AUTOMATED RED-EYE DETECTION AND CORRECTION IN DIGITAL PHOTOGRAPHS

Lei Zhang, Yanfeng Sun, Mingjing Li, Hongjiang Zhang

Microsoft Research Asia, 49 Zhichun Road, Beijing 100080, China

ABSTRACT

Caused by light reflected off the subject's retina, red-eye is a troublesome problem in consumer photography. Although most of the cameras have the red-eye reduction mode, the result reality is that no on-camera system is completely effective. In this paper, we propose a fully automatic approach to detecting and correcting red-eyes in digital images. In order to detect red-eyes in a picture, a heuristic yet efficient algorithm is first adopted to detect a group of candidate red regions, and then an eye classifier is utilized to confirm whether each candidate region is a human eye. Thereafter, for each detected red-eye, we can correct it by the correction algorithm. In case that a red-eye cannot be detected automatically, another algorithm is also provided to detect red-eyes manually with the user's interaction by clicking on an eye. Experimental results on about 300 images with various red-eye appearances demonstrate that the proposed solution is robust and effective.

1. INTRODUCTION

Caused by light reflected off the subject's retina, red-eye is a troublesome problem in consumer photography. When using a flash bulb that is immediately adjacent the camera lens, the relative position of the illumination source, the lens and the photographic subjects often results in substantial reflection from the subject's retina. This is exacerbated by dilation of the subject's pupils in response to the ambient illumination conditions that also made the added illumination desirable in order to be able to take the picture. In human subjects, such retinal reflections often result in unnatural reddening of the pupil region of the subject's eyes or a halo of red in that vicinity; in other types of animals, other colors may result. Although most of the cameras have the red-eye reduction mode, the result reality is that no on-camera system is completely effective.

Various types of editing have been employed in the past to address these distortions. Darkroom techniques historically have been employed for photographs using emulsion and other photochemical image recording technologies. Digitization of such images and direct digital image capture led to techniques requiring powerful processors and sophisticated software requiring extensive training and experience of human operators to achieve desired image manipulation goals.

However, the advent of digital photography, coupled with increasingly powerful microprocessors, permit editing of digital images in a personal computer. Some exemplary products for such manipulation are known as "iPhoto", available from Apple, and "Picture Maker", developed by Eastman Kodak Company. These products require user input in order to be able to identify target regions within the image that include red-eye artifacts.

Recently, a number of solutions have been proposed to automatically detect and correct red-eyes [1,2,3]. In general, these algorithms attempt to locate the red-eye regions with the help of a face detector and/or eye detector. In [1], only pairs of red-eyes can be automatically detected. In [2], the algorithm heavily depends on the face detection result, which is another challenging problem in academia. In [3], the detection result is the intersection of learning-based red-eye detector and face detector, and the outline of red-eye is detected by a learning-based red-eye edge detector. Such approach requires a large number of red-eye samples and may not be able to detect red-eyes in small size.

In this paper, a novel solution is proposed to automatically detect and correct red-eyes in digital photographs. In order to detect red-eyes in a picture, a heuristic algorithm is first adopted to detect a group of candidate regions, and then an eye classifier is used to confirm whether the candidate region is a red-eye. For each detected red-eye, an adaptive correction algorithm is utilized to reduce the red-eye effect. In case that a red-eye cannot be detected automatically, another algorithm is also provided to detect the red-eye manually with the user's interaction by clicking on the eye.

2. AUTOMATED RED-EYE DETECTION AND CORRECTION

The proposed red-eye reduction solution consists of three stages. In the *auto detection stage*, image processing and

classification techniques are used to find and mark red-eyes automatically. In the *manual detection stage*, the user is enabled to mark any additional eye by clicking on the eye. In the *auto correction stage*, the marked red-eyes are corrected by adjusting colors automatically.

2.1 Auto-detection stage

Automated detection of red-eyes in a picture is the most difficult task among three stages. In order to find the red-eyes in a picture, we divide the task into two parts, candidate region selection and red-eye confirmation.

