Expert Systems With Applications 194 (2022) 116505

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Real-time embedded eye detection system

Camilo A. Ruiz-Beltrán [a], Adrián Romero-Garcés [a], Martín González [a], Antonio Sánchez Pedraza [b], Juan A. Rodríguez-Fernández [a], Antonio Bandera [a,*]

[a] Department of Electronic Technology, University of Malaga, Spain
[b] LDA Audio-Tech, Andalusia Technology Park, Málaga, Spain

ARTICLE INFO

Keywords:
Eye detection
Viola–Jones algorithm
All programmable System-on-Chip

ABSTRACT

The detection of a person's eyes is a basic task in applications as important as iris recognition in biometric identification or fatigue detection in driving assistance systems. Current commercial and research systems use software frameworks that require a dedicated computer, whose power consumption, size and price are significantly large. This paper presents a hardware-based embedded solution for eye detection in real-time. From an algorithmic point-of-view, the popular Viola–Jones approach has been redesigned to enable highly parallel, single-pass image-processing implementation. Synthesized and implemented in an All-Programmable System-on-Chip (AP SoC), this proposal allows us to process more than 88 frames per second (fps), taking the classifier less than 2 ms per image. Experimental validation has been successfully addressed in an iris recognition system that works with walking subjects. In this case, the prototype module includes a CMOS digital imaging sensor providing 16 Mpixels images, and it outputs a stream of detected eyes as 640 × 480 images. Experiments for determining the accuracy of the proposed system in terms of eye detection are performed in the CASIA-Iris-distance V4 database. Significantly, they show that the accuracy in terms of eye detection is 100%.

1. Introduction

Relevant tasks such as iris and face recognition, gaze tracking, behaviour and expression interpretation, or detection of driver fatigue, are based on the previous estimation of the position of the eyes of the person. Depending on the application, this estimation must be performed in high-resolution and distortion-free images and at a high frame rate. For instance, this is the case when we need to capture iris images while the person is in motion. With a need to have at least a 70-pixel radius for the iris (Daugman, 2004), the system will need to handle large images if it is to capture a field of view of a certain size. The shallow depth of field of this system will also require a large number of images per second to be captured if one or two iris images with the required level of contrast are to be available. In such a scenario, the eye detection task cannot be underestimated (Ji et al., 2005). As it has been pointed out, the initial eye localization stage can be the most consuming stage on the whole iris recognition framework (Kumar et al., 2018).

Given the importance of this task, several detection approaches have been proposed. All these approaches take into account that a person's eyes have very significant distinguishing attributes. Thus, the behaviour of the eye under infrared (IR) illumination (e.g. the red-eye effect) has been the basis for the development of active methods, in which different light sources and/or cameras are used. For instance, Quan et al. (2013) use two groups of lights which all lie on the camera axis. One of these clusters utilizes 850 nm IR LED and the other 950 nm IR LED. The approach matches the two images and, based on the property of the retina to reflect only about 40 per cent of the incident light at 950 nm while about 90 per cent at 850 nm, the differences in brightness between the two images allow the pupils of the eyes to be located. Other approaches use passive lighting, and rely on the construction of a model of the eye. Then, they use this model to locate the eyes in the input image. Several authors have suggested extending this model to include facial landmarks, but this can be a problem when the image include partial faces (Nsaif et al., 2021). With the advances in computing ability, eye detection task has been dominated by the use of deep learning. The manual definition of the model can be now automated using deep learning via condensed data and frequent training (Krafka et al., 2016). Although deep convolutional neural networks (CNNs) have given state-of-art results when addressing many computer vision tasks, their use in eye search is limited to a certain extent. Nsaif et al. (2021) pointed out that this can be due to a lack

* Corresponding author.
E-mail addresses: ruizbeltran@gmail.es (C.A. Ruiz-Beltrán), argarces@uma.es (A. Romero-Garcés), martin@uma.es (M. González), aspedraza@gmail.com (A.S. Pedraza), jarodriguez@uma.es (J.A. Rodríguez-Fernández), ajbandera@uma.es (A. Bandera).

https://doi.org/10.1016/j.eswa.2022.116505
Received 10 July 2021; Received in revised form 30 December 2021; Accepted 1 January 2022
Available online 19 January 2022

of large data sets. However, the major problem is related with the efficiency of these approaches. As aforementioned, several applications such as iris recognition require very high resolution, and this demands large images if we want to cover the whole face. The use of CNNs requires then a large amount of computer resources and time if the approach must perform a global search in the image. The solution is to include a quick and effective approach for proposing candidate regions such that only these regions will be fed into the CNNs (Li & Fu, 2018). Eye attributes are used for guiding this preliminary step. For instance, Li and Fu (2018) make use of the pupil and iris being darker than other parts of the eye. Thus, the locations of the local extreme points in the image are more likely to be the rough centre positions of the eyes. Choosing the top N extreme points, they ensure that the candidate regions can completely cover the eye region.

This paper presents a highly optimized eye detector that works in real time over high-definition videos. The proposal is stated over two major premises. On the one hand, we believe that the geometry of the eye is sufficiently distinguishable in the image that algorithms already proposed in the literature have achieved high performance. This is the case for the popular Viola–Jones approach (Viola & Jones, 2001). For instance, Raja et al. (2019) report a 100% of accuracy rate when using this approach as eye detector with the CASIA-Iris-distance V4 database. On the other hand, we consider that a highly parallel, single-pass image-processing implementation is possible for the characterization and classification processes. To achieve this result, a parallelized version of the Viola–Jones algorithm is proposed and synthesized and implemented in the programmable logic part of an All-Programmable System-on-Chip (AP SoC). In this implementation, both feature extraction and classification are carried out in parallel. Instead of trying to reduce the number of weak classifiers to be evaluated, our proposal will run all the classifiers simultaneously. The result is a classifier whose central core can process up to 752 fps. When a final labelling stage is added, this result is reduced to 88 fps.

The rest of the paper is organized as follows: Section 2 discusses related work on passive illumination-based eye detection. This Section presents the basis of the Viola–Jones approach. The proposed approach is described in Section 3. Section 4 provides details about our hardware implementation. Experimental results are presented in Section 5. Finally, conclusions and future work are drawn in Section 6.

2. Related work

Object detection is an important research topic within the field of computer vision. From a perspective considered nowadays as conventional, this task is approached by estimating features and adjusting classifiers, in a more or less supervised way, through a certain training process. In this way, these algorithms usually have three parts: determining the type of detection window, the selection of the features to be estimated, and the construction of the classifier. The first step, the selection of the detection window, will determine the scale or size of the object, in our case the eye, to be detected. For the second step, there are different options, among which we find the histogram of oriented gradients (HOG), the local binary patterns (LBP), etc. (Saif et al., 2017). As for the classifier, many object detection systems use a decision tree or Support Vector Machine (SVM) (Benrachou et al., 2015). Moreover, depending on the features used, these methods are based on the appearance or geometry of the eye, or on the use of a model of the eye itself. In the latter case, it is a matter of defining this model of the eye, with the classifier being responsible for searching the image for an appropriate correspondence between the image data (regions or edges) and the defined model. The proposal can combine some of these techniques. For instance, Kawaguchi and Rizon (2003) detects the face region in the image and then extracts intensity valleys from this region. Using a feature template and a separability filter, they extract iris candidates from these valleys. And finally, using the costs for pairs of iris candidates, the algorithm selects a pair of iris candidates

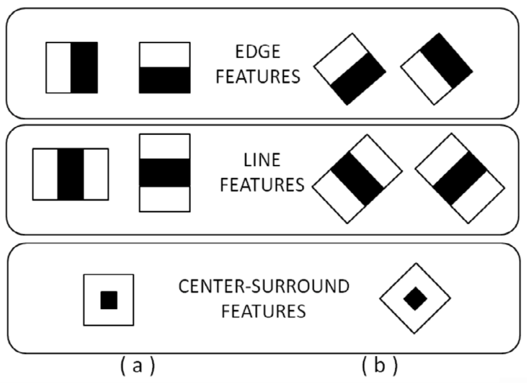

Fig. 1. Examples of feature prototypes of simple Haar-like and centre-surround features.

corresponding to the irises. These costs are computed by using Hough transform, separability filter and template matching.

