



Automated in-camera detection of flasheye defects

Title	Automated in-camera detection of flasheye defects
Author(s)	Corcoran, Peter;Bigioi, Petronel
Publication Date	2005
Publisher	IEEE

Automated In-Camera Detection of Flash-Eye Defects

Peter Corcoran, Petronel Bigioi, Eran Steinberg and Alexei Pososin.

Abstract — *This paper examines the problem of performing automated real-time detection of flash eye defects (redeye) in the firmware of a digital camera. Several different algorithms are compared, timing and memory requirements on several embedded architectures are presented. A discussion on advanced in-camera techniques to improve on standard algorithms is also presented¹.*

Index Terms — **digital camera, image processing, redeye, image defect detection.**

I. INTRODUCTION

Flash eye-defects continue to be the most frequent problem encountered by users of digital cameras and related imaging appliances. They occur due to reflections within a subject's eye and typically manifest themselves in a digital image as a reddish region where the black pupil of the subject's eye would normally appear. The unnatural glowing red of an eye is due to internal reflections from the vascular membrane behind the retina, which is rich in blood vessels.

This phenomenon is well understood to be caused in part by a small angle between the flash of the camera and the lens of the camera. This angle has decreased with the miniaturization of cameras with integral flash capabilities. This trend will continue with the continued miniaturization of cameras and the incorporation of flash units in next-generation camera-phones.

Because images can now be captured as or converted into digital images within an image acquisition device such as a digital camera, it is thus possible to correct the red-eye problem in an image digitally. Some prior art schemes have been proposed to correct the red-eye problem digitally as will be described in the next section.

II. AUTOMATED DETECTION OF EYE DEFECTS – LITERATURE REVIEW

Currently, although desktop image processing software programs incorporate functions to allow the correction of eye-defects these programs require significant user-input both to identify the eye-defects and to manage the subsequent correction of these.

The authors of [1, 2] have sought to improve on the manual correction of eye defects by adding semi-automated mechanisms to analyze the areas around an eye and simulate the colors, regional transitions and “glint” of the eye. However these techniques are dependent on the eye region being large

enough to partition into a number of distinct regions and will only work in a subset of cases. Further, they require the user to manually select a region around the eye prior to the automated analysis of the eye region.

Some improved automated techniques have also been presented in the literature and in the patent prior art. These will now be reviewed, in particular with a view to evaluating the algorithmic complexity, speed, memory requirements and overall performance:

A. Face Detection Algorithms

Algorithms which employ face detection and/or eye detection techniques prior to testing for an eye-defect. Unfortunately such algorithms are computationally quite intensive and even state-of art face detectors are hard-pressed to manage *15 fps* on high-end desktop computers.

The authors of [3, 4] describe several approaches which eliminate user intervention and detect red-eye defects in a completely automatic manner. These techniques share a common approach of firstly detecting the face regions of persons in a digital image, secondly detecting the eye region in each face and finally determining if red-eye defects exist in the subject's eyes. Both patents adopt quite complex, and thus resource intensive, image processing techniques to detect face and eye regions in an image and subsequently verify the presence of red-eye defects. Further, there are some disadvantages to this approach when two, or more, facial regions overlap or are in close proximity to one another, particularly when a technique is weighted heavily to detect balanced eye-pairs.

In addition to the aforementioned disadvantages, the authors of [3] employ a complex procedure to detect faces and balanced eye-pairs from a skin-map of the image. This task requires several partitioning and re-scaling operations. Significant additional processing of a potential face region of the image then follows in order to determine if a matching pair of eyes is present. Finally the image pixels in the detected eye regions go through a complex scoring process to determine if a red-eye defect is present. Further, this technique does not propose a means for correcting a red-eye defect after it is detected.

A more comprehensive algorithm which does include an automated means of defect correction is described in [5]. However this technique has problems with outliers and smaller eye regions. It also employs a face detection algorithms which, although relatively fast and accurate requires substantial amounts of computing power. Further, if we consider such algorithms in terms of fundamental image processing then an algorithm based on face detection requires a three-step segmentation process: firstly the face regions within an image must be detected, segmented and confirmed as face regions; secondly the eye regions within each face region must be

¹Peter Corcoran is the Director of IP at FotoNation Ireland Ltd, Galway (e-mail: pcor@fotonation.com).

