Automatic Red-Eyes Detection Based on AAM*

Jian Wan, XuePing Ren
Software School
Hangzhou Dianzi University
Hanghzhou, PR. CHINA
wanjian@software.hziee.edu.cn
lilyrxp@sina.com.cn

Guanghuan Hu
College of Computer Science
Zhejiang University
Hanghzhou, PR. CHINA.
huguanghuan@hotmail.com

Abstract - *Red-Eye is a phenomenon that the pupils of people appear unnaturally red when an image is captured using photoflash lamp. There are some algorithms existing for red-eyes detection, but all of them have less accuracy and cannot detect single red-eye. In this paper, a novel approach is proposed to automatically detect red-eyes in photos, which is based on Active Appearance Models, and a practicable automatic correction method is presented. The goal of the algorithm is to detect all the red-eyes in photos without any human intervention. Finally, experimental results demonstrate the high accuracy and efficiency of the method.*

Keywords: Red-eye, eyes detection, Active Appearance Model, correction.

1 Introduction

Red-eye is a special phenomenon in flash photography, which is the appearance of an unnatural red hue around a person's pupils. When a flash is needed to illuminate a people, the ambient illumination is generally low, and the people's pupils will be dilated. Light from the flash can thus reflect off the blood vessels in the people's retina and appear red in color. If the flash illuminates the people's eyes with an angle, and the camera lens is sufficiently small, this reddish light will be recorded by the camera., the red-eye is produced.

Up to now, there are a few results [4][5] published specific to automatic red-eye detection. However, a great deal of research has been done on the detection of human eyes [2][3][6] and facial feature tracking [1]. A general approach to eye detection is to divide the task into two parts, 1) the detection of faces within an image and 2) the detection of eyes within a face. At the same time, many methods tried to locate the eye regions directly by using color or intensity information [2][6] and some geometrical facial features [3]. They can only either handle intensity images or detect pairs of eyes automatically. Our method is based on AAM(Active Appearance Models) [8], which are always hired to tracking facial features [1]. AAM is derived from Active Shape Models, which appears in 1995 and has good performance at represent deformable object's shape. AAM is a statistical model of shape and gray-level appearance of the object of interest, which can be used to locate deformable objects in many applications. During a training phase we learn the relationship between model parameter displacements and the residual errors induuced between a training image and a synthesised model example. To match an image we measure the current residuals and use the model to predict changess to the current parameters, leading to a better fit. A good overall match is obtained in a few iterations, even from poor starting estimates.

1.1 Detection of Eyes

In paper [1], Ashish Kapoor and Rosalind W. Picard presented an approach to track eyes and eyebrows in real time. In their system, the pupils were tracked using the red eye effect by infrared LEDs. And a template was used to parameterize the eyes and eyebrows. For each frame, the pupil coordinates were used to extract cropped images of eyes and eyebrows. Tracking results were satisfactory, but they mainly depended on the hardware.

In paper [2], they proposed a new algorithm to detect eyes only in grayscale images. Though their algorithm is robust, it is not practical.

In paper [3], the Japanese proposed a novel method to extract region of eyes out of facial image. But their approach is too complex.

In paper [4] and [5], a complete system for red-eye detection was implement, but for commercial confidence, the authors descriptions were too simple.

R. Thilak Kumar et al [6] proposed a heuristic approach for detection of eyes in close-up images. They employed a hierarchical search space reduction technique to localize possible eye areas in the image. The distinct human skin color and low intensity areas of the eyeballs are the primary cues used to locate eye regions. Further on, eye validation is performed using the mean and variance projection functions. Their flaw is that they can only work with close-up image.

* 0-7803-8566-7/04/$20.00 © 2004 IEEE.

1.2 Correction of Red-eye

Jay S. Schildkraut et al [4] and Matthew Gaubatz *et al* [5] proposed red-eye's correction algorithm both. Jay S. Schildkraut *et al* removed the red color of redeye defect while maintaining the natural appearance of the pupil. The red color is removed and the luminance of the pupil was lowered. Next, the corrected region is blended with its neighboring region. Finally, the region is re-grained. It is important to preserve the glint in the redeye correction process in order to preserve the natural appearance of the pupil. Their algorithm appears too simple and conceptual.