Since a couple of decades, eye detectors using machine learning techniques have become much more common. These methods can be divided into two groups. The first involves initial feature extraction following a conventional scheme, followed by a cascading classifier. Thus, for example, D'Orazio et al. (2004) propose an initial geometric feature extraction followed by a neural classifier to detect eye regions. With the popularity of deep learning algorithms (Xie et al., 2017), some researchers have proposed using convolution neural network (CNN) to train eye detectors, which is the second group. The scheme includes feature extraction and feature classification in the same network. Deep learning-based methods have demonstrated high robustness and detection accuracy compared to traditional methods. However, efficiency remains an issue (Li & Fu, 2018). Facial images are usually larger than 640 × 480 pixels, and this requires a large number of computational resources when the CNN has to perform a global search on the image. The solution proposed by some authors is to include a fast and efficient method to propose candidate regions, so that only selected candidate regions are fed to the CNN. For instance, the Faster R-CNN (Ren et al., 2017) can be divided into three sections: a feature extractor, a region proposal network (RPN), and a classifier. The RPN and the feature extractor of the Faster R-CNN detector share the same convolution layers. They are encoded in a network of fully convolutional layers that is used to identify candidate regions with a wide range of aspect ratios and scales. Then, the classifier is employed to filter the set of candidates. Nsaif et al. (2021) combine this Faster R-CNN with Gabor filters and the naive Bayes model for determining the final eye regions.

2.1. The Viola–Jones method and implementation improvements

Using the Haar-like features and a variant of the AdaBoost procedure, Viola and Jones (2001) proposed a fast and robust object detection algorithm, which was successfully applied to this scenario (Thongleng & Kaewapichai, 2018). The Haar-like features are a reminiscent of the Haar basis function and can detect edges or line features (Fig. 1). Subsequent work has augmented the original set of features to include rotated versions (Fig. 1b) (Lienhart & Maydt, 2002). On the other hand, AdaBoost is a supervised learning scheme employed to boost the classification performance of a simple learning algorithm. The proposal from Viola and Jones works as follows: first, the sum of pixel values within the white or black regions in each Haar-like feature are computed respectively, and then the difference of the weighted sum of these regions is calculated. The so-called Integral Images are employed to speed up the sum of pixel values within a rectangle (or a rotated one) (Viola & Jones, 2001). Once each difference is computed, it is compared with a predefined threshold. If the difference exceeds the threshold

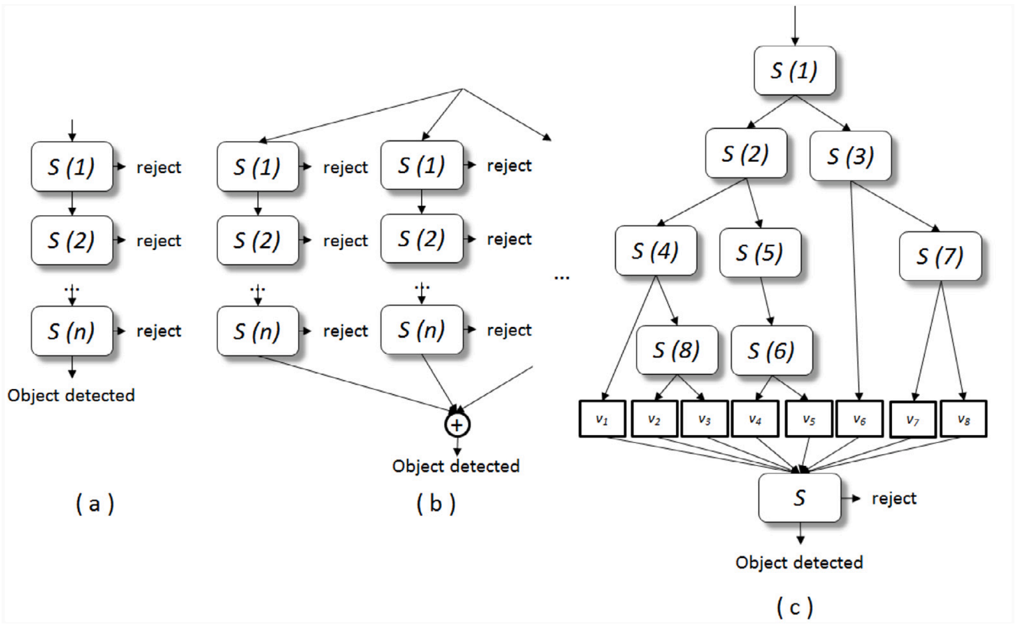

Fig. 2. Different classifier structures for encoding the Viola–Jones approach: (a) cascade classifier; (b) multiple cascade classifiers; and (c) tree classifier. The scheme at (c) differs from the original proposal by Lienhart and Maydt (2002). Each weak classifier ($S(1), S(2)\ldots$) implies the evaluation of one specific feature and uses a threshold to determine if the next branch to evaluate is the left or the right one. Several branches are not associated to new evaluations, but to score values ($v_1, v_2, \ldots$ on the figure). The result of the tree is one score value that is then compared with a threshold (classifier S). This last classifier determines the detection result.

value, the weak classifier outputs true. If not, the weak classifier outputs false. These weak classifiers are then combined to generate a boosted strong classifier. The cascade of classifiers is originally a sequence of stage (weak) classifiers (see Fig. 2). The cascade structure reduces processing times as the classifier is trained to reject non-objects in the first stages, spending more time on promising object-like images. Finally, the procedure is applied on a rectangle-shaped region (sub-window), which scans the whole input image. When all sub-windows in one image are evaluated, the whole procedure can be repeated using a different sub-window size (multiscale detection).

As it was pointed out by Lienhart et al. (2003), there are significant advantages on the use of decision trees instead of sequential cascades or multiple cascades. In this implementation, the classifier is encoded using several sequential stages, each one of them consists of multiple decision trees. Trees are arranged according to the scheme at Fig. 2c. Thus, each tree is composed by eight nodes, which evaluate one Haar-like feature and use a threshold value for directing the execution to the right or left branch. When one of these branches have associated a specific score value (see Fig. 2c), the execution on the tree finishes and the value is provided. When the chosen branch redirects to a new node, the evaluation and thresholding procedures are repeated. In each stage it is typical to find one to three trees. Score values obtained from all trees on the stage are added and the result is compared with a stage threshold value. According to the result of this comparison the window is rejected or passes for evaluation to the next stage.