2.1.1 Candidate region selection

In this part, a group of red regions can be detected as candidates of red-eyes based on a series of heuristic algorithms, which can be summarized as the following.

Step 1. Find and mark pixels in the picture as red color or non-skin color.

For each pixel in the picture, if the RGB value of the pixel satisfies the following condition, it will be marked as red color pixels.

$$\begin{cases} R > 50 \\ R/(R+G+B) > 0.40 \\ G/(R+G+B) < 0.31 \\ B/(R+G+B) < 0.36 \end{cases}$$

Otherwise, if the RGB value of the pixel satisfies the following condition, it will be marked as non-skin pixels.

$$G/(R+G+B) > 0.40 \text{ or } B/(R+G+B) > 0.45$$

Because all the following procedures will be processed based on this step, these constraints are very loose in order to fit all cases of images and guarantee all the pixels in a red-eye will not be lost.

Step 2. Find and mark highlight pixels in brightness image and use them to restrict red color areas.

In this step, a template is utilized to find the highlight pixels among the red color pixels marked by the previous step. By region growing in the next steps, highlight pixels will be used to form the highlight regions, which represent red-eye regions or highlight part in the center of a red-eye.
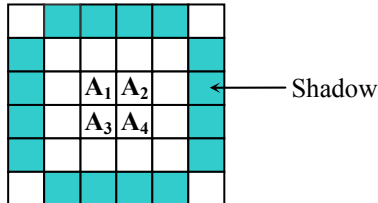

Figure 1. The template for highlight pixels finding

As shown in Fig.1, for position A_1, the difference between certain pixels, A_1, A_2, A_3, A_4, and their neighbor

pixels in a local 6x6 area is calculated in the red channel of the image as the following

$$Diff(A_1) = \frac{1}{4} \times \sum_{i=1,2,3,4} I(A_i) - \frac{1}{16} \times \sum_{B \in shadow} I(B)$$

If $Diff(A_1) > 140$, and the number of red color pixels marked in Step 1 in this 6x6 area is greater than 10, then the pixel A_1 will be marked as the highlight pixel.

Step 3. Group highlight pixels to highlight regions

In this step, the highlight pixels are grouped into a set of regions by the region growing algorithm based on 4-conection principle.

Step 4. Remove invalid highlight regions by checking neighbor regions

The highlight pixels located in a typical red-eye are usually grouped into a single region, whereas some of the highlight pixels located in other place are most likely to scatter and spread everywhere like some noises. Based on this observation, we design a rule to remove those invalid isolated regions.

For each highlight region, we will check its surrounding regions in an extended area by inflating the region with 25 pixels in both width and height. If the number of the pixels, which locate around the highlight region and have the similar color with the region, is greater than 8, the region is removed.

The color similarity is defined as follows:

Let A be the number of pixels belong to the specified highlight region.

$$avePctR = \frac{1}{A} \sum_{i \in Region} \frac{R_i}{R_i + G_i + B_i}$$

$$avePctG = \frac{1}{A} \sum_{i \in Region} \frac{G_i}{R_i + G_i + B_i}$$

$$aveR = \frac{1}{A} \sum_{i \in Region} R_i$$

A pixel p has the similar color with the region, if it satisfies the following conditions:

$$\left| R_p/(R_p + G_p + B_p) - avePctR \right| < 0.03 \text{ and}$$

$$\left| G_p/(R_p + G_p + B_p) - avePctG \right| < 0.03 \text{ and}$$

$$\left| R_p - aveR \right| < 20$$

Step 5. Remove invalid regions by size restriction

In this step, by checking the average color, the size, the shape, those regions, which are too dark, too small, too narrow, too hollow, or too large, will be removed based on a series of empirical rules. For example, the shape of a valid highlight region should be something like a circle or a rectangle, whereas a strip-like region usually indicates the edges of a red object. As well, a valid region should be neither too large nor too small.

