Advanced Hardware Real Time Face Detector

Petronel BIGIOI, *Senior Member, IEEE*, Corneliu ZAHARIA and Peter CORCORAN, *Fellow, IEEE*

***Abstract*—Face detection and tracking can be considered an enabling technology for a range of image enhancements technologies, authentications and advanced UIs in handheld devices. Several modes of implementing a face detection algorithm in hardware are explored and a hybrid approach is found to offer the best trade-off between physical hardware resources and computational software resources. This uses simpler weak classifier templates in hardware, using these to determine the more complex set of classifiers which are eventually applied by the software engine.**

I. INTRODUCTION

With the proliferation of image acquisition devices face detection is now an expected feature. The further applications of face detection include improvement of focus and acquired image, affective computing, user authentication and biometrics. In this paper we present a real time face detection technology specifically designed to retain the advantages of both hardware and software implementations. AHFD (Advanced Hardware Face Detector) has been designed to combine the speed and low computational requirements of hardware implementations with the programmability of software detectors.

II. USES OF FACE DETECTION IN CONSUMER DEVICES

A. Improved Capture Quality

Having the ability to detect human faces in a scene, during the preview, image quality can be greatly enhanced. Facial location can be used to improve both the efficiency and/or quality of auto-focus, auto-exposure, and auto-white balance algorithms. As faces are the most important element of most photos and movies it is important that they are captured to best effect and, in particular, un-natural skin tone reproduction is avoided. The faster that face detection is performed the more time available for additional image processing based on the face data. For instance, if the face detection can be achieved in less than a single frame interval then auto-focus algorithms can be greatly improved in terms of both quality, i.e. focus on the right object or FOI, but also in terms of convergence speed, typically by a factor of 3 times when compared with classic auto-focus algorithms.

B. Improved Capture Experience

We all understand the challenge posed by capturing a good picture when you have single or multiple human faces: persons not looking at the camera, persons blinking and the right amount of smile. Having specific smile and/or blink detectors operating on top of a fast and robust face detector can simplify and improve the image capturing experience. A range of such features can be found in many of the latest digital cameras.

C. 3D Displays

3D displays for handheld devices have come into vogue recently. Because it would not be practical to use 3D glasses with a handheld, such displays are typically auto-stereoscopic where the 3D effect is optimized for a certain distance from the display. However, these displays require that the user has to seek, find and maintain the proper pose in order to enjoy 3D content correctly. And if the optimal position is lost, the effect is rather unpleasing, causing nausea for some users. There is much active research by industry into developing a programmable parallax barrier. However for such a system to function properly the capability to have real time face and eye detection and tracking at 30 fps or more is needed to control the parallax barrier in a real-time feedback loop. The main technology enabler is a reliable face detector which can acquire a face in a fraction of the time available during a single image frame (i.e. 30 ms for a 30 fps image stream).

D. Authentication and Profile Switching

Using face detection in combination with face recognition consumer devices can be automatically brought back from power saving modes and a user can be authenticated. For certain shared devices, such as gaming consoles, the same information can be used for automatic user profile switching and to allow for personal customizations and settings to be automatically applied according to the user detected.

E. Enhanced User Interfaces

Knowing with high degree of precision where the eyes of the player are and having the ability to track facial features and emotions, the game designers can provide enhanced character representation (e.g. animated avatar) as well as enhanced content presentation. User emotions can be further used as feedback to the machine to dynamically set the difficulty of the levels and/or decide the content presented to the user. Again, the main enabler for such technologies is a fast and robust face detector.

III. IMPLEMENTATION

A. State of the Art Survey

Various types of algorithms have been attempted and described in the literature for face detection and tracking. The better known algorithms in terms of performance are described by Rowley, Baluja & Kanade in [1], Schneiderman & Kanade in [2] and Viola & Jones in [3], [4]. Rowley, Baluja & Kanade describe a neural network-based upright frontal face detection system whereby a neural network examines small windows of an image, and decides whether each window contains a face. Schneiderman & Kanade describe a statistical method for generic 3D object detection, with direct application to human faces and cars. Their work is based on statistical representation of object appearance using a product of histograms. Most importantly the first practical real-time algorithm to provide real time object detection was described by Viola & Jones in 2001 [3], [4].