Petronel Bigioi is VP of Engineering at FotoNation Ireland Ltd, Galway (e-mail: petronel@fotonation.com).

Eran Steinberg is CEO of FotoNation Inc, San Francisco, (e-mail: eran@fotonation.com).

Alexei Pososin is a Senior Research Engineer at FotoNation Ireland Ltd.

further detected, segmented and confirmed; finally the redness of these eye regions must be analyzed prior to applying a correction algorithm.

For in-camera applications all of these approaches described in this section are currently impractical in terms of computational requirements.

B. Neural Network and other “Trained” Algorithms

Training algorithms which employ techniques such as neural networks, and require substantial training data sets. Algorithms of this type have been described in the prior art and they are reasonably effective for general purpose detection of flash-defects.

However most such techniques are typically not easily separable into modular subprocesses due to the non-linearity of the training process. In other words, if we wish to train for a specific form of flash-defect the entire algorithm must be retrained. This presents significant disadvantages as the training must be undertaken externally to the camera.

The authors of [4] make extensive use of neural network technology in the detection algorithms. Such techniques are well known and are optimized for implementation as a dedicated hardware solution, e.g. as a dedicated ASIC (application specific integrated circuit). Typically, the implementation of a neural network on a conventional computer will require significant memory and computing power. In addition this patent requires multiple re-scaling and rotations of the original digital images, both tasks increasing significantly the requirements for both computing power and memory storage. Further, the authors of [4] do not propose a detailed technique for correcting any red-eye defects but only suggests replacing the color of an identified defect pixel with a predetermined color. This step, in turn, suggests that user intervention will be required in any practical embodiment of the invention described in this patent.

C. Feature Extraction and Templates

Feature extraction approaches in which a detailed set of regional characteristics of an eye-defect region are matched to a template based regional scan of an image are also popular in the prior art. Although such feature & template based techniques are relative fast they do not scale well and are not very good at detecting outliers. Adding additional templates to compensate for these deficiencies is not practical given the resource restrictions within a digital camera.

D. Novel Image Processing Techniques

Other novel image processing techniques, include the use of specular reflections or the hough transform to assist in locating the potential eye-defect regions [6, 7 & 8]. Now although such techniques can locate a significant percentage of flash-defects very quickly, there are still many outliers and cases where the specular reflection does not occur within an eye region and it is time-consuming to search a second time for these missed regions.

Each of these approaches has benefits and drawbacks. On a desktop computer, with relatively “unlimited” resources it makes sense to use the most sophisticated image-processing

techniques available. However the requirements for incorporation of an algorithm into a camera and to support real-time detection functionality on an embedded platform with more limited resources and processing capabilities are significantly different.

III. A PRACTICAL IN-CAMERA ALGORITHM

In order to identify red eye areas in the image, specific characteristics must be determined. Red eyes are image areas characterized by:

- *a reddish tone;*
- *a round shape;*
- *an area that is relatively small with respect to the whole image area;*
- *compactness, i.e., they do not contain large holes;*
- *they are located in the neighborhood of skin patches;*

Red-eye segments may additionally be characterized by:

- *the presence of whitish pixels, associated with the sclera of the eye around the cornea, in close proximity to the segment*
- *the existence of a relatively high local variation in contrast across the segment*
- *the existence of a distinctive step increase in the local histogram of the red-color component of the image*

The detection algorithm comprises the following main sub-steps: (a) the image is converted into Lab color space, a well defined close to uniform Luminance chrominance color space, where the “L” axis denotes luminance, the “a” axis denotes the red-green component of color chrominance and the “b” axis denotes the blue-yellow color component; (b) the image is segmented and red pixels with the correct luminosity and chrominance to match red-eye defects are labeled; (c) connected segments are labeled in a single raster-scan of the image; (d) segments with inappropriate areas (too large or too small) are rejected; (e) elongated segments are rejected; (f) non-compact segments are eliminated; (g) segments which are not located in the vicinity of skin patches are eliminated; (h) segments with low contrast are eliminated. At the end of this elimination process any remaining red segments in the image have a very high probability of being red-eye defects.