Step 6. Remove invalid regions by checking the surrounding non-skin pixels

Let w and h be the width and height of the bounding box of the specified region.

In this step, the surrounding pixels in an extended area by inflating the region with max(w,h)/4 pixels in both width and height are checked for each region. If there are enough non-skin pixels marked in step1 around a region, then the region will be removed. More specifically, if the number of non-skin pixels is greater than ($w*h$)/4, the region is removed.

Step 7. Grow highlight areas into red regions

In the previous steps, a lots of invalid highlight regions are removed and the number of them usually reduced from thousand to hundred. In this step, each highlight region is extended, one pixel by one pixel, to form a precise red-eye region. The red color pixels, which are selected in the step 1 and have similar color with the highlight region, will be extended as a part of the red-eye region.

2.1.2 Red-eye confirmation

In this part, an eye classifier is utilized to confirm each candidate region found in the previous steps. Only those confirmed by the classifier as human eyes will be passed to the *auto-correction* stage.

This part is very important in that most of false positive regions will be removed in this step. Therefore a classifier with the high classification accuracy is trained based on a large number of training samples. The classifier is trained based on AdaBoost method [4] and the training samples are 20x20 gray images. It means that eyes are confirmed based on gray images and thus the difficulty in collecting numerous red-eye images is avoided. We totally collected about 7400 eye images as positive samples, and about 12,000 high resolution non-face images to bootstrap negative samples in the training process. As there are usually tens of red regions that need to be confirmed, the difficulty of eye confirmation here is much easier than eye detection in whole image.

Note that the candidate region is only the red part in a red-eye and each candidate region is represented by a bounding rectangle. Because the ratio between the size of candidate region and the size of whole eye is not consistent, we enlarge the rectangle by 3, 4, 5, 6 and 7 times and verify the eye in these 5 enlarged areas. If any of the enlarged area is classified as an eye, then the candidate region is classified as an eye.

2.2 Manual detection stage

It is generally hard for an automatic detection algorithm based on statistical techniques to handle all red-eye cases. In case that a red-eye cannot be detected automatically, manual detection algorithm will be very useful for users.

In this stage, the user is allowed to click on a red-eye. The search area is constrained to a sub-rectangle, centered at the point clicked by the user. The width and height of the sub-rectangle are set to 1/8 of the whole image. Then the same algorithm as in the *auto-detection* stage is conducted on this sub-rectangle. In order to find the red-eyes which cannot be detected by *auto-detection* stage, the threshold of the eye classifier in the confirmation step is lowered. Therefore, among the rectangles which contain the specified point, the one with the highest confidence score will be outputted as the detection result.

2.3 Auto-correction stage

Correction seems much easier compared to detection, but actually it is difficult to generate a perfect picture. The correction algorithm works on pixel level in the image. The basic idea is to decrease the value of red color to remove the red-eye and increase the value of green color and blue color to remain the highlight point on the eye.

Two operations on pixel level, brightness adjustment and contrast adjustment, are defined as the following.

AdjustBrightness(BYTE c, float factor)
= (BYTE)(0.4 * factor * 255 + c)
AdjustContrast(BYTE c, float factor, BYTE meanL)
= (BYTE)(c + (float)(c - meanL) * factor)

where c is the pixel color in R, G or B channel, factor is the parameter from -1.0 to 1.0, which controls the quantity of adjustment, and meanL is the average luminance of the red-eye region.

First we calculate the factor adaptively according to the whole red-eye region as the following:

factor = (fAvgR - max(fAvgG, fAvgB) / 255 * 3.2

where fAvgR, fAvgG, fAvgB are the average of red, green, blue colors in the red-eye region, respectively, excluding the pixels that do not satisfy the condition $r/(r+g+b) > 0.35$. That is, we only consider the pixels which really appear red.