Much work has been performed in attempts to speed up the Viola–Jones approach. Software solutions use optimized implementations on OpenCV (Open Source Computer Vision Library), the library for image processing originally developed by Intel; or have translated the algorithms to GPUs (Fredj et al., 2020; Jain & Patel, 2016). However, the optimized code does not provide good results on embedded platforms (Kim et al., 2015). An alternative to exploit parallelization is to use a hardware approach that speeds up the computations using an application-specific design. To increase the processing speed, Theocharides et al. (2006) encoded the proposal on an ASIC platform. They use a grid array processor as the structure of their architecture, the so-called CDTU (Collection and Data Transfer Unit), for exploiting parallelism. The simulation results in their paper reported that they obtained a rough estimate of 52 fps targeting 500 MHz clock cycle. However, the proposal requires a large processor array, which is identical to the input image size. For reducing development time and cost, the popular alternative to ASIC are FPGAs. Wei et al. (2004) proposed an architecture on a Xilinx Virtex™-II FPGA for simulating only a part of the whole algorithm. They report rates up to 15 fps at 91 MHz for small images with 120 × 120 pixels. They scale the input image and use fixed-point expressions to achieve fast processing with less circuit area. They only use non-rotated features and parallelize the estimation of all the features obtained from AdaBoost training. The synthesized detector divides the cascade into three sequential stages containing 9, 16, and 200 Haar-like features respectively. The parallelization of the computations within the stages that compose the single cascade (Fig. 2a) has been repeated by other subsequent work (Kim et al., 2015; Lai et al., 2007). Taking into consideration that only former weak classifiers on the cascade are almost always executed, Hiromoto et al. (2007) proposed a partially parallel approach. Parallel modules are assigned to these former stages while the latter ones are mapped into sequential modules. The number of former classifiers is determined in consideration of a trade-off for circuit area and processing performance. The main disadvantage is that the separation of the parallel and sequential stages requires additional hardware resources as it is necessary to hold the integral image values for current subwindow to process in sequential stages while parallel stages compute a new subwindow. Other solution to increase the computation speed is to simplify the classifier. Thus, Lai et al. (2007) proposed a architecture which uses a piped register module for computing the integral image and only employs 52 classifiers in a single stage. It can achieve 143 fps detection using images of 640 × 480 pixels. But the performance is significantly poorer than the one provided by the OpenCV's implementation (lower detection rate and higher false alarm rate). Another implementation that sacrifices the performance was reported by Yang et al. (2006). This low-cost architecture was implemented on an inexpensive ALTERA Cyclone-II FPGA. Their architecture achieves 13 fps but the detection rate falls to about 75% (in the original method by Viola and Jones (2001) it is more than 90%). To avoid the large number of resources needed to synthesize and implement the integral image computing, Gao and Lu (2008) suggested to only implement the classifiers in the

FPGA, computing this integral image in a host microprocessor. Nair et al. (2005) described a solution for people detection based on the Microblaze softcore processor from Xilinx. They achieve 2.5 fps for image sizes of 216 × 288.

3. The proposed scheme

The structure of the proposed classifier for eye detector is shown in Fig. 3. Initially, we can consider that the scheme is the same than the one proposed by other researchers (Lai et al., 2007; Wei et al., 2004): after estimating the integral and tilted integral images, and the standard deviation (for brightness normalization), for a detection window (all these elements are calculated as the input image is read), the whole set of required Haar-like features are computed in parallel. A unique scale is considered here, and the evaluated features are associated to this specific detection window. Then, all stages are executed in parallel, and the results are combined for obtained the tentative positions of the eyes in the image. There are however two major differences with respect to previous contributions. Firstly, the structure does not resemble a cascade classifier, but to the decision trees shown in Fig. 2c. This scheme has the advantage of reducing the number of Haar-like features to evaluate in the whole classifier. On the contrary, it manages more parameters (left and right values) than the single or multiple cascades. The relative complexity of this scheme is possibly the reason why it has not been the chosen option for hardware implementation. Thus, previous approaches described at Section 2 choose as Viola–Jones model the cascade of classifiers and put the emphasis on adding more features/weak classifiers for increasing the performance. As commented in this Section, when they reduce the number of features, the performance rate also decreases. Secondly, the responses of all nodes in a stage are grouped into index values, which allows us to search the final result using Lookup tables (LUTs). This is the basis of the parallelization of the decision trees and is detailed below. The score value obtained from the LUT is compared with a threshold. The comparison returns a Boolean value, being 1 if the evaluated window is a candidate region to be an eye. In the structure in Fig. 2c, this encodes the S classifier. As aforementioned, this new scheme has made it possible to parallelize the execution of decision trees, reducing the time but also the resources needed to synthesize and implement them.

As Fig. 3 shows, the structure implemented in our classifier consists of five stages. Each stage includes a maximum of three trees, each one of them including eight nodes. Each of these nodes evaluates a Haar-like feature. Briefly, the idea is to do not travel the trees for obtaining the final score value but to always evaluate the eight nodes, to obtain the eight 'greater/lesser than' decisions, and to use these eight Boolean values for assigning a specific score value to the tree. Fig. 4 schematizes how the score value is obtained. Each stage has an internal LUT, which is addressed with a 10-bit vector (2 bits for encoding the tree and 8 bits associated to the evaluated nodes). As we have three trees per stage, the LUTs have a size of 256 × 3 = 768 values. The computation of the stage value is encoded in three search processes in the LUT and a final summation. This allows us to compute the final value in only two steps: one for evaluating all nodes and an additional one for assigning a score value to the obtained feature vectors. Finally, the result value is compared with a stage threshold. Thus, the stage evaluation does not provide a score value but a Boolean one: 1 for positive evaluation of the detection window and 0 for rejection.

The classification ends when all detection windows are evaluated. As aforementioned, in our proposal, a single scale is used and then the classification results can be summarized within a matrix whose size is equal to $(M - m + 1) \times (N - n + 1)$, being $M \times N$ the size of the input image and $m \times n$ the size of the evaluated detection window. Due to the pixel-to-pixel shift of the window, it is typical that positive detection values result on clouds of points on the evaluation matrix around a single, real presence of the desired object. Similar to the post-processing step employed on OpenCV, we use a masking procedure to merge points associated to positive detection values, filtering the evaluation matrix (the Detection group module in Fig. 3). Briefly, the values within a given window are added and thresholded with a specific value to determine whether it is a positive detection (1) or not (0). Fig. 5 shows the final cloud of positive detection values obtained after filtering the output of a classification process.

Our aim is to achieve the same accuracy rate that the software proposal implemented in the OpenCV library (Raja et al., 2019). Thus, the internal parameters of our design are obtained from training tools based on the ones employed by the OpenCV framework. This training phase is then encoded in C++ language and is executed off-line on a personal computer. The only significant difference with respect to the OpenCV implementation is the use of 16.16 fixed-point representation for encoding those parameters that do not need more fine precision (threshold values). The training phase is bounded to manage our fixed structure (five stages, with a maximum of three trees per stage, and with eight nodes per tree). It could be modified if required, but it provides very good result for eye detection. For instance, this structure allows the system to obtain a 100% success ratio when tested in the CASIA-Iris-distance V4 database.

4. Hardware implementation

This Section describes the effort doing for optimizing the speed of the eye detector. The classifier realization is done in a Ultrascale+ XCZU4EV micromodule from Trenz and the synthesis and implementation is performed by Xilinx Vivado. Fig. 6 shows the logical architecture of our first implementation. As described below, it has been modified for improving the efficiency and resource consumption, but this image provides a clearer view of how it works. The integration of the classifier within a complete framework is described in Section 5.

The classifier takes the image from the RAM and feeds it into the data stream through Video Direct Memory Access (VDMA), then the axis_Haar_Preprocess core generates an integral image, a tilted integral image, and the standard deviation of the detection window. After that the data is fed simultaneously to the five axes_haar_stage cores. Each stage provides a Boolean result, all of them must be true in order to mark this detection window as true. The accuracy is improved by means of the axis_detection_group core that groups the true detections within a window using a threshold. Finally, the data is stored on the RAM using another VDMA.

