International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 2, May 2014, PP 58-63

Red Eye Detection and Correction

Ms. P. Revathy
Assistant Professor
Department of Computer Science and Engineering
revathy_engg@yahoo.com
Dr.Sivanthi Aditanar
College of Engineering, Tiruchendur.

Abstract: *Flash light passing through pupil, while taking the photographs. It is reflected on the blood vessels, and arrives at a camera lens. This phenomenon makes red-eyes in photographs. Several algorithms have been proposed for removal of red-eyes in digital photographs, but the corrected eyes are looking unnatural. This paper proposes a red-eye removal algorithm using in painting and eye-metric information, which is largely composed of two parts: red-eye detection and red-eye correction. For red-eye detection, frontal face image is taken as input. Next, red-eye regions are segmented in the face regions using multi-cues such as redness, shape, and color information. By region growing, we select regions, which are to be completed with iris texture by in painting method. Then, for red-eye correction, pupils are painted with the appropriate radii calculated from the iris size and size ratio. Experimental results with a large number of test photographs with red-eye effect show that the proposed algorithm is effective and the corrected eyes look more natural than those processed by the conventional algorithms.*

1. Introduction

When taking pictures in dark lighting conditions, people use flash light to make objects visible. However, flash light causes red-eye effect. Flash light passing through enlarged pupils is reflected on the blood vessels inside eyes, and arrives at a camera lens. This flash light makes eye in the photograph red. Red-eye removal methods are categorized into two classes: physics-based and software-based. A physical-based method the red-eye effect can be avoided by changing the lighting condition or by increasing the distance between a camera lens and flash. A software-based method post-processes digital photographs using algorithm to remove existing red-eye in them.

2. Red Eye Effect

This section explain the cause of red-eye effect first and describe the methods of handling red-eye effect.

2.1. Cause of Red-eye Effect

Red-eye effect results from flash light of a camera. When it is dark, pupils of eyes are enlarged. Through the enlarged pupils, flash light from a camera passes the eye lens and is reflected back. This reflected light is blood-color since it is reflected from the blood vessels inside an eye. Fig. 1 describes the geometric situation in which red-eye effect occurs. The light reflected from the eye composes the red-eye beacon. If a camera lens which receives the light is located inside the red-eye beacon, pupils in the picture look red. In other words, red-eye effect occurs if

$x<r_c/2$ **(1)**

Where x is the distance between a camera lens and flash light, and r_c is the radius of the red-eye beacon at the camera position is shown in figure 1.

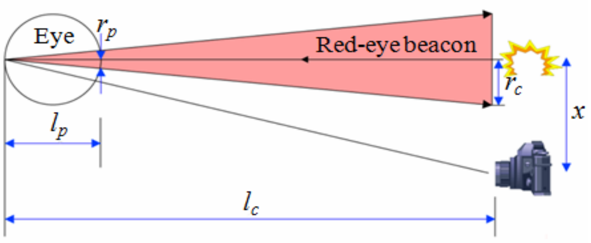


Fig1. *Geometry of red-eye reflection*

$r_c = r_p \times l_c / l_p$ (2)

where l_p, l_c, and r_p denote the diameter of eye sphere, the distance between an eye and a camera, and radius of pupil, respectively. From equation (1) and (2), it is noted that red-eye effect can occur more frequently when x is small and l_c and r_p are large.

2.2. Methods for Red-Eye Removal

First one is to prevent the red-eye by avoiding the situation physically that satisfies the inequality in (1). We can increase the distance between flash and lens (x) so that the lens is located outside the red-eye beacon. However, the size of a camera should be large enough. Or we can also make the size of pupil (r_p) small by using pre-exposure flash. Camera will consume much power and people feel annoyed when seeing the pre-exposure flash. Second, red-eye effect can be removed by software algorithms.

3. Proposed Red-Eye Detection

This section describes a proposed red-eye detection algorithm. The proposed red-eye detection algorithm is composed of three modules: face detection, redeye detection, and region-growing. When an image, I_{in}, is given as input, faces are detected and eye regions, R_f are defined. Then, red-eye regions, R_r, are searched inside R_f. Red-eye regions are expanded to R_r' by a region-growing method for the next step of red-eye correction.