We next consider each of the algorithm substeps in further detail:

A. Color Space Considerations

Our initial work was entirely in Lab space as this offers an representation of color space which is optimized to human perception. While there are advantages to working in Lab space it is often not practical for embedded systems therefore we have developed a number of optimized mapping techniques between this color space and the more common HUV, YCC and RGB color spaces. When such mappings are carefully chosen they can retain the benefits of using filter thresholds developed in the perceptual color space while allowing fast computations within the internal embedded systems of a digital camera.

B. Initial Determination of “Red” Pixels

The next step is to categorize pixels as potential members of a red-eye region. An initial estimate of the suitability of individual pixels is best achieved by setting a conservative threshold on the “a” and “L” components of a pixel. Thus if the conditions $a > a_{RED}$ and $L > L_{RED}$ are met, then a pixel is considered sufficiently red to be a potential member of a red-eye region.

We have found useful values of a_{RED} and L_{RED} to be +17 and +40 respectively. Note that the a component covers values from -500 up to +500. Negative values indicate green tones while positive values indicate red tones. The additional luminosity constraint used in the decision step is useful in order to avoid dark pixels, the color of which is barely perceivable by the human eye, to be labeled as red.

C. Image Segmentation and Labeling

Next it is necessary to group these “red” pixels into connected segments 103. Identification of connected segments in the binary image is achieved by applying a standard labeling algorithm that works in a single raster-scan of the image using a label correspondence look-up table.

Labeling is a well-known technique in image processing and detailed descriptions of suitable algorithms can be found in [1,2]. Note that the processing time requested by a labeling algorithm actually depends upon the number of distinct segments present in the binary image being labeled. The algorithm described in this paper is the fastest on average. It completes the labeling of all segments within a single raster scan of the image, by using a label correspondence LUT.

Fig 1(a) shows a diagrammatic representation of a 4-pixel neighborhood, shaded light grey in the figure and containing the three upper pixels and the pixel to the left of the current pixel, shaded dark grey in the figure. This 4-pixel neighborhood is used in our preferred labeling algorithm. A look-up table, LUT, is defined to hold correspondence labels. The labeling algorithm as described in Fig 1(b) begins by loading the image, initializing the LUT and the current pixel pointer. It then begins recursive iteration through all the pixels of an image.

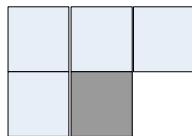


Fig 1(a): The current pixel (dark gray) and current pixel neighborhood (light gray) for our segmentation algorithm.

The image is raster-scanned from top-left to bottom-right. If a pixel satisfies membership criteria for a segment, i.e., if it is sufficiently “red”, then the algorithm checks for other “red” pixels in the current 4-pixel neighborhood: (i) if there are no other “red” pixels, then the current pixel is assigned membership of the current label; the LUT is then updated and the current label value is incremented; alternatively, (ii) if there are other “red” pixels in the 4-pixel neighborhood then the current pixel is given membership in the segment with the lowest label value and the LUT is updated accordingly.

After the current pixel has been labeled as part of a “red” segment, or has been categorized as “non-red”, a test is then performed to determine if it is the last pixel in the image. If the current pixel is the last pixel in the image then a final update of the LUT is performed. Otherwise the next image pixel is obtained by incrementing the current pixel pointer and it is then processed in the same manner.