Most of the modules of the classifier core are synthesized using High-level synthesis (HLS). In Fig. 6, they are marked with the Vivado HLS logo inside. The HLS tool provides directives to allocate resources, define the latency and determine how to pipeline the modules. One of the major limitations is the memory allocation. To address this, the Ultrascale+ XCZU4EV offers 128 block RAM (BRAM) modules and 48 Ultra RAM (URAM) modules. Each BRAM module has 18 Kb of memory, and each URAM has 288 Kb. Nevertheless, the election of each module for our application entails advantages and drawbacks. Thus, using BRAM allows us a faster execution at the cost of more resources, conversely using URAM reduces resource usage at the cost of more latency. In order to optimize the efficiency, the scheme presented in Fig. 6 has been modified. Basically, our aim was to merge all the cores of the classifier into a single one, allowing the synthesis and implementation tool to further optimize and share resources. Firstly, the five Haar stages were merged into only one core. This allows us to remove the haarStagesResultRead_0 and the axis_broadcaster_0 cores from the design. This resulted in significant resource savings and reduced complexity. In a second step, the axis_Haar_Preprocess, the stages, and the axis_detection_group cores have been merged. This saves even more resources. Finally, the functionality of the axis_detection_group core was modified so that, instead of giving as a result 1 or 0, the result is the number of points grouped in each window. This will allow the

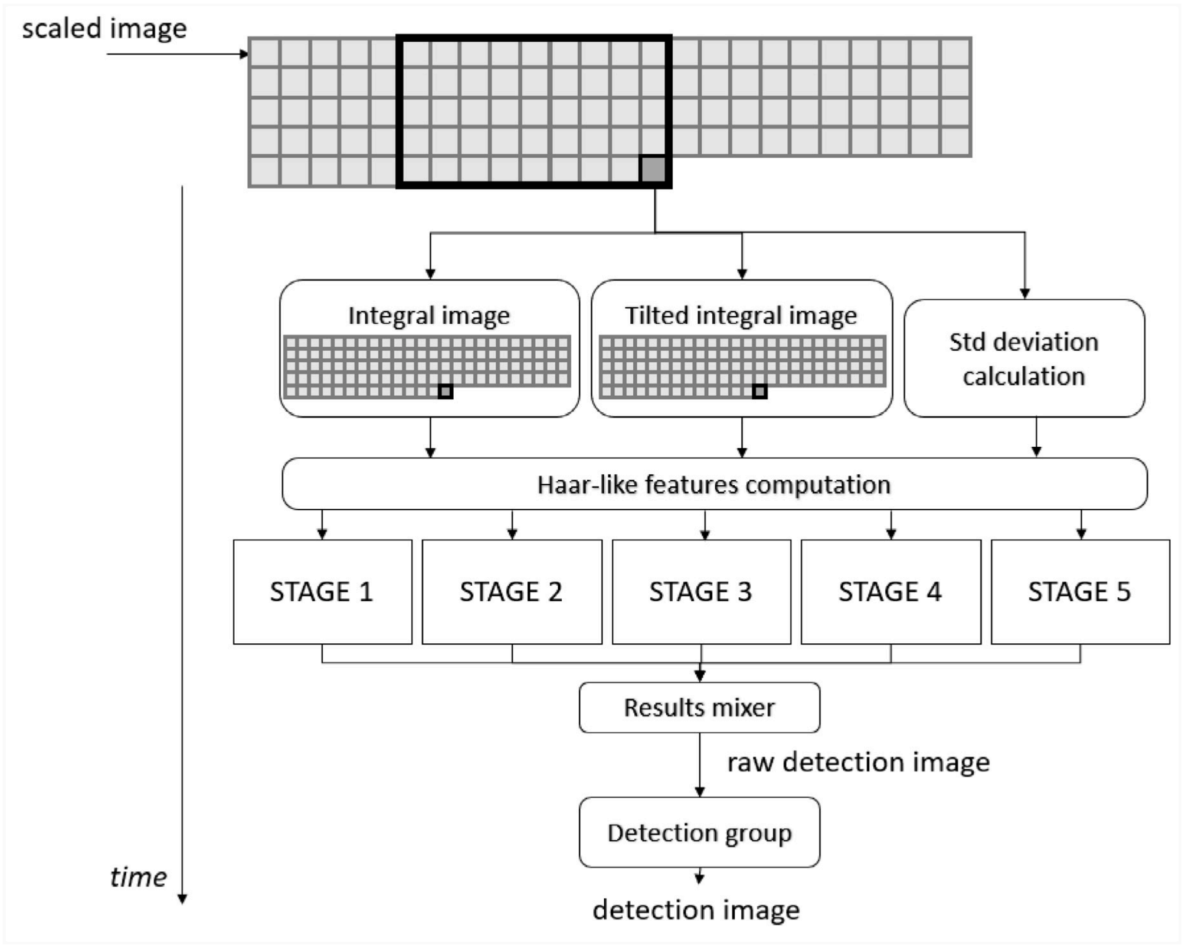

Fig. 3. Proposed classifier structure.

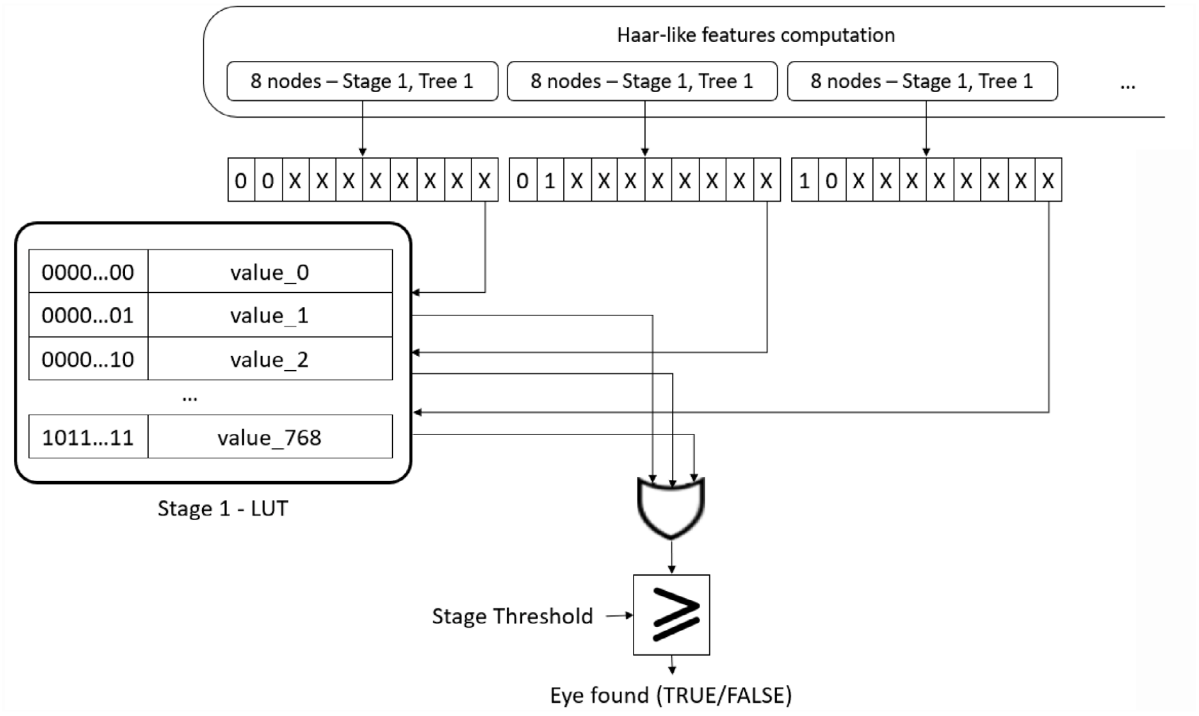

Fig. 4. Design of a complete stage of the classifier.