3.1. Face Detection

Initially, the frontal face region is to be taken. Eyes are located in specific area of the face region. When we divide the face region into 20 cells, in four rows and five columns, we observe that two eyes are mostly placed in the second-row, as shown in Figure 2.

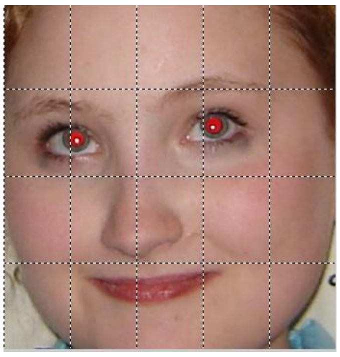


Fig2. *Eye region*

3.2. Red-Eye Detection

Red-eyes are detected by using features of red-eye extracted from the eye region Figure 3 shows an example of the red-eye detection process. B

3.2.1. Redness detection

Redness is the primary feature for red-eye detection. Redness is defined as

$$redness(x,y) = \frac{R^2(x,y)}{G^2(x,y)+B^2(x,y)+K_r} \quad (3)$$

Where R, G, and B denote red, green, and blue components, respectively, and k_r is a constant to avoid zero denominator. The pixel at (x, y) is considered red if

$$redness(x, y) > \tau_{r1} \quad (4)$$

Where the value of redness threshold, τ_{r1} is adjusted experimentally. Figure 3(b) shows the result of redness detection from an eye region in Figure 3(a).

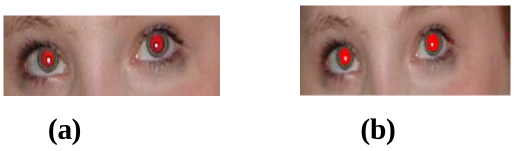


Fig3. *An example of red-eye detection process of the proposed algorithm. (a) Eye candidate region, (b) Redness detection*

Since human skin often contains red component much, redness is calculated high in skin region using equation 5,6 & 7. So there are many false alarms in skin region. These false alarms can be reduced by eliminating skin color region. $YC'_gC'_r$ spaces are proposed for face detection and RGB color space is converted to $YC'_gC'_r$ as follows

$$\begin{bmatrix} Y \\ c_g' \\ c_r' \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.317 & 0.438 & -0.121 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (5)$$

$$C_g' = c_g'\cos30^0 + c_r\sin30^0 - 48$$
$$C_r' = c_g\sin30^0 + c_r\cos30^0 + 80 \quad (6)$$

Skin color is in the range of,

$$123 \le c_g' \le 140 \wedge 136 \le c_r' \le 217 \quad (7)$$

in $YC'_gC'_r$ color space.The resultant skin color removal region is shown in the figure4

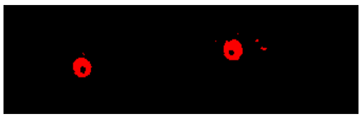


Fig4. *Skin color removal*

3.2.2. Closing operation

Since previous two steps are pixel wise, result regions are scattered and difficult to handle. Using closing morphology operation, neighboring regions are merged and tiny holes in the regions are filled. The regions become simple and easy to handle by closing operation, as shown in Figure 5

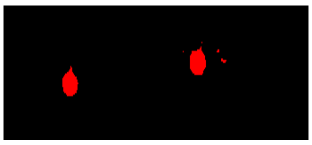


Fig5. *Closing operation*

3.2.3. *Labeling*

Then, connected regions are labeled by sequential connected components algorithm with 4-connectivity .Using labeling, we distinguish each connected component and take advantage of the information of size and shape of each component. Figure 6 shows the result of the labeling, in which each component region has different color.

