Red Eye Removal Algorithm

Dhaya Sagar K (19BCE1726), Shaan Sunder (19BPS1082), ESUKAPALLY BHARGAV VENKATA SIVA (19BCE1628)

VIT University Chennai

**Abstract— The red-eye effect is typically produced by consumer photos taken with the built-in camera flash. Section presents a new algorithm for automatically removing red-eye from digital photography. The proposed algorithm can automatically remove red eyes without manual intervention. First, it uses the Cascade Classifier algorithm to recognize the plane. The red eye is then placed using segmentation operations, morphology, and geometric constraints. Finally, completely corrects the red eye found. The experimental results of the are satisfied with the high correction factor, the relatively low computational complexity of the , and the robustness.**

**I. INTRODUCTION**

When a strong light strikes the eye, the red reflection of the blood vessels in the retina causes red-eye. The pupils' centres appear unnaturally crimson. The usual red eye effect is seen in Figure 1. The redeye effect gets stronger as the flashgun gets closer and closer to the camera lens. As a result, developing an effective red-eye reduction method for digital images is critical.



FIGURE1

There are a variety of red-eye reduction techniques available, including algorithms based on face identification, skin colour feature, and machine learning. There are also various approaches based on the architecture of eyes to directly locate the red-eye . Some existing applications, such as PHOTOSHOP have a redeye-removal function, but they require manual intervention to circle the red-eye areas first, followed by the function correcting the red-eye.

The current tendency is to entirely automate the removal of red eyes from digital photos. However, the majority of currently available automated systems have significant flaws and restrictions. Some algorithms are dependent on the face orientation; several solutions are not only difficult, but also perform poorly . As a result, a completely automated red-eye reduction system with good detection quality and a high corrected rate is required. The study describes a better-performing algorithm for removing "red-eye" from digital photographs automatically.

**II.Objective of the Work**

To make a flawless red-eye removing algorithm

**III.The Proposed Approach**

The following is the flow diagram for the fully automated redeye removal system: First, utilising the Cascade Classifier method to detect faces . Second, techniques including segmentation, close operation, and geometric restriction are used to locate the red-eyes. Finally, entirely fixing the red-eyes that have been found. Face detection, red-eye localization, and red-eye correction are the three primary components of the system. The suggested algorithm's detailed flow block diagram is shown in Figure 2.

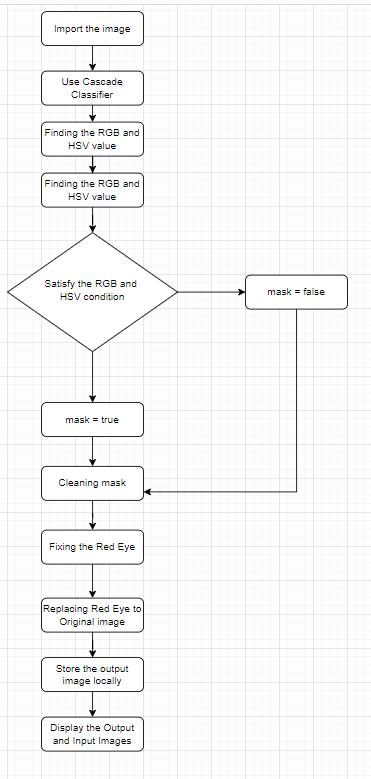


FIGURE 2

**IV.The Proposed Algorithm**

Step 1 : Eye detection :

The first stage is to automatically recognize eyes. For detecting the eyes, we utilize the standard Cascade Classifier for eyes. It's sometimes easier to run a face detector first, then look for the eyes within the face region. We're going to run the eye detector immediately on the image to keep things simple. When the input image is a portrait shot have a closeup of the eyes, skipping the face detector works.

Step 2 : Finding the Red Eyes and Creating a Mask:

The next step is to determine which area of the pupil is impacted by the redness. Finding anything red can be done in a variety of ways. One thing to keep in mind is that our colour is not just red; it might be brilliant red, dark red, or any other red. We transform the image to HSV colour space and apply a hue and brightness threshold.

(r > 60) & (r > bg\_sum) & (s > 10.0) & (v < 70.0) & (sv\_sum < 160.0) & ((15 > h) | (h < 345))

Step 3: Cleaning the mask:

The mask you made in the previous stage is likely to have holes in it. The raw mask obtained by colour processing is shown on the left in the figure3.

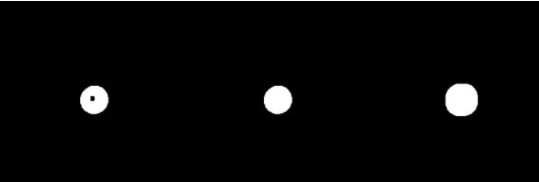


Figure 3

Furthermore, we dilate the mask to cover a somewhat wider area than is required. This is because the color fades away gradually at the boundary, and some redness may have escaped our original mask. The right image in the figure3 is the dilated mask.

Step 4: Fixing the Red Eyes:

We now have a mask that only covers the red part of each eye. We know that red eyes saturate the image's red channel. To put it another way, all data in the red channel is destroyed. Only the red channel is affected by the red-eye phenomenon; the blue and green channels are unaffected. A lost red channel can be created using a mix of the green and blue channels. Next we add some gaussian noise to smooth the eye.

Step 5 : Replace eye region to the original image:

We fixed the three channels in the previous step. The next step is to combine the three channels to generate our RGB image, then re-insert the fixed eye region into the original image.

**V.The Output of The Work**



OUTPUT 1



OUTPUT 2

**V. CONCLUSIONS AND FUTURE WORK**

Automatic red eye removal algorithm has been presented based on the Cascade Classifier algorithm. The experimental results are satisfied with high PSNR rates, relatively low Mean Square Error as mentioned in table1. The next step will focus on improving the performance of feature detection. The results show that the result of the base paper is better than our work but close.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MSE** | **PSNR** | **SSIM** |
| **Base Paper** | 5.452 | 42.633 | 0.9757 |
| **Proposed Algorithm** | 8.536 | 40.466 | 0.9622 |

Table 1

**REFERENCES:**

1. Yueli Cui, Caiming Chen, Shiqing Zhang, Zhigang Chen “A New Algorithm of Fully-automatic Red-eye Removal” JOURNAL OF SOFTWARE, VOL. 8, NO. 12, DECEMBER 2013.
2. <https://www.geeksforgeeks.org/python-peak-signal-to-noise-ratio-psnr/>
3. <https://towardsdatascience.com/image-classification-using-ssim-34e549ec6e12>
4. <https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/>
5. https://stackoverflow.com/questions/2612361/convert-rgb-values-to-equivalent-hsv-values-using-python

1.