On the other hand, Matthew Gaubatz *et al* proposed another correction algorithm. After a candidate red-eye location is detected, the artifact can be corrected by removing the color of all *excessively red* pixels within an eye-sized neighborhood. Handle of the artifact in this manner will remove the redness, but can introduce other undesirable artifacts. If pixels are de-saturated based only their corresponding redness values, reddish pixels outside the eye could be de-saturated. Also, a patch of eye pixels de-void of color can look even more unnatural than a patch of slightly reddish pixels, especially if the picture has a high mean luminance with a small dynamic range of color values. Finally, a visible hard boundary between corrected and uncorrected pixels can have a displeasing look. This algorithm has something we can adopt.

Nevertheless, the correction algorithm has some distance to practical application.

2 Algorithm Overview

There are mainly two approaches for red-eye detection. One is using low-level image features to search potential red-eye regions, such as histogram and geometry information. Their evidences are always incomplete because low-level features could not fully represent the objects. Another one is using the template-based models to locate the objects. Though that method is robust enough, it can only locate single interesting object in an image. Our work is to integrate the two methods into a unitary framework and benefit from the characteristics of both.

2.1 First Step

Firstly, we use some typical red-eyes images of different kinds to construct our single red-eye's active appearance model. This is a manual training process. We label the single red-eye and eyebrow with 28 points, 7 points for eyebrow, 12 points for the boundary of eye, 8 points for the pupil and 1 for the "glint" in the pupil.

2.2 Second Step

Secondly, we use color cues to locate potential red-eye regions. We analyze the whole image and get several rectangles that potentially contained the red-eyes by searching the most possible red-eye pixels.

2.3 Final Step

Finally, the matches between our single red-eye model and real image are performed in each rectangle. That is an AAM search procedure. Therefore, low-level features as well as deformable template are both utilized by our algorithm. Small image rectangle can increase the search speed of AAM. Furthermore, as we construct single red-eye's active appearance model, we can locate not only pairs of red-eyes, but also single red-eyes.

2.4 The benefits of our algorithm

The benefits of our algorithm are that we can achieve high speed by getting several small rectangles from the image and high accuracy using AAM. Experimental results show that our algorithm is very robust in different environment.

2.5 The limitation of our algorithm

The limitation of our algorithm is that the accuracy of detection is affected by the quality of single red-eye's active appearance model. So in the future we will do our best to construct an all-purpose single red-eye model.

2.6 Architecture of our algorithm

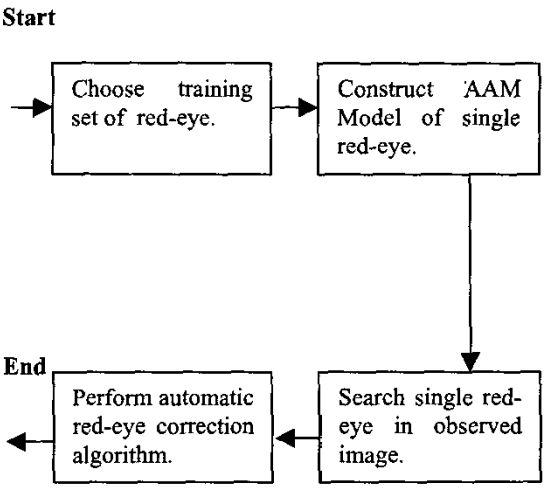

Figure 1. The Flow of our algorithm

3 Construction of Model

Following the Active Shape Model algorithm[7], our statistical appearance model of single red-eye is generated by combining a model of shape variation with a model of texture variation. By *texture* we mean the pattern of intensities or colors across an image patch. To build a model, we require a training set of annotated images where corresponding points have been marked on each example.