Fig6. *Labeling*

3.2.4. *Shape filtering*

In order to reduce false alarms, three features (region's width and height, ratio of width to height, compactness) are used, defined as follows in equation (8) & equation (9)

$$k_w^{\min} W \leq w_i \leq k_w^{\max} W$$
$$k_h^{\min} H \leq h_i \leq k_h^{\max} H \qquad (8)$$

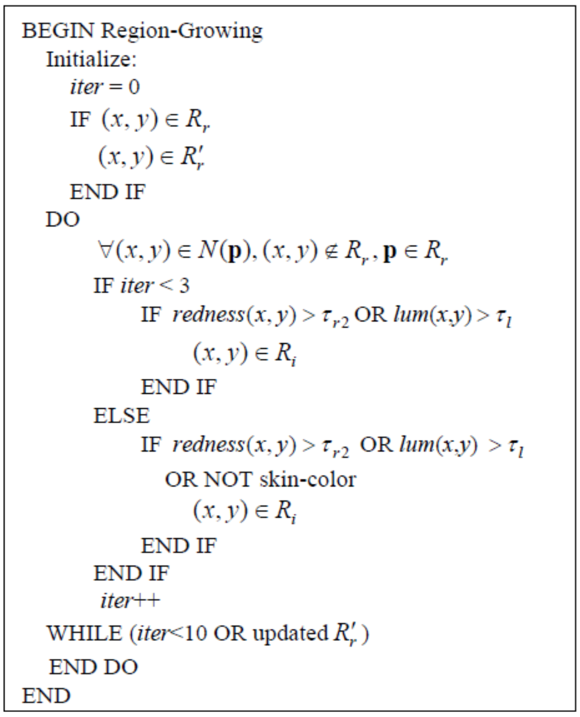
$$\frac{1}{2} \leq \frac{w_i}{h_i} \leq 2$$
$$a_i \geq \frac{w_i h_i}{2} \qquad (9)$$

where w_i and h_i denote the width and height of the connected component of i-th label, with W and H signifying the width and height of the face region, to which the component belongs. $k_w^{\min}$, $k_h^{\min}$, $k_w^{\max}$, $k_h^{\max}$, and are the constants signifying the range of possible pupil size relative to the width and height of a face region. The constants are selected as $k_w^{\min} = k_h^{\min} = 1/50$, $k_w^{\max} = k_h^{\max} = 1/12$. a_i is area of the connected component of i-th label. Most of the false alarm are removed by shape filtering is shown in the figure 7.

Fig7. *Shape filtering*

3.3. Region-Growing

We use a region-growing algorithm as shown in figure 8, so that the region fits to the next process-in painting. In the previous red-eye detection stage, some red-eye regions may be excluded by the threshold, τ_{r1} if redness is weak. Red-eye regions need to be grown to cover all the regions which appear red in comparison to the original color of iris. If not, results of inpainting contain red pixels in iris region and become unnatural.



```
BEGIN Region-Growing
  Initialize:
    $iter = 0$
    IF $(x, y) \in R_r$
      $(x, y) \in R'_r$
    END IF
  DO
      $\forall (x, y) \in N(\mathbf{p}), (x, y) \notin R_r, \mathbf{p} \in R_r$
      IF $iter < 3$
          IF $redness(x, y) > \tau_{r2}$ OR $lum(x,y) > \tau_l$
              $(x, y) \in R_i$
          END IF
      ELSE
          IF $redness(x, y) > \tau_{r2}$ OR $lum(x,y) > \tau_l$
            OR NOT skin-color
              $(x, y) \in R_i$
          END IF
      END IF
      $iter$++
  WHILE ($iter$<10 OR updated $R'_r$)
  END DO
END
```

Fig8. *Pseudo-code for region growing*

After applying the region growing algorithm, the resultant of region growing area is shown in the figure 9.

Fig9. *Region growing of red-eye*

Resultant of Red Eye Detection

In STOIK, the detection rate of red eye is 92.22% and false alarm rate is 24.44%.But in the proposed System the detection of red eye rate is increased as 93.33% and false alarm rate is decreased to 13.3%.The comparison results for Red Eye Detection are shown in the Table II.The performance is evaluated by Detection rate and number of false alarms.Detection rate is determined by the ratio of the number of correctly found red-eyes to the total number of eyes. Red eye images are used to determine the detection rate.Non red eye imaged are used to test the false alarms. And if the algorithm detects non red-eye as red-eye,the count of false alarm increases is shown in the TABLE I.