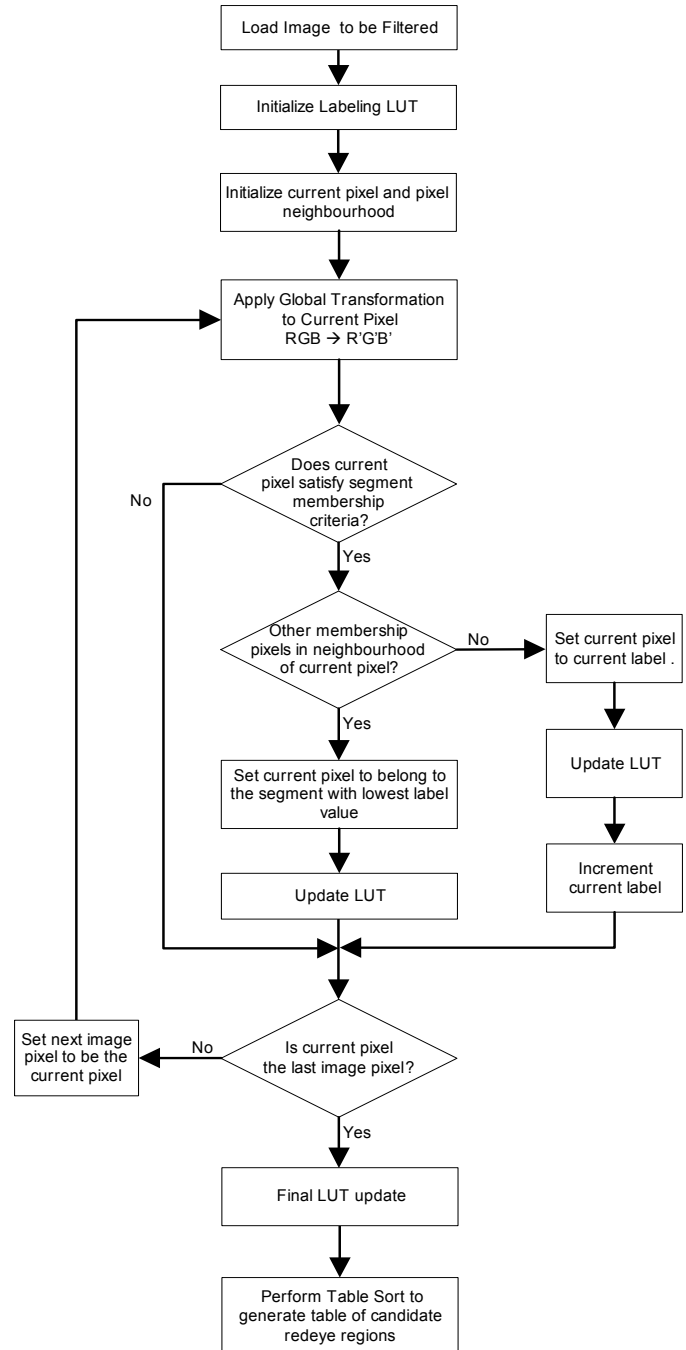


Fig 1(b): Pixel locating, segmenting and labeling algorithm

Once the final image pixel is processed and the final structure of the LUT computed, all of the pixels with segment

membership are sorted into a labeled-segment table of potential red-eye segments.

After this initial segmentation process the detected regions are subjected to a range of additional image processing filters to reduce false positives and determine additional data relating to the surviving regions. Thus we next proceed to begin eliminating members of the initial set of potential red-eye segments based on a range of additional segment attributes.

D. Geometric Analysis

A number of geometric and shape related attributes of the initial set of potential defect segments can be readily computed from the LUT data. As these filters mostly involve binary decisions they are computationally efficient.

1) Calculation of Segment Area

The first criteria we apply is to eliminate segments of an inappropriate size for a red-eye segment, i.e., segments that are too small, or too large. This elimination step begins by loading a list of pixels, described by their i, j co-ordinates, that are members of a particular segment. The segment area is determined by obtaining the maximum and minimum i and j co-ordinates of its member pixels. The area of the segment is then calculated as $A_K = \Sigma P_{RED}$, or in words the effective area of the potential red-eye segment is the sum of the confirmed "red" pixels (i.e. P_{RED}) in that segment.

Only segments S_K with the area A_K between given limits of $A_{MAX} \geq A_k \geq A_{MIN}$ are further inspected. Now red-eye defect segments can be quite small, but it is generally unproductive to process segments below a certain minimum number of "red" pixels as such defects are not particularly noticeable and do not have sufficient detail to benefit from additional processing. For a standard digital image of 1-2 megapixels in size a suitable lower threshold for the area, A_{MIN} , is of the order of 20-50 pixels. Thus any segments of a size less than A_{MIN} are immediately eliminated from the LUT.

Experiments have led to the conclusion that the area occupied by the red pupil of an eye is normally less than 0.1% of the total area of a standard photograph. This sets the threshold for the first upper limit on segment area, A_{MAX_1} . Any segments that survive these initial elimination steps are next marked as valid red-eye segments on the basis of segment size. However, in certain instances, for example close-up portraits, the upper size limit may be as large as 0.5% of the total area, but the red pupil should now have a distinct surrounding region of whitish color, being the cornea of the eye. Thus any segments that failed the size test are given a second size test and if $A_K = A_{MAX_2}$. Note also that a portrait mode picture can offer be determined from autofocus or scene selection metadata which is typically available to an in-camera application.

