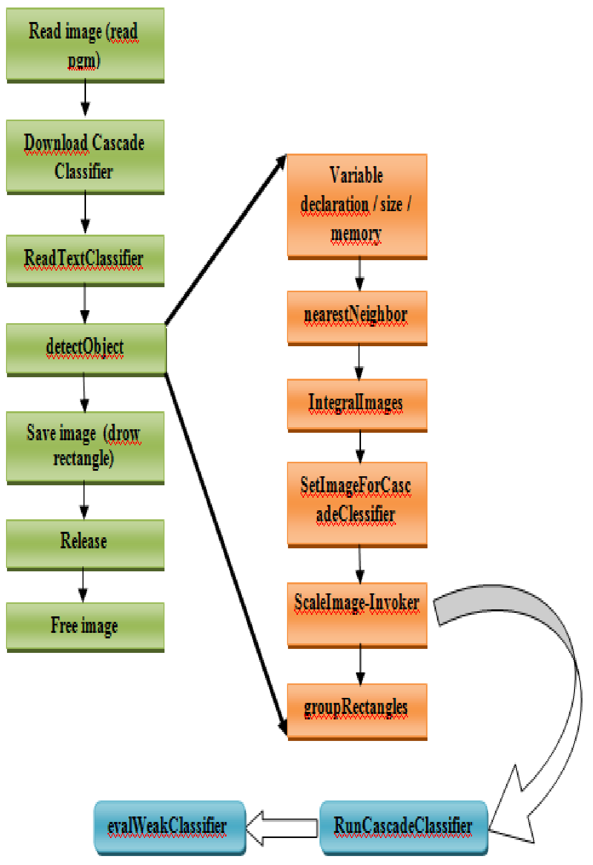

Fig. 2 Algorithm as implemented on C/C++

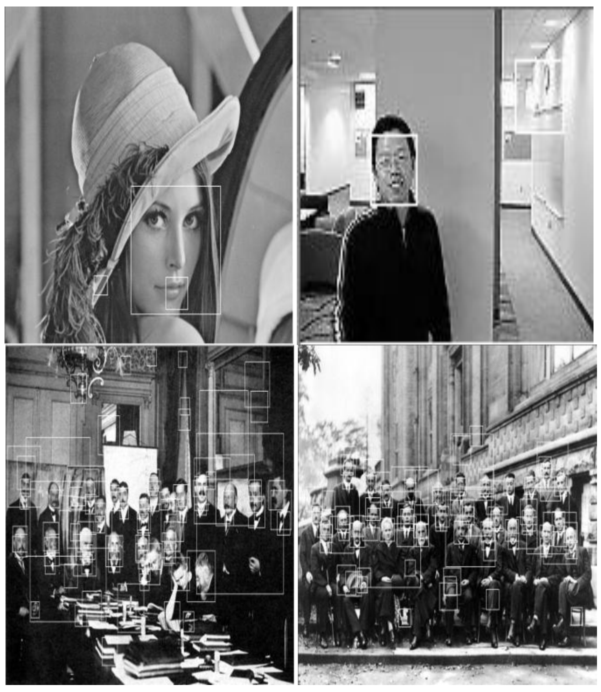

Fig. 2 Result of image processing on C/C++

We test our algorithm on the windows platform. Software environment includes Microsoft Visual Studio 2008, Hardware environment includes a consumer-level PC with an Intel Core i5, M560 2.6GHz CPU, 4G RAM.

In order to evaluate our face detection implementation, we measured the time taken to carry out this treatment (figure 4).

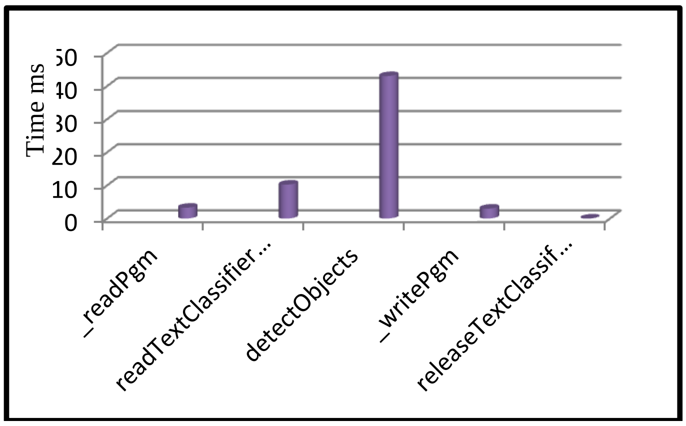

Fig. 4 Measured time of face detect on C/C++

Among the methods to calculate the execution time of a function or a procedure, there is a profiler Visual C + +. It provides statistics and very precise calculations. Whose percentages execution time are extracted with respect to time of the main function (main). With exclusive time is the time spent in the function, while the inclusive time is the time spent in the function and its children.

The results obtained with this profiler are given in the following table:

TABLE I
EXECUTION TIME OF FACE DETECTION

Function name	%Application Inclusive Time	%Application Exclusive Time	N°= Call
Main	99,7	40,09	1
readPgm	3,28	0,00	1
readTextClassifier	10,26	0,37	1
detectObject	43,05	0,00	1
writePgm	3,00	0,21	1
releaseTextClassifier	0,01	0,00	1

We note that the function Objectdetect is the most critical time of execution, to accelerate our treatment, we will make a second implementation using OpenCV library

IV. Acceleration by OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

The figure below shows the flowchart used to accelerate our treatment.