Table1. *Red Eye Detection Comparison*

Algorithm	**Detection rate(%)**	**Falsealarms**
STOIK	**92.22**	**24.44**

Proposed	**93.33**	**13.3**

4. PROPOSED RED-EYE CORRECTION

The proposed correction algorithm uses a in painting algorithm to correct the red-eye region while other algorithm simply desaturates the red color components.

4.1. Iris Detection

Size of iris depends on person and scale of an input image. Since it is difficult to detect iris directly. It is determined by following steps.

4.1.1. Canny edge detection

Canny edge detector is applied to the red-eye region mask in order to detect the boundary of the red-eye region. Since the mask is binary, clear boundary can be obtained. Figure 10 shows an example.

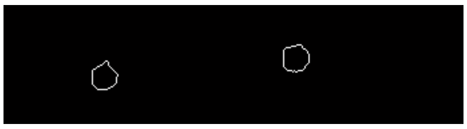


Fig10. *Canny Edge Detector*

4.2. Hough Transformation

Detected red-eye regions often do not have circular shape. We cannot get the accurate center position and radius of the iris from the crooked shape of detected region. Thus, we detect the most similar circle with the red-eye region by using Hough transformation. Figure 11 shows the result of Hough transformation.

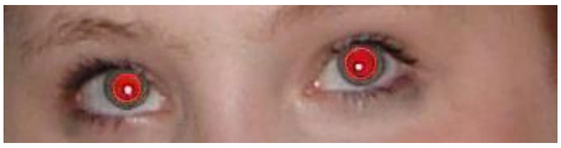


Fig11. *Circle detection by Hough transformation*

4.3. Sclera Detection

At the boundary between iris and sclera, we can find edges by using gradient operation is shown in the equation (10), and obtain the radius of iris. From both sides of iris regions, the pixels with large horizontal gradient are searched vertically, where Sobel masks are used.

We decide that a pixel is in the boundary if,

$$g_{R,v} > \tau_g, \; g_{G,v} > \tau_g, \; g_{B,v} > \tau_g \qquad (10)$$

Where $g_{R,v}$, $g_{G,v}$, and $g_{B,v}$ denote gradient values in red, green, and blue components, respectively, and τ_g signifies the fixed threshold for gradient values is shown in the figure 12.

4.4. Iris Size Calculation

Radius of iris is the distance between the center of the redeye region and the boundary of iris is shown in the figure 13. Therefore, it is sum of red-eye radius and search distance from the red-eye boundary to the iris boundary.

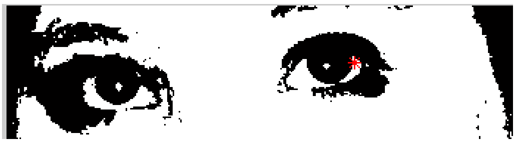


Fig12. *Sclera Detected*

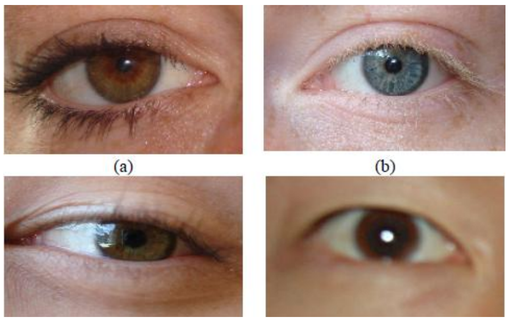


Fig13. *Pupil size in various light environments. (a), (b) direct light, (c) room light, (d) darkness.*

4.5. Pupil Size Calculation

It is important to calculate the accurate size of pupil in order to correct the expanded pupil in dark environment. Pupil size changes according to light conditions as shown in Table II, while iris size is almost constant. Thus, we can calculate appropriate pupil size if we know iris size using Table III. The pupil size in three types of lighting conditions (direct light/room light/darkness) are determine using the equation(11) . Martin and Holden measured horizontal visible iris diameter and reported that its average is 11.64 mm (standard deviation: 0.49 mm). Using these data, we calculate the ratio of pupil size to iris size, as shown in Table III.