2) Elongated Segments

Elimination of elongated segments is described in detail by the flowchart in Fig. 5. For each segment S_K , the variances λ_{kMIN} and λ_{kMAX} along the principal axes are computed. We note that the variance λ of a set of 1-D samples x_i with $i \in [1...N]$ is computed according to the formula:

$$\lambda = \frac{1}{N} \sqrt{\sum_{i=1}^N (x_i - \bar{x})^2}$$

where

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

is the mean. These variances, λ_{kMIN} and λ_{kMAX} are obtained from the eigenvalues of the covariance matrix C_K given by the matrix formula (i, j are the horizontal and vertical co-ordinates of a pixel in the image, said pixel being a member of segment S_K , the K th segment remaining in the LUT of potential red-eye segments):

$$C_k = \frac{1}{A_k} \sum_{i,j \in S_k} [ij]^T \cdot [ij] - [\bar{i}\bar{j}]^T \cdot [\bar{i}\bar{j}]$$

where T denotes the matrix transpose operator, and:

$$\bar{i} = \frac{1}{A_k} \sum_{(i,j) \in S_k} i \quad \bar{j} = \frac{1}{A_k} \sum_{(i,j) \in S_k} j$$

The test process is as follows: the next surviving potential red-eye segment is loaded and the covariance matrix, C_K , for this segment is calculated. The maximum and minimum variances along the principle axes of this segment are next calculated from the eigenvalues of C_K . The aspect ratio R_K of segment S_K is then computed as:

$$R_k = \frac{\lambda_{kMIN}}{\lambda_{kMAX}}$$

Segments that are too elongated 505 to be red-eye defects, i.e., segments with

$$R_k \leq R_{MAX}$$

with R_{MAX} being a pre-defined threshold, are exempted from further processing 506, as they are unlikely to represent red eyes. Non-elongated shapes have the ratio R_k close to unity. A useful criteria for segments with $A_K = A_{KMAX1}$ is to eliminate all segments having $R_K < 0.2$. For larger segments, those having area $A_{MAX_1} = A_K = A_{MAX_2}$, the criteria $R_K < 0.5$ is used to eliminate potential red-eye candidates.

3) Determination of non-Compact Segments

The elimination process for non-compact segments involves loading a segment into memory from the LUT. The segment fill factor F_K is next computed as:

$$F_k = \frac{A_k}{(i_{kMAX} - i_{kMIN})(j_{kMAX} - j_{kMIN})}$$

where i_{kMIN} , i_{kMAX} , j_{kMIN} , and j_{kMAX} represent the minimum and maximum coordinates of pixels within the segment S_K . All segments with $F_K \leq F_{MIN}$ are eliminated from the LUT.

In practice we have found that ratio of the segment area to minimal bounding box area, must be higher than 0.5 for a segment to be sufficiently compact to be a potential red-eye defect. Segments that pass this test for compactness are marked as such and retained in the LUT.

E. Supplemental Image Analysis

We now describe a number of additional filter techniques as examples of methods to identify and eliminate false positive regions or to further refine the initial analysis of the determined segments.

1) Skin Characteristics of Bounding Region

The procedure to eliminate segments that are not located in the neighborhood of skin patches is now described in more detail. Firstly, a surviving red-eye segment is loaded into memory from the LUT; a bounding region delimited by corners $(i_{KMIN} - d, j_{KMIN} - d)$ and $(i_{KMAX} + d, j_{KMAX} + d)$, where d is pre-determined number of pixels, is defined around the potential red-eye segment; each pixel within this bounding region is now tested to determine if it is a skin pixel.

A pixel is classified as "skin" if the minimum Euclidean distance in *Lab* coordinates to one of a set of predetermined skin-prototypes is lower than a given threshold. This "skin pixel" decision is taken based on the use of predetermined skin prototypes. If the skin-pixel ratio, E_K , in the bounding box is lower than a predefined limit, E_{MIN} , then the segment S_K is eliminated. Otherwise the segment S_K is validated for the skin-pixel test and retained in the LUT.

2) Eye-White Characteristics of Bounding Region

The bounding region of each segment is also tested for the presence of any whitish pixels. If there are sufficient whitish pixels the segment is validated. The exact threshold for validation depends on a number of determining factors including the segment size, the local luminance and contrast of the bounding region and the overall redness of a segment; otherwise the segment is rejected and the next potential red-eye segment is loaded from the LUT.