All of those algorithms have two parts: an offline part, or training part, and an online part, or the detection part. The offline part or the training provides the description of the object to be matched against the input frame(s) to detect the object of interest. A set of positive examples (e.g. faces) properly aligned and cropped, together with a set of negative examples (e.g. non faces area) are fed into the training algorithm. The training algorithm is one of a neural network [1], statistical wavelet decomposition [2], or local Haar features based [3], [4]. After the training is completed, the result of this step is the "signature" of the object, such as a face, at a set of given resolutions (e.g. 22x22 pixels), also known as object or face template. In other words, the training is done in order to extract a set of common denominator features for the searched object, it is done offline, so the execution speed and complexity of the algorithms are not too important.

Now the second part of any object detection algorithm is the online part. During the online part, the object templates are compared against the input picture/frame, typically using a sliding window approach, searching for both the position and scale to find objects. The template matching algorithm is applied either directly on the input image (by down-sampling the input image to obtain the scales) [1], wavelet space [2], or integral image space [3], [4]. This part is computationally intensive and depending on the specific algorithm needs to be applied multiple times for different scales and poses of the face.

B. *Algorithm Partitioning and Flow*

The main idea of our implementation is to offload the processing power required for template matching from the software domain (CPU) into dedicated hardware units specialized in template matching (TMM – Template Matching Machine). The overall block schematic is presented in Fig 1.

Before arriving at our final optimized hybrid implementation, the underlying algorithm went through a number of iterations. In the initial version, we have tried to implement the template-matching algorithm using a full and complete definition of the templates, stored in a local SRAM memory close to the TMMs. This required a large dedicated SRAM memory for storing chains of classifiers as large as 120KB each. In a second approach we tried loading the classification chains into main memory, but this approach had its own major disadvantage as the memory bus bandwidth to achieve 30 frames per second processing rate was in excess of 2GB/second and impractical on most state-of-art handheld devices.

In the final implementation, we have understood that, while it is important to have the templates close to the TMMs, it is also required that the definition of each classifier template should be optimal from a size point of view. This led to the conclusion that the TMMs should only perform a complex scanning with a weak and small in size (less than 5KB per chain) set of classifiers, to determine the scale and pose of the face. The SW part of the algorithm can then be selected to complete the filtering using a more complex and complete set of templates.

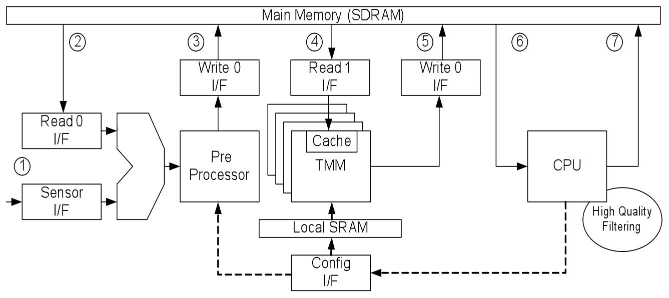

Fig 1: Advanced Hardware Face Detector Block Diagram

The algorithm flow for our implementation is shown in Fig 1: the input frame can be sourced either directly from the sensor (1) or from the main memory (2). The image data is processed by a pre-processing engine (3) that ensures one clock per pixel. This pre-processor module, writes in the main memory a conditioned copy of the input frame (e.g. QVGA resolution, YCrCb 4:2:0) together with some statistics and optional maps that allow the SW parts to execute faster. A number of synthesis configurable TMM units will receive the Y data together with some other information obtained in the previous step (4), and perform a pre-scanning of the input frame using a set of week object/face classifiers. At the end of the scanning phase, the TMM engine(s) will write (5), into the main memory a list of suspected areas (coordinates, scale, angle and associated template). The CPU (6), will receive this list, together with the input frame and statistics/maps, and complete the filtering; writing in the main memory the list of detected faces (7). The final filtering uses a set of complex templates and a number of other content specific filters.

Using this approach, the CPU will have to perform filtering on only a few tens of windows of interest versus over 40,000 windows using a SW-only optimized implementation.

C. *Performance Indicators*

Examples of performance improvement will be given, including comparisions with various software implementations.

REFERENCES

[1] Rowley, Baluja, and Kanade: Neural Network-Based Face Detection, PAMI, January 1998

[2] H. Schneiderman, T. Kanade. "A Statistical Method for 3D Object Detection Applied to Faces and Cars". IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2000

[3] Viola P, Jones M.: Robust Real-time object detection. IEEE ICCV Workshop on Statistical and Computational Theories of Vision 2001.

[4] Viola P, Jones M.: Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2001