Table2. *Pupil Size*

Light Condition	Average Pupil size	Standard deviation
Direct light	3.35	0.72
Room Light	3.86	0.93
Darkness	6.41	1.55

Table3. *Size Ratio of Pupil to Iris*

Light Condition	Size ratio
Direct light	0.2878
Room Light	0.3316
Darkness	0.5507

We calculate the diameter of pupil by using the diameter of iris and size ratio of pupil to iris, which is given by

$d_{pupil} = d_{iris} \times r_{pi}$ (11)

Where r_{pi} denotes the ratio of pupil to iris.

4.5.1. Inpainting

Inpainting is a restoring process of damaged parts of an image. Here, red-eye regions can be thought of as damaged regions or missing regions. The regions lost their own color and texture in part by red-eye effect and expanded pupil in dark environment. However, since we deal with the special case of red-eye regions, the filling order is to be decided differently. Sides of iris and pupil are in touch with sclera while top and bottom of them are covered in part.

R_r denotes the red-eye region with the center at $\mathbf{x}_c$. **u** is the candidate pixel to be inpainted while **v** is one of neighbor pixels of **u**. And $\Psi_\mathbf{u}$ and $\Psi_\mathbf{v}$ are the patches, of which **u** and **v** are the center pixels, respectively. The pixel to be inpainted is chosen among the boundary of red-eye region, ∂R_r , and the priority of the pixel, $P(\mathbf{u})$ is decided by the horizontal and vertical distances between **u** and $\mathbf{x}_c$ as

$$p(u) = \lambda\left|u_x - x_{c,x}\right| - \left|u_y - x_{c,y}\right| \qquad (12)$$

Where, λ is multiplied by the horizontal distance to give more weight and set to 100^0 in our experiment.The most similar patch is searched in the neighbourhood of u.u' is choosen according to

$$u' = \arg\min_{v} D(\psi_u, \psi_v) \qquad (13)$$

$$D(\psi_u, \psi_v) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\left\|c_{ui} - c_{vi}\right\|^2 + \left\|c_{ui} - c_{vi}\right\|^2\right)} \qquad (14)$$

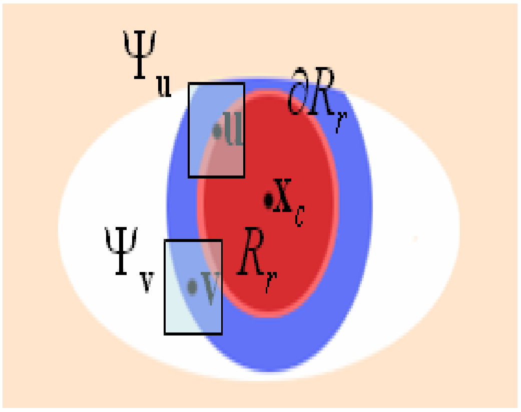


Fig14. *Notation diagram for inpainting of red eye*

The resultant after applying the inpainting is shown in the figure 15

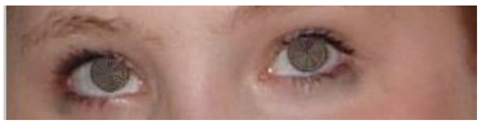


Fig15. *Inpainting*

4.5.2. Pupil painting

There exists specular reflectance by light source in pupil, so pupil looks natural with highlight in it. The position of the highlight depends on the position and direction of light source. When red-eye effect occurs, highlight is located on the center of pupil because the flash light comes from the front side directly. Thus, we paint highlight at the center of pupil with constant color, $\mathbf{c}_h$ ($R = G = B = 255$) and with the size of highlight, $d_{highlight}$, as follows

$f(\mathbf{x}) = \mathbf{c}h$, $\forall\mathbf{x}\in Rh$ (15)

$$R_h = \{X \mid \|X - X_c\| < d_{highlight}/2\} \qquad (16)$$

where $d_{highlight}$, is experimentally selected to quarter of the pupil size. To smooth the boundaries of pupil and highlight, we use a weighted average filter defined as using the Equation (17)

f(x,y)=1/16{f(x-1,y-1)+2f(x-1,y)+

f(x-1,y+1)+2f(x,y-1)+4f(x,y)+2f(x,y+1)

+f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)} (17)

By filtering in iris region, not only pupil and highlight regions but also iris texture look better.