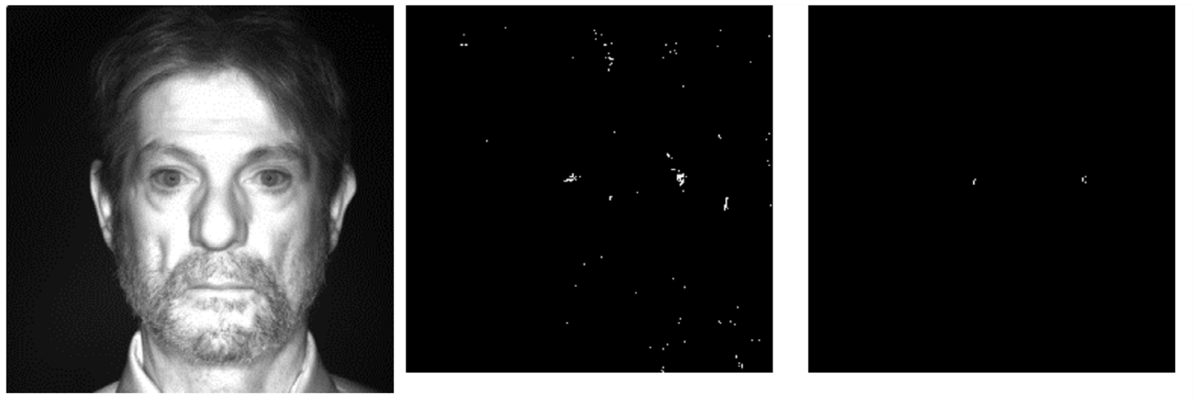

Fig. 5. (Left) Input image; (Middle) Positive detection values; and (Right) Final detection values obtained using the Detection group module. The obtained images are slightly smaller than the input one (see text for details).

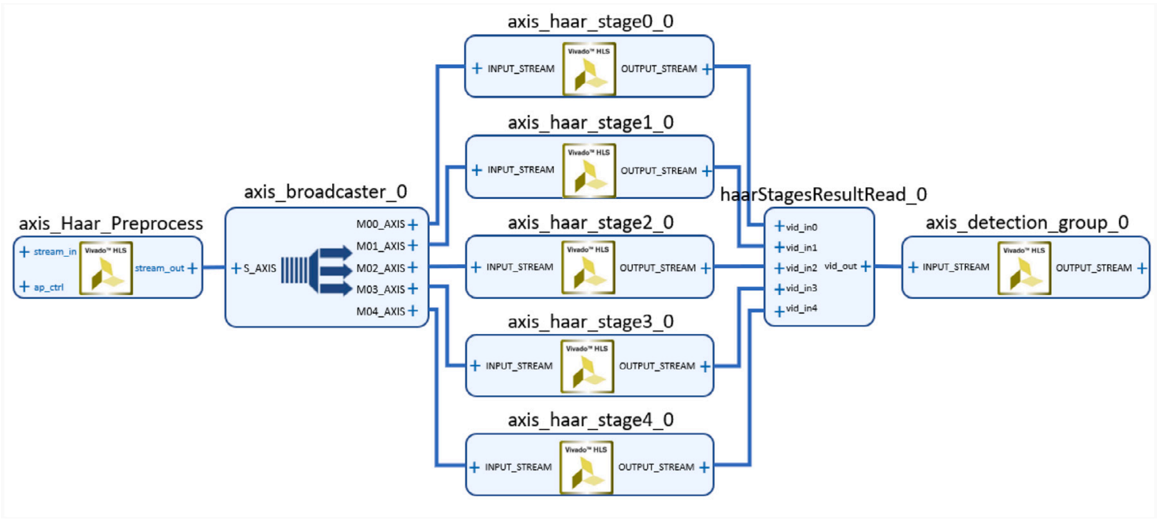

Fig. 6. Block diagram of the first version of the classifier implementation.

system to choose as eyes the most valuated windows when all the frame is processed.

Table 1 illustrates the resource consumption associated to the synthesis and implementation of the proposed classifier. The table shows the values for two implementations of our final proposal. In these implementations, we have tried to either optimize resource consumption or reduce latency. Thus, in the first implementation (Opt1), the latency was 270,085 clock cycles. This provides a speed of 555 frames per second (fps) using an input clock frequency of 150 MHz. The Opt1 proposal uses 67 BRAM. The second implementation (Opt2) uses 81 BRAM. In this second case, the latency is only 74,242 clock cycles, and, with the clock frequency of 150MHz, the proposal should provide 2020 fps. However, the VDMA channels limit this value and the real speed is 752 fps. If we were able to maintain this speed by adding the rest of the cores of the system, we could process each frame in 1.33 ms. As described in Section 5, this has not been possible, and the speed will decrease due to the constraints imposed by the image sensor and the post-processing stage and final sending of eye captures.

5. Experimental evaluation

5.1. Experimental setting

The proposed framework was designed for an iris recognition system where subjects walk through an access control point. They are walking at normal speed (approx. 1–2 m/s) and should be able to wear normal eyeglasses or contact lenses. The camera system is about 2 metres in front of the subject,

Table 1
Total resource usage for the classifier.

Name	BRAM_18K	DSP48E	FF	LUT	URAM
Opt1 - Total	67	579	34780	25714	0
Opt1 - Utilization (%)	26	79	19	29	0
Opt2 - Total	81	582	34645	25740	0
Opt2 - Utilization (%)	31	79	19	29	0
Available	256	728	175680	87840	48

The vision module employs a 16MP EMERALD high-speed image sensor from Teledyne e2v. This sensor can provide up to 47 FPS through 16 LVDS lanes. It is mounted in a specific board (the Sensor board). Our hardware implementation is synthesized for an Ultrascale+ XCZU4EV micromodule from Trenz. The micromodule is mounted in a carrier that exposes the GPIO and that provides Ethernet connectivity, SD card, HDMI, debug interface and FPGA Mezzanine Card (FMC) connector. An adaptation board is needed to adequate the signals between the sensor board and the carrier one. Fig. 7 shows all boards employed in the vision module.

Taking the classifier in Fig. 6 as the core, the whole framework needs to add the deserializer for capturing the input frames from the

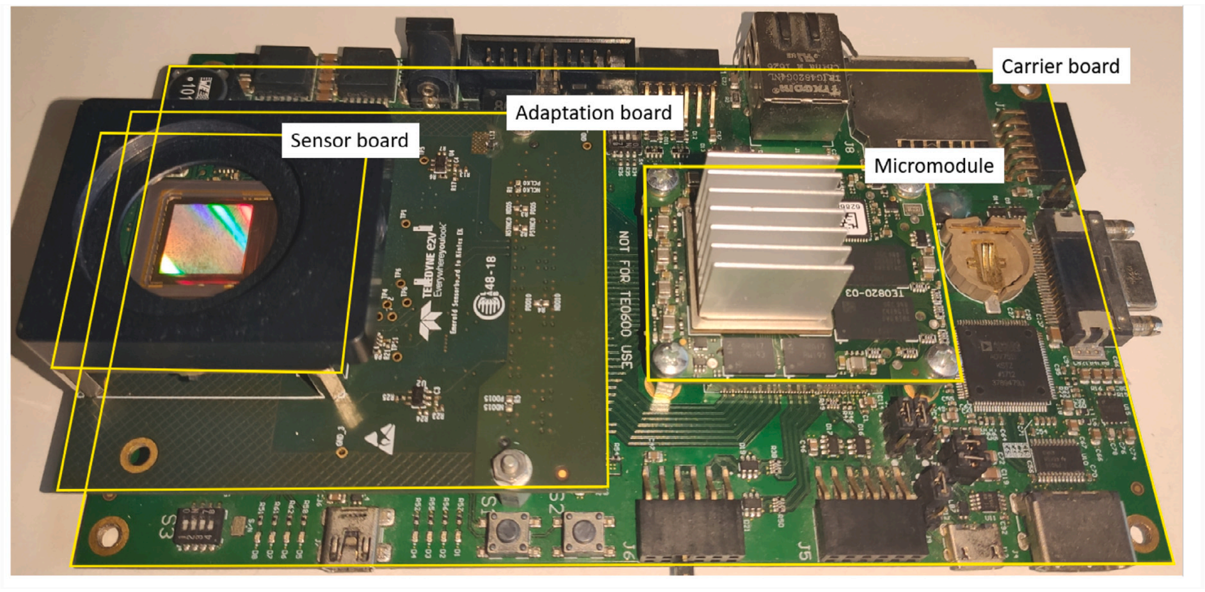

Fig. 7. Boards involved in the vision module.