Then we apply the operations, AdjustBrightness and AdjustContrast, pixel by pixel in the red-eye region as the following:

      rr = AdjustBrightness(r, -factor);
      gg = AdjustBrightness(g, factor / 3);
      bb = AdjustBrightness(b, factor / 3);

      r = AdjustContrast(rr, -10, iMeanLight);
      g = AdjustContrast(gg, -10, iMeanLight);
      b = AdjustContrast(bb, -10, iMeanLight);

where iMeanLight is the average luminance of the red-eye region. From the equations we can see that the value of red color is decreased and the values of green and blue colors are increased. Meanwhile, the contrast in the red-eye region is decreased slightly.

Finally, in order to avoid the sharp edge at the boarder of the rectangle, we blend the modified image with the original image linearly based on the distance from the pixel to the edge. In this way, we cannot discern the edge of the rectangle in the blended image. A perfect picture with red-eyes removed is thus generated.

3. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed solution, we collected about 300 red-eye images from web and personal albums. These test images are not particularly taken for this experiment, and varies in quality and size, with subjects in fontal view and profile view, focused and unfocused. We cropped tens of red-eyes and patched them into an image, as show in Fig.2, from which it can be seen how large the variations of red-eyes are in pose and appearance.

(a) Before correction

(b) After correction
Figure 2. A collection of red-eye samples in the test set

We first manually labeled red-eyes in the test set as the groundtruth. In 300 images, there are total 877 red-eyes, among which 693 severe red-eyes are labeled as MUST repair, and other 184 red-eyes are labeled as inoffensive. If inoffensive red-eyes are detected, we do not treat them as false positives. After *auto detection* stage, about 76% red-eyes are detected, with about 8% false positives. For the miss detected red-eyes, *manual detection* operation is also adopted. As *manual detection* requires only a user clicking on a red-eye, instead of carefully selecting the red-eye bounding box, thus the operation is significantly simplified. Totally, about 95% red-eyes can be detected by either *auto detect* or *manual detection* approach.

As our target is to automatically correct red-eyes, we conducted *auto detection* and *auto correction* together, and asked two subjects to evaluate the correction result. In the test set, about 70% images are completely fixed and about 5% images are partially fixed, i.e. one or more red-eyes are fixed, but there are still some red-eyes left. Interestingly, only five images are fixed with noticeable false positives, because correction of false positives appearing in cluttered background is usually unnoticeable. Fig.2 shows a set of example red-eyes before and after auto detection and correction. It can be seen that the correction is very natural, with highlight points still remained on the eyes.

Though the heuristic search algorithm seems very complex, the detection and correction were significantly optimized and run very fast, taking about 0.8 second on a 1600x1200 image on a 2.7GHz CPU. The speed is two more times faster than other red-eye detection approaches based on face detection, as it unusually takes one or few seconds to detect faces in such an image.

4. CONCLUSION

In this paper, we have presented a fully automatic approach to detecting and correcting most red-eyes in digital photographs, and proposed a one-click manual detection approach to detecting red-eyes missed by auto-detection approach. In the detect stage, a heuristic red region search algorithm is first employed to find out as more as possible candidate red-eyes, and then an eye classifier is adapted to confirm in gray images if each candidate red-eye is an human eye. Thereafter, for each detected red-eye, we can correct it by an adaptive correction algorithm. The experimental result on 300 red-eye images shows that the proposed red-eye reduction solution is robust and efficient.

5. REFERENCES

[1] J.SSchildkraut and R.T.Gray. A fully automatic red-eye detection and correction algorithm. In Proc. *International Conference on Image Processing ICIP*, 2002.

[2] G. Matthew and U. Robert. Automatic red-eye detection and correction. In Proc. *International Conference on Image Processing ICIP,* 2002.

[3] S. Ioffe. Red eye detection with machine learning. In Proc. *International Conference on Image Processing ICIP*, 2003.

[4] P. A. Viola and M. J. Jones, Robust real-time object detection, *Technical report*, COMPAQ CRL, 2001.