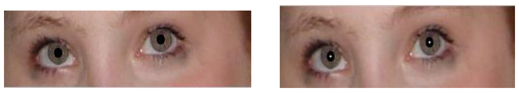


Fig16. *(a)Pupil Painting (b) Final Output*

5. Red Eye Correction

The comparison between Redbot , STOIK and Photoshop and the proposed system are to be shown in the figure 3.13.The Redbot is simply be desaturate the Red region with out retain the original size of the pupil.

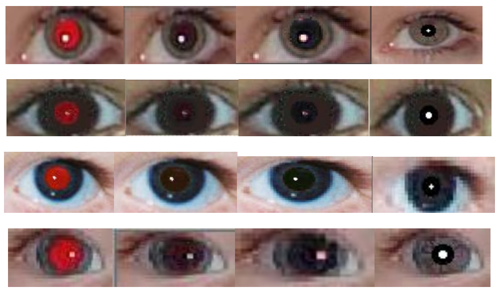


Fig17. *Comparision of Red Eye Correction results*

a)Red Eye b)Redbot c)Photoshop d)Proposed

Using Photoshop we can able to remove the red eye by simply be paint the black color in the enlarged pupil without reducing the enlarged size of the pupil.

So, the corrected eyes are looking unnatural .But the proposed system remove the red eye also retain the original size of the pupil. Thus the resultant of Proposed corrected eye are looking so natural than the existing system is shown in the figure 17.

6. Conclusions

Thus the proposed method for removing the Red Eye is looking so natural than the existing method such as Redbot &Adobe Photoshop CS2. In Redbot, the removal method is simply be desaturate the red eye region. In Adobe Photoshop CS2,the resultant is shown in figure 18 , is simply be paint the expanded pupil region with black color. so the resultant is looking unnaturally .Thus, the proposed resultant gives natural & best result than the existing one.

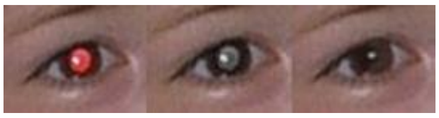


Fig18 *a)Red eye.b)method of redbot c)Proposed Algorithm.*

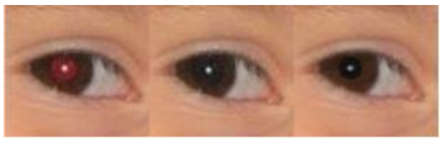


Fig19 *a)Red eye.b)method of Adobe Photoshop CS2 c)Proposed Algorithm.*

References

[1] M. Gaubatz and R. Ulichney, “Automatic red-eye detection and correction,” in Proc. IEEE Int. Conf. Image Processing, vol. 1, pp. 804–807, Rochester, NY, Sep. 2002.

[2] S. Ioffe, “Red eye detection with machine learning,” in Proc. IEEE Int. Conf. Image Processing, vol. 2, pp. 871–874, Barcelona, Spain, Sep. 2003.

[3] J. J. de Dios and N. Garcia, “Face detection based on a new color space YCgCr,” in Proc. IEEE Int. Conf. Image Processing, vol. 3, pp. 902– 912, Barcelona, Spain, Sep. 2003.

[4] J. J. de Dios and N. Garcia, “Fast face segmentation in component color space,” in Proc. IEEE Int. Conf. Image Processing, vol. 1, pp. 191–194, Singapore, Oct. 2004.

[5] Adobe Photoshop CS2, San Jose, CA: Adobe, 2005.

[6] R. Ulichney, M. Gaubatz, and JM Van Thong, “Redbot – a tool for improving red-eye correction”, in Proc. IS&T/SID Eleventh Color Imaging Conference, Nov. 2003.