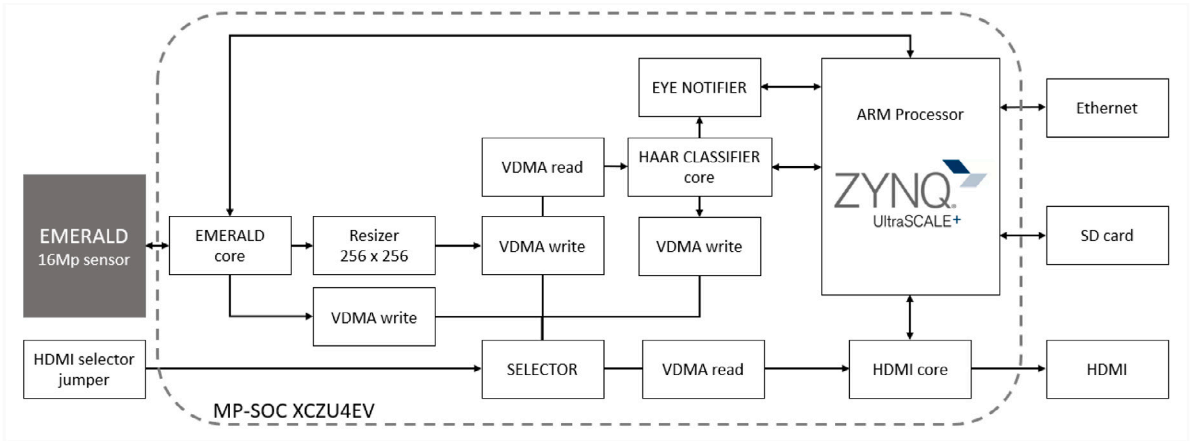

Fig. 8. Overview of the proposed framework.

LVDS channels and the postprocessing module, for quantifying the relevance as an eye of each tentative detection and notifying the eye positions to the software part of the AP SoC. Fig. 8 shows the logical architecture of our evaluation system.

The proposed design was synthesized and implemented using Xilinx Vivado. As aforementioned, our target device was the Ultrascale+ XCZU4EV micromodule from Trenz. Sensor calibration and image capture are implemented on the Programmable Logic (PL) and Processing Subsystem (PS) of the AP SoC. Briefly, the EMERALD deserializer core is the responsible of managing all control signals of the sensor (reset, stop, trigger...). It is also in charge of the correct wake up of the sensor and LVDS calibration. This calibration process is commanded from the ARM core through this module. Due to the high-speed characteristic of the 16 LVDS lanes some delays can occur on PCB traces or in the input buffers of the FPGA, therefore each lane must be calibrated by means of adding or removing delays in order to synchronize the all.

Deserialized signals are processed into a video stream using the EmeraldFrameVideo core on the FPGA. Briefly, it captures the data to generate frames and sends them into a video stream, which is used by a VDMA channel to store up to 8 16MP frames in the onboard DDR3 RAM. Simultaneously, the stream is resized into a 256 × 256 video stream, which is also stored on RAM through another VDMA.

The 256 × 256 video stream is used by the HAAR CLASSIFIER core, and it is processed into a video stream on which white dots are positive detections (Fig. 5). This stream is then processed by the EYE NOTIFIER core to provide the frame number and detected coordinates to the ARM processor.

The HDMI interface allows viewing a video stream in real time. Since there are various video streams within the framework, a SELECTOR core is included to select which one to display. This selection is made using jumpers. We can choose to display the full captured frame, the resized frame, or the detections provided by the HAAR CLASSIFIER CORE.

On the PS part, the ARM is working in AMP (Asymmetric Multiprocessing) mode, so that one core is for capturing eye windows and the other core is for collecting these windows (640 × 480 pixels) and sending them via Ethernet for recognition. The core 1 oversees configuring all the peripherals, among these, the direct accesses to memory of the sensor frames, the resized images, and the processed images of the eye detection. Finally, it configures the mechanism for being informed of a window detection (interrupt controller) and waits for a valid image to be detected. When a valid image is detected, the processor stores the resized version of the input frame and the detection window/s in a shared memory with the second core.

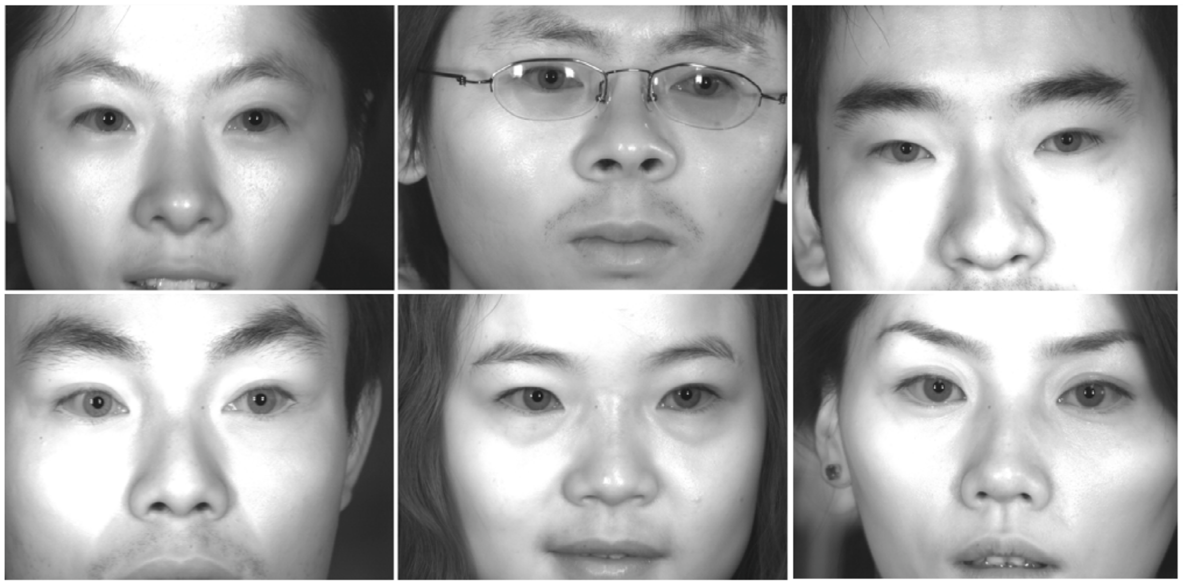

Fig. 9. Example images from CASIA-Iris-Distance v4.0 database.

Core 2 takes care of communication, configures the entire Ethernet stack using the LWIP library, and configures a UDP server to which an external computer can connect. Finally, it keeps checking the shared memory and sends the images found in the shared memory via UDP.