For instance, to build a red-eye and eyebrow model, we require face image marked with points defining the main structure of single red-eye (Fig. 2). We apply Procrustes analysis to align the sets of points (each represented as a vector, x) and build a statistical shape model. We then warp each training image so that the points match those of the mean shape, obtaining a *shape-free patch*. This is raster scanned into a texture vector, g, which is normalized by applying a linear transformation,

$$g \to (g - \mu_g 1)/\delta_g ,$$

where 1 is a vector of ones, μ_g and δ^2 are the mean and variance of elements of g. After normalization,

$$g^T * 1 = 0 \quad \text{and} \quad |g| = 1 .$$

Eigen-analysis is applied to build a texture model. Finally, the correlations between shape and texture are learned to generate a combined appearance model.

The appearance model has parameters, c, controlling the shape and texture (in the model frame) according to

$$x = \bar{x} + Q_s c$$
$$g = \bar{g} + Q_g c ,$$

where $\bar{x}$ is the mean shape, $\bar{g}$ is the mean texture in a mean shaped patch, and Q_s, Q_g are matrices describing the modes of variation derived from the training set.

A shape, X, in the image frame can be generated by applying a suitable transformation to the points, $x : X = S_t(x)$. Typically, S_t will be a similarity transformation described by a linearity, we represent the scaling and rotation as (s_x, s_y), where $s_x = (s\cos\theta - 1)$, $s_y = s\sin\theta$. The pose parameter vector $t = (s_x, s_y, t_x, t_y)^T$ is then zero for the identity transformation and $S_{t+\delta t}(x) \approx S_t(S_{\delta t}(x))$.

The texture in the image frame is generated by applying a scaling and offset to the intensities, $g_{im} = T_u(g) = (\mu_1 + 1)g_{im} + \mu_2 * 1$, where μ is the vector of transformation parameters, defined so that $\mu = 0$ is the identity transformation and $T_{u+\delta u}(g) \approx T_u(T_{\delta u}(g))$.

A full reconstruction is given by generating the texture in a mean shaped patch, then warping it so that the model points lie on the image points, X [8].

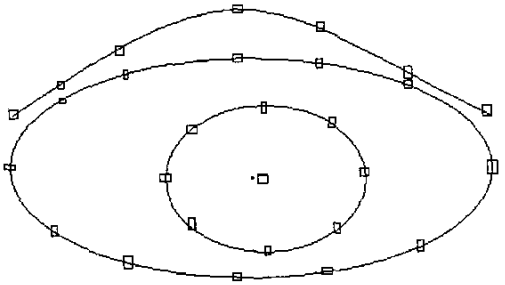

Figure 2. Main structure of single red-eye

4 Red-eye Detection

After the Active appearance model of single red-eye is constructed, we can use AAM algorithm to locate red-eye in an observed image.

We now address the central problem: We have an image to be searched, a full appearance model as described above and a reasonable starting approximation. We treat interpretation as an optimization problem in which we minimize the difference between a new image and one synthesized by the appearance model. A difference vector δI can be defined:

$$\delta I = I_i - I_m ,$$

The simplest model we can choose for the relationship between δI and the error in the model parameters is linear:

$$\delta_c = A\delta I$$

The training algorithm is then simply to randomly displace the model parameter in each training image, recording δ_c and δ_g. We then perform multi-variate regression to obtain the relationship:

$$\delta_c = A\delta_g$$

The best range of values of δ_c to use during training is determined experimentally. Ideally we seek to model a relationship that holds over as large a range errors, δ_g, as possible. However, the real relationship is found to be

linear only over a limited range of values. Our experiments on the face model suggest that the optimum perturbation is around 0.5 standard deviations (over the training set) for each model parameter, about 10% in scale and 2 pixels translation.

We can visualize the effects of the perturbation as follows. If a_i is the i^{th} row of the regression matrix A, the predicted change in the i^{th} parameter, δ_{c_i} is given by

$$\delta_{c_i} = a_i \bullet \delta_g,$$

and a_i gives the weight of attached to different areas of the sampled patch when estimating the displacement.

Given a method for predicting the correction that needs to be made in the model parameters, we can construct an iterative method for solving our optimization problem.