Note that for reasons of computational efficiency this particular test is normally split into several separate tests, each of which is applied when an appropriate determining factor is computed.

3) Contrast & Texture Analysis of Bounding Region

The final test step in this embodiment is the elimination of the surviving segments having low internal contrast. The contrast C_k within the a segment S_k is defined by the ratio:

$$C_k = \frac{L_{kMAX} - L_{kMIN}}{L_{kMAX} + L_{kMIN}}$$

where L_{KMIN} and L_{KMAX} are the minimum and maximum values of the luminosity, or *L* component within the bounding box for the segment S_K . As before one of the surviving segments is loaded from the LUT and the maximum and minimum values of the luminance are calculated by iterating through the pixels of this segment and its bounding region. The internal contrast of the segment is then calculated. Only segments S_K with $C_k > C_{min}$, where C_{min} is a predetermined limit for internal contrast in a segment, are retained. Segments that fail this test are eliminated as potential red-eye segments.

Any segments that remain after completing this final elimination test are marked as valid. The algorithm continues to check each segment remaining in the LUT until all segments

have been tested. The segments which are retained in the LUT and thus have survived all of the elimination steps are now passed on to the color correction stage of the process.

Some typical results for execution time and memory requirements for the algorithm are described in the next section.

IV. MEMORY REQUIREMENTS & EXECUTION TIMES

A statistically mixed batch of 500 flash-defect images have been used for this analysis. The detection has been performed on down-sampled image data of 1 megapixel (1024×768 pixels) and the correction is applied to the full resolution image data - typically 3.1 megapixels (2048×1600 pixels).

A. Working Memory Usage

Working memory size varies depending on the complexity of the image data; however it does not grow significantly with an increase in the image size. This is possible because the design of the flash-defect detection algorithm is such that it eliminates 95-98% of image data on its first segmentation pass.

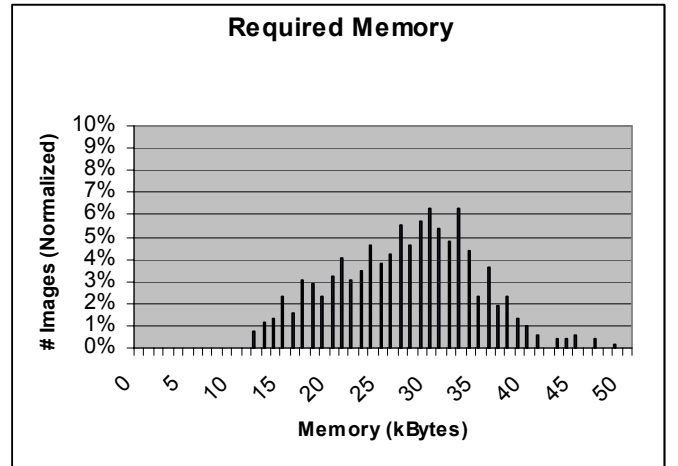


Fig 2: Working Memory Requirement of the Algorithm

Average Memory	32.62 kB
Standard Deviation	12.84 kB
Skew	3.94 kB
Minimum Memory	16 kB
Maximum Memory	150 kB

Table 1: Working Memory Usage for the Algorithm

B. Code Memory Requirements

Typical program memory size for a general purpose operating system is between 30KB and 70KB. The size varies slightly based on the target system environment and supported image format(s).

C. Execution Time Requirements

We have performed execution timing tests for the same set of images on a number of target CPUs. The results presented

here are for a 74MHz ARM 7 which is typical of a low-end consumer camera, with an average execution time of 1.84 seconds and a 266 MHz ARM 9 with an average execution time of 0.24 seconds. Execution times on a desktop PC are an order of magnitude faster.

1) ARM 7 (74MHz, 32 bit memory bus)

The following data represents the execution time for a typical embedded device based on an ARM7 processor, running at 74MHz. The system memory is 32 bit memory. The execution time will of course vary depending on the memory speed, size of cache, etc, of the target embedded system.