5.2. Obtained results

The whole framework has been successfully integrated in the iris recognition system. The detection system sends a stream of eye images of 640 × 480 pixels to a host computer managing the iris pattern extraction and identification. However, for providing qualitatively comparable results, we evaluated our method on the CASIA-Iris-Distance, version 4.0 database.[1] This database contains 2567 images of 142 people, most of them graduate students of the Chinese Academy of Sciences' Institute of Automation (CASIA). The age of the people spans from 19 to 61. The database was captured indoor, with a distance of more than 2 m, and using a self-developed long-range multi-modal biometric image acquisition and recognition system (LMBS). Detailed specifications of the physical system, magnification factor and focal length of the camera lens are not unveiled. Fig. 9 shows some examples of images. The size of the images is 2352 × 1728, and they include both eyes in a facial image. The pixel diameter of the iris area is less than about 170 pixels. In addition, in order to consider the various capturing environments along the long Z-distance, various noise factors such as severe off-angle, specular reflection on glasses, low illumination and hair occlusion were considered. Using this database, this section presents experimental results for the precision and efficiency of the proposed approach. The classifier was calibrated (Haar features and threshold values) using images out of this database. Thus, all images were used as the test data set.

Table 2 shows the results obtained on this database by different methods. These methods used a 3.6 GHz Intel i7-7700 CPU, NVIDIA GeForce GTX 1070 (1920 CUDA cores and 8 GB of memory) and 16 GB of memory (Nsaif et al., 2021). In the table, we can find the proposal by Kawaguchi and Rizon (2003) that uses the brightness of the image to detect the face and, in this, uses pattern matching and the geometry associated with the two eyes to detect them. Following a similar scheme (considering face and eyes), the proposal of Uhl and Wild (2012) uses completely different features and classifier, based on the proposal of Viola–Jones and Lienhart and Maydt (Lienhart & Maydt, 2002). The algorithm of Chai et al. (2019) starts with a Gaussian filtering of the image, which reduces the effect of possibly different lighting conditions. Next, iris candidates associated with the detection of iris reflections are generated. A cost is then calculated for each iris candidate using how the area is matched to a generic eye template, the intensity variation factor, the circularity factor and the reflection factor. Finally, a matching process is used to determine the actual iris pair in order to locate both irises. The other three methods with which this proposal is compared in Table 2 are Deep Learning methods using CNNs. In general, all three methods employ algorithms that propose regions where eyes can be located. These detection networks are slow but proposals such as SPPnet and Fast R-CNN have reduced the execution time. We have described the Faster R-CNN (Ren et al., 2017) in Section 2. In this proposal, a Region Proposal Network (RPN) is introduced that shares the convolutional characteristics of the full image with the detection network, allowing them to perform region proposals at almost no cost. The RPN is a fully convolutional network that simultaneously predicts object boundaries and "objectivity" scores at each position. The regions proposed by the RPN are used by a Fast R-CNN for detection. Significantly, the RPN and Fast R-CNN are merged into a single network, sharing their convolutional characteristics. Using the recently popular fusion of neural networks with attention mechanisms, the RPN component will inform the unified network where to look. This same scheme is found in the FR-CNN-NB and FR-CNN-GNB (Nsaif et al., 2021). Basically, these methods use Faster R-CNN to determine the features and provide a first estimation of the position of the eyes, which are then improved using a Bayesian model (FR-CNNN-NB) that can be complemented with the use of Gaussian Filters (FR-CNNN-GNB).

For conducting these tests, we must slightly change our design. Thus, the image capture from sensor was disabled and a single image loaded from the SD card was used. The load into the frame buffer of the sensor VDMA is addressed by the ARM (PS part), which is also the responsible of launching the trial. Several video analyser cores were included on the video stream from which information about video resolution and rate (fps) were extracted. During the test it is possible to see the resulting image through HDMI and receive the eye crops through Ethernet. Therefore, it is possible to evaluate the resulting image and analyses if it matches the expected result. The system runs

[1] http://biometrics.idealtest.org/

Table 2
Comparison of the accuracy and time of eye detection of the proposed approach with state-of-art algorithms, on the CASIA v4 Distance database.

Approach	Success rate	Time (s)
Kawaguchi and Rizon (2003)	92.91%	–
Uhl and Wild (2012)	96.4%	1.28
Chai et al. (2019)	95.61%	–
Faster R-CNN (Ren et al., 2017)	98.21%	0.59
FR-CNN-NB (Nsaif et al., 2021)	99.1%	0.59
FR-CNN-GNB (Nsaif et al., 2021)	100%	0.59
Proposed	100%	0.011

Table 3
Total resource usage.

Resource	Utilization	Available	%
LUT	64190	87840	73.1
LUTRAM	3437	57600	6
FlipFlop	59678	175680	34
BRAM	72	128	56.2
URAM	1	48	2.1

at 88 fps when the eye notifier core is reporting the coordinates to the processor and reached the aforementioned 752 fps when the eye notifier core is disabled.

Table 3 illustrates the implementation results and resource utilization of our proposal based on the logic elements of the target device. Scale images and detection window have a size of 256 × 256 and 25 × 20 pixels, respectively. The classifier implements the Opt1 version (see Section 4). Obtained detection images are provided at 640 × 480 pixels.

6. Conclusions and future work

This work proposed an effective redesign for embedding the Viola–Jones approach in an AP SoC to detect the eye location in a video sequence. The proposed approach estimates the eye region directly, without requiring localization of the human face. It is also resistant to reflection by glasses or occlusion, as the evaluation with the CASIA v4 Distance database has shown. Although our approach makes use of machine learning techniques to set up the internal parameters of our classifier, we do not rely on CNNs for feature extraction. As we stated in Section 1, the appearance and geometry of the eye is distinguishable and feature extractors, such as the one based on Haar-like features, can successfully solve the problem (Raja et al., 2019). Our effort has been to design a highly parallel, single-pass image-processing approach that includes the characterization and classification processes. The classification step in the Viola–Jones approach was originally designed to be sequentially executed, prioritizing that a negative response of one weak classifier allows to abort the whole identification process. This idea of reducing the number of evaluated weak classifiers is also inherent to the tree implementation. In our design, all weak classifiers are always evaluated, but this is not the problem for achieving a high efficiency value as these evaluations are all of them addressed in parallel. It is important to note that our classifier can process up to 752 fps, without reducing the success ratio of the Viola–Jones approach for eye detection.

Future work focuses on improving the final post-processing step on our detection system. It reviews all the detection image for marking the most relevant eye locations and then it sends the notifications to the PS part in the AP SoC with the coordinates of the eye images. When this final stage is added, the result is reduced to 88 fps. Another line of research is to use the free resources in the AP SoC. We have designed an approach for evaluating the contrast index of the eye images before sending to the host computer for iris identification. Poorly focused images can then be discarded. We are now evaluating to integrate a presentation attack detection (PAD) module to distinguish between authentic iris images (perhaps wearing clear contact lenses) and irises with textured contact lenses. Binary Statistical Image Features (BSIF) for extracting PAD-related features and Support Vector Machine (SVM) classifiers can be implemented in the AP SoC.

CRediT authorship contribution statement

Camilo A. Ruiz-Beltrán: Formal analysis, Software, Validation. **Adrián Romero-Garcés:** Software. **Martín González:** Formal analysis, Software, Supervision. **Antonio Sánchez Pedraza:** Formal analysis, Software, Validation. **Juan A. Rodríguez-Fernández:** Formal analysis. **Antonio Bandera:** Software, Supervision, Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been partially developed within the project RTI2018-099522-B-C4X, funded by the Gobierno de España and FEDER funds, Spain, and the ARMORI project (CEIATECH-10) funded by the University of Málaga, Spain. Portions of the research in this paper use the CASIA-Iris V4 collected by the Chinese Academy of Sciences - Institute of Automation (CASIA). Funding for open access charge: Universidad de Málaga/CBUA.