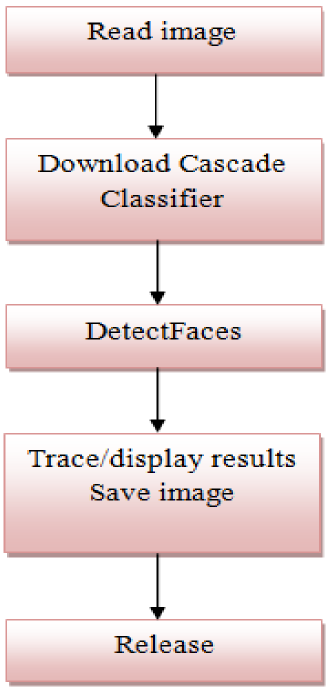

Fig. 5 Algorithm as implemented on OpenCV

This step consists in detecting the face by applying the method based on Haar descriptors, which will start the steps described above. This method is defined in the library OpenCV in C as "CVHaarDetectObjects.

After compiling our application (Face Detection) respectively on C/C++ and OpenCV, the results obtained are presented in Table:

TABLE II
THE TIME-CONSUMING COMPARISON BETWEEN C/C++ BASED AND OPENCV BASED ALGORITHMS

	OpenCV Time (s)	C/C++ Time (s)
Read image	0,003	0,011
Download Cascade Classifier	0,06	0,036
Detection	0,064	0,11
Display results	0,001	0,01
Total	0,128	0,167

V. Implementation on FPGA

FPGA can achieve extremely high performance in many applications in spite of its low operational frequency.

In this context we will define a methodology to implement automatically and optimize algorithms for image processing in complex electronic systems.

We conducted a second implementation in VHDL on embedded platform to reduce the execution time. The previous algorithm was translated into VHDL, it has an entity consisting of 5 input ports.

The architecture of this algorithm contains 7 components that contain the values of the classification functions of the AdaBoost learning algorithm. Our architecture also contains 7 process.

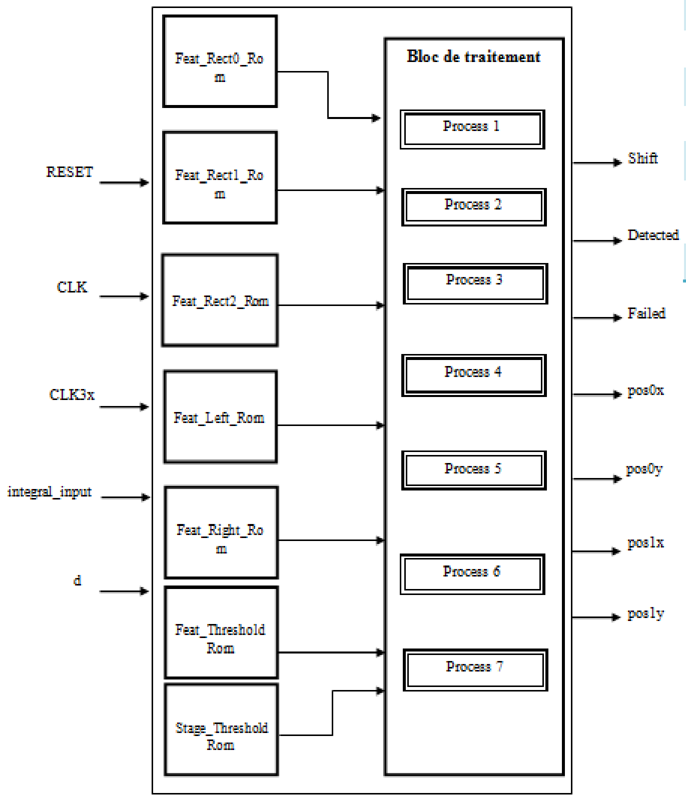

Fig. 6 Architecture of the block of face detection in VHDL

After the simulation (figure 6), the next step is to synthesize the hardware description for a component performing the desired functions, using concrete logic elements.

For the development of our work, we choose to use the synthesizer "Xilinx ISE .9" as a synthesis tool.

To realize the hardware implementation, much technology architecture synthesis is implemented. The synthesis phase of our application was assessed on FPGA Spartan3A: type XC3S200A, package FG320 and degree speed -5.

Completion of the synthesis gives us the following results (Table III)

TABLE IIII
SPACE OCCUPIED IN FPGA

	Used	*Available*	*Percentage*
Number of Slice Registers	*7830*	*1792*	*436%*
Number of Slice Flip Flops	*6978*	*3584*	*194%*
Number of LUTs	*8645*	*3584*	*241%*
Number of bonded IOBs	*55*	*248*	*22%*
Number of BRAMs	*25*	*16*	*156%*
Number of MULT18X18SIOs	*3*	*16*	*18%*
Number of GCLKs	*3*	*24*	*12%*

The parameters of our application are:
The period of the clock: 9.675ns (frequency: 103.363MHz)
t = number of cycles x period of a cycle = 2590 x 9675 x 10 -9 = 25058.25 ns.
The FPGA implementation also uses a custom circuit to return first compared C++ and OpenCV

VI. Conclusions

In this work, we reviewed one of the most basic methods for face detection. This method was a variant of the popular Viola & Jones method based on rectangular haar-like features, as described by Viola and Jones.

Different implementations were made: one using C / C + +, the second on OpenCV , and the latter on FPGA to accelerate processing.

References

[1] M-H Yang, D. J. Kriegman et N. Ahuja, *Detecting Faces in Images : A Survey*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34-58, Jan. 2002.

[2] P. Viola et M. Jones, *Rapid object detection using boosted cascade of simple features*, Proceedings IEEE Conf. on Computer Vision and Pattern Recognition 2001.