Given the current estimate of model parameters, c_0, and the normalized image sample at the current estimate, g_s, one step of the iterative procedure is as follows:

I. Evaluate the error vector, $\delta_{g_0} = g_s - g_m$;

II. Evaluate the current error, $E_0 = \left|\delta_{g_0}\right|^2$;

III. Compute the predicted displacement, $\delta_c = A * \delta_{g_0}$;

IV. Set k = 1, let $c_1 = c_0 - k\delta_c$;

V. Sample the image at this new predication, and calculate a new error vector, δ_{g_1};

VI. If $\left|\delta_{g_1}\right|^2 < E_0$, then accept the new estimate, c_1;

VII. Otherwise try at k = 1.5, k = 0.5, k = .025, etc.

The procedure is repeated until no improvement is made to the error, $\left|\delta_{g_1}\right|^2$, and convergence is declared. We can use a multi-resolution implementation, in which we iterate to convergence at each level before projecting the current solution to the next level of the model. This is more efficient and can converge to the correct solution from further away than search at a single resolution. Now we can detect *single red-eye* in an observed image if red-eye existed.

5 Red-eye Correction

We mainly use the color cues to refine the red-eyes. Several steps can be taken to improve the visual quality of the corrected image.

Eye boundaries are marked by changes in luminance. Thus, pixels associated with a significant change in luminance are removed from the edge of the corrective mask, to prevent the de-saturation of non-red-eye pixels. Undesired effects resulting from hard color de-saturation are mitigated by de-saturating pixels by an amount proportional to their redness. Perceived edges that are due to hard decision boundaries between corrected pixels and uncorrected pixels are reduced by tapering the boundaries. This process yields a patch centered on the red-eye artifact, with a *correction factor* computed for each pixel. The correction factor, $p(x, y) \in [0,1]$, is the percentage by which the color of pixel in the patch is de-saturated:

$$Y(x,y)_{corrected} = Y(x,y)_{orig};$$

$$Cr(x,y)_{corrected} = (1 - p(x,y)) * Cr(x,y)_{orig};$$

$$Cb(x,y)_{corrected} = (1 - p(x,y)) * Cb(x,y)_{orig}.$$

6 Experimental Results

A detector implemented with our algorithm is tested on a collection of 300 consumer images with faces, about half of which included subjects of red-eye. These images varied in quality and size. The algorithm successfully detected about 98% of the red-eye artifacts from image in the test samples. Compared to previous algorithms, our algorithm solves the problem of single red-eye's detection and improves the accuracy of detection.

7 Conclusions

A robust and high-performance algorithm is proposed in this paper. Experimental results show most of the artifacts missed by the algorithm occurred in the case that the red-eyes can not be recognized by human eyes. The algorithm utilizes both low-level features of red-eyes and template-based model to detect red-eyes in consumer images. It is highly robust so that it can detect most red-eyes in many applications.

References

[1] A. Kapoor and R. W. Picard. "Real-Time, Fully Automatic Upper Facial Feature Tracking", In Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition (FGR'02).

[2] H. Han, T. Kawaguchi and R. Nagata. "Eye detection based on grayscale morphology", In Proceedings of IEEE TENCON'02.

[3] H. Tani, K. Terada, S. Oe and J. Yamaguchi, "Detecting of One's Eye from Facial Image by using Genetic Algorithm", In Proceedings of IECON'01.

[4] J. S. Schildkraut and R. T. Gray, "A Fully Automatic Redeye Detection and Correction Algorithm", In Proceedings of IEEE ICIP 2002.

[5] M. Gaubatz and R.Ulichney, "Automatic Red-Eye Detection and Correction", In Proceedings of IEEE ICIP 2002.

[6] M. Gaubatz and R.Ulichney, "Automatic Red-Eye Detection and Correction", In Proceedings of IEEE ICIP 2002.

[7] R. T. Kumar, S. K. Raja and A. G. Ramakrishnan, "Eye Detection using color cues and projection functions", In Proceedings of IEEE ICIP 2002.

[8] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active Shape Models---Their Training and Application," COMPUTER VISION AND IAMGE UNDERSTANDING, VOL. 61, NO.1, Jan, pp. 38-59, 1995.

[9] T. F. Cootes, G. J. Ewards, and C. J. Taylor, "Active Appearance Models", IEEE TRANSACTIONS ON PATTERN AND MACHINE INTELLIGENCE, VOL. 23, NO. 6, JUNE 2000.