References

Benrachou, D. E., dos Santos, F. N., Boulebtateche, B., & Bensaoula, S. (2015). Online vision-based eye detection: LBP/SVM vs LBP/LSTM-RNN. In A. P. Moreira, A. Matos, & G. Veiga (Eds.), *CONTROLO'2014 – proceedings of the 11th Portuguese conference on automatic control* (pp. 659–668). Cham: Springer International Publishing.

Chai, T.-Y., Goi, B.-M., Tay, Y.-H., & Khoo, Y.-H. (2019). Vote-based iris detection system. In *ICDSP 2019*, *Proceedings of the 2019 3rd international conference on digital signal processing* (pp. 114–118). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3316551.3316558.

Daugman, J. (2004). How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, *11*(3), 21–30.

D'Orazio, T., Leo, M., & Distante, A. (2004). Eye detection in face images for a driver vigilance system. In *IEEE intelligent vehicles symposium, 2004* (pp. 95–98). http://dx.doi.org/10.1109/IVS.2004.1336362.

Fredj, H. b., Sghair, S., & Souani, C. (2020). An efficient parallel implementation of face detection system using CUDA. In *2020 5th international conference on advanced technologies for signal and image processing* (pp. 1–6). http://dx.doi.org/10.1109/ATSIP49331.2020.9231723.

Gao, C., & Lu, S.-L. (2008). Novel FPGA based haar classifier face detection algorithm acceleration. In *2008 International conference on field programmable logic and applications* (pp. 373–378). http://dx.doi.org/10.1109/FPL.2008.4629966.

Hiromoto, M., Nakahara, K., Sugano, H., Nakamura, Y., & Miyamoto, R. (2007). A specialized processor suitable for AdaBoost-based detection with haar-like features. In *2007 IEEE conference on computer vision and pattern recognition* (pp. 1–8). http://dx.doi.org/10.1109/CVPR.2007.383415.

Jain, V., & Patel, D. (2016). A GPU based implementation of robust face detection system. *Procedia Computer Science*, *87*, 156–163. http://dx.doi.org/10.1016/j.procs.2016.05.142, URL: https://www.sciencedirect.com/science/article/pii/S1877050916304811, Fourth International Conference on Recent Trends in Computer Science & Engineering (ICRTCSE 2016).

Ji, Q., Wechsler, H., Duchowski, A., & Flickner, M. (2005). Special issue: eye detection and tracking. *Computer Vision and Image Understanding*, *98*(1), 1–3. http://dx.doi.org/10.1016/j.cviu.2004.07.006.

Kawaguchi, T., & Rizon, M. (2003). Iris detection using intensity and edge information. *Pattern Recognition*, *36*(2), 549–562. http://dx.doi.org/10.1016/S0031-3203(02)00066-3, URL: https://www.sciencedirect.com/science/article/pii/S0031320302000663, Biometrics.

Kim, M., Lee, D., & Kim, K.-Y. (2015). System architecture for real-time face detection on analog video camera. *International Journal of Distributed Sensor Networks*, *11*(5), 251–386. http://dx.doi.org/10.1155/2015/251386.

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. In *2016 IEEE conference on computer vision and pattern recognitio* (pp. 2176–2184).

Kumar, V., Asati, A., & Gupta, A. (2018). Hardware accelerators for iris localization. *Journal of Signal Processing Systems*, *90*, 655–671.

Lai, H.-C., Savvides, M., & Chen, T. (2007). Proposed FPGA hardware architecture for high frame rate (100 fps) face detection using feature cascade classifiers. In *2007 First IEEE international conference on biometrics: theory, applications, and systems* (pp. 1–6). http://dx.doi.org/10.1109/BTAS.2007.4401930.

Li, B., & Fu, H. (2018). Real time eye detector with cascaded convolutional neural networks. *Applied Computational Intelligence and Soft Computing, 1439312.*

Lienhart, R., Liang, L., & Kuranov, A. (2003). A detector tree of boosted classifiers for real-time object detection and tracking. In *2003 International conference on multimedia and expo. ICME '03. proceedings (Cat. No.03TH8698), vol. 2* (pp. II–277). http://dx.doi.org/10.1109/ICME.2003.1221607.

Lienhart, R., & Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing, vol. 1* (p. I). http://dx.doi.org/10.1109/ICIP.2002.1038171.

Nair, V., Laprise, P.-O., & Clark, J. J. (2005). An FPGA-based people detection system. *EURASIP Journal on Advances in Signal Processing*, (7), 1047–1061. http://dx.doi.org/10.1155/ASP.2005.1047.

Nsaif, A. K., Ali, S. H. M., Jassim, K. N., Nseaf, A. K., Sulaiman, R., Al-Qaraghuli, A., Wahdan, O., & Nayan, N. A. (2021). FRCNN-GNB: Cascade faster R-CNN with gabor filters and naïve Bayes for enhanced eye detection. *IEEE Access*, *9*, 15708–15719. http://dx.doi.org/10.1109/ACCESS.2021.3052851.

Quan, D., Zhao, Y., Li, J., & Zheng, H. H. (2013). An effective method and device for eye detection with physical properties. In *Proceedings of the 2nd international conference on computer science and electronics engineering.*

Raja, K. B., Raghavendra, R., & Busch, C. (2019). Morton filters for iris template protection - an incremental and superior approach over bloom filters. In *2019 IEEE 10th international conference on biometrics theory, applications and systems* (pp. 1–8). http://dx.doi.org/10.1109/BTAS46853.2019.9185986.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. http://dx.doi.org/10.1109/TPAMI.2016.2577031.

Saif, A. F. M. S., Garba, A. G., Awwalu, J., Arshad, H., & Zakaria, L. Q. (2017). Performance comparison of min-max normalisation on frontal face detection using haar classifiers. *Pertanika Journal of Science and Technology*, *25*, Article 163171.

Theocharides, T., Vijaykrishnan, N., & Irwin, M. (2006). A parallel architecture for hardware face detection. In *IEEE computer society annual symposium on emerging VLSI technologies and architectures* (p. 2). http://dx.doi.org/10.1109/ISVLSI.2006.10.

Thongleng, C., & Kaewapichai, W. (2018). Case studies to improve viola-jones for eye detection. In *ICIGP 2018, Proceedings of the 2018 international conference on image and graphics processing* (pp. 48–52). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3191442.3191456.

Uhl, A., & Wild, P. (2012). Combining face with face-part detectors under Gaussian assumption. In A. Campilho, & M. Kamel (Eds.), *Image analysis and recognition* (pp. 80–89). Berlin, Heidelberg: Springer Berlin Heidelberg.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, vol. 1* (p. I). http://dx.doi.org/10.1109/CVPR.2001.990517.

Wei, Y., Bing, X., & Chareonsak, C. (2004). Fpga implementation of AdaBoost algorithm for detection of face biometrics. In *IEEE International workshop on biomedical circuits and systems, 2004* (pp. S1/6–17). http://dx.doi.org/10.1109/BIOCAS.2004.1454161.

Xie, D., Zhang, L., & Bai, L. (2017). Deep learning in visual computing and signal processing. *Applied Computational Intelligence and Soft Computing*, 1–13.

Yang, M., Wu, Y., Crenshaw, J., Augustine, B., & Mareachen, R. (2006). Face detection for automatic exposure control in handheld camera. In *Fourth IEEE international conference on computer vision systems* (p. 17). http://dx.doi.org/10.1109/ICVS.2006.26.