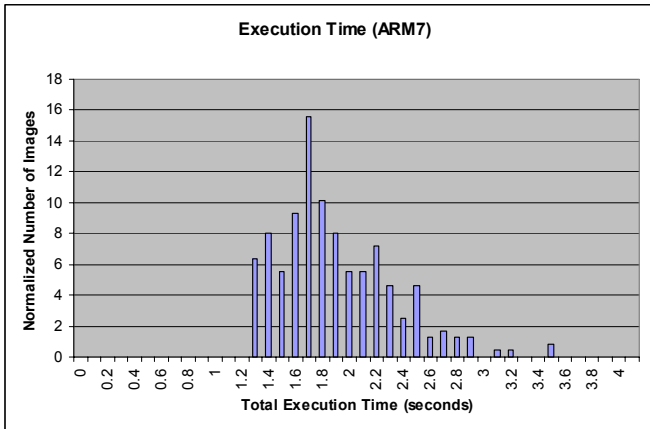


Fig 3: Execution Times ARM 7 CPU

Average (seconds)	1.84
Standard Deviation	0.43
Skewness	0.98
Minimum (seconds)	1.2
Maximum (seconds)	3.5

Table 2: Execution Time Statistics for ARM 7 CPU

2) ARM 9 (266MHz, 32 bit memory bus)

The following data represents the execution time for a typical embedded device. The used processor is an ARM9, running at 266MHz. The execution time will of course vary depending both on the image being analyzed and the nature of the target embedded system.



Fig 4: Execution Times ARM 9 CPU

Average (seconds)	0.24
Standard Deviation	0.08
Skewness	2.19
Minimum (seconds)	0.14
Maximum (seconds)	0.72

Table 3: Execution Time Statistics for ARM 9 CPU

REFERENCES

- [1] K. Acker, D. Bien, and D. Lawton, "Automated removal of red eye effect from a digital image", US patent 6,009,209.
- [2] N. Gupta, "System and method for adjusting color data of pixels in a digital image", US patent 6,204,858.
- [3] J. Schildkraut and R. Gray, "Computer program product for redevye detection", US patent 6,252,976.
- [4] J. Wang and H. Zhang, "Apparatus and a method for automatically detecting and reducing red-eye in a digital image", US patent 6,278,491.
- [5] M. Gaubatz and R. Ulichney, "Automatic redevye detection and correction", *ICIP-2002 Proceedings*, p804-807.
- [6] L. Xin, X. Yanjun, and D. Limin, "Locating facial features with color information," in *ICSP '98 Proceedings*, 1998, vol. 2, pp. 889-892.
- [7] T. Kawaguchi, D. Hikada, and M. Rizon, "Detection of the eyes from human faces by hough transform and separability filter," in *ICIP 2000 Proceedings*, pp. 49-52.
- [8] N. Jarman, R. Lafferty, M. Archibald, M. Stroud, N. Biggs, and D. Normington, "The detection and correction of redevye features in digital images", PCT patent application WO 03/071781.



Peter Corcoran received the BAI (Electronic Engineering) and BA (Math's) degrees from Trinity College Dublin in 1984. He continued his studies at TCD and was awarded a Ph.D. in Electronic Engineering for research work in the field of Dielectric Liquids. In 1986 he was appointed to a lectureship in Electronic Engineering at NUI, Galway. His research interests include microprocessor applications, home networking,

digital imaging and wireless networking technologies. He is a member of IEEE.



Petronel Bigoi received his B.S. degree in Electronic Engineering from "Transylvania" University from Brasov, Romania, in 1997. At the same university he received in 1998 M.S. degree in Electronic Design Automation. He received a M.S. degree in electronic engineering at National University of Ireland, Galway in 2000. From Sept. 2001 he is a lecturer at NUI, Galway. His research interests include VLSI design, communication network protocols, digital imaging

and embedded systems.



Eran Steinberg received his BSc. in mathematics from HU, Jerusalem, Israel. He received a MSc. in Imaging Science from RIT, Rochester NY. With a vast experience of over 15 years in industry, project leader of ISO/WG18/TC42/IT10 - 15740, 8 granted patents and over 20 pending patents, he is currently CTO of FotoNation. His research interests include digital image processing and connectivity.



Alexei Pososin received his BS degree in computer science from Moldova State University. With an experience of eight years in embedded systems and telecommunications he currently is working as R&D senior research engineer at FotoNation Ireland Ltd. .