Red-eye detection and removal

OpenCV

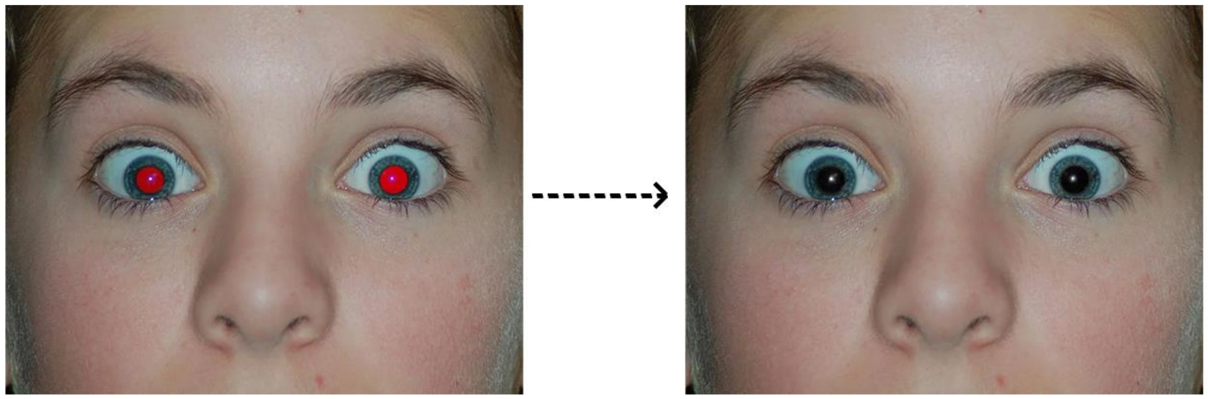
Linca Paul Tudor

E_30432

1. Subject specification

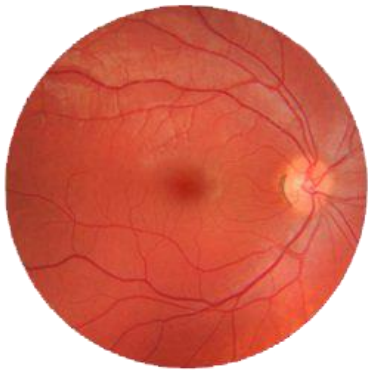
The purpose of this project is to implement a method of detecting and subsequently removing red-eye effects from digital images using the OpenCV library.

The desired effect should look like this:



2. Red – eye effect

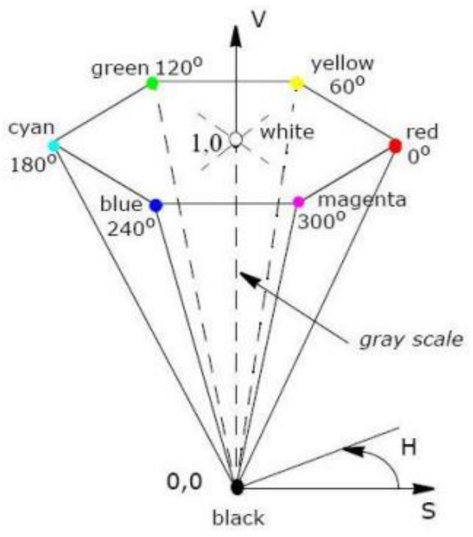
The red-eye effect in photography is the common appearance of red pupils in color photographs of the eyes of humans and several other animals. In flash photography the light of the flash occurs too fast for the pupil to close, so much of the very bright light from the flash passes into the eye through the pupil, reflects off the fundus at the back of the eyeball and out through the pupil. The camera records this reflected light. The main cause of the red color is the ample amount of blood in the choroid which nourishes the back of the eye and is located behind the retina.

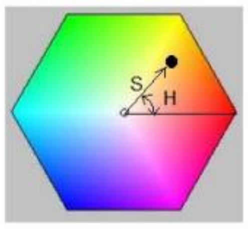


3. Chosen approach

The project will be implemented in C++ using the OpenCV library. The desired effect can be achieved through 3 simple steps.

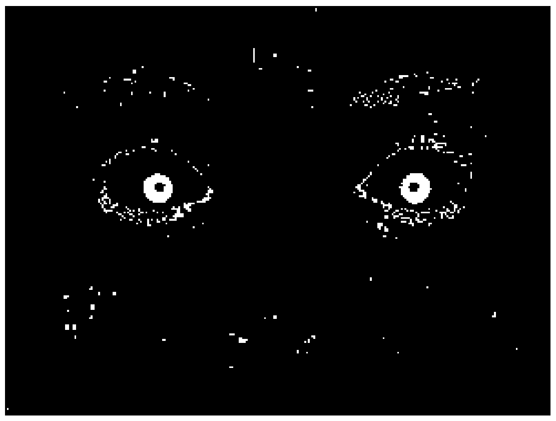
I. The first step is to detect and mask the color red of the original image. This will be done by first converting the color space of our image from RGB to HSV. Since the Hue value encodes color information, it will be the only value we need to achieve the effect. Thus, upon calculating the hue value, we will compare it to the values the red color takes. These values will be approximated by consulting the following image.





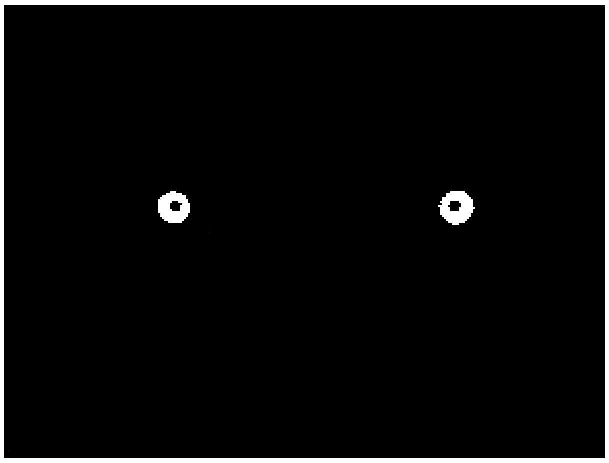
We can see that the pure red color has the hue value 0. We will take into account the variations of red as well and set the range to (340, 360) along with (0, 5).

After doing this, the resulting image will look something like this.



II. We can see that the pupils were masked correctly but some parts of the image were masked as well. The second step is to isolate the pupils. We can do this by detecting the round objects and the objects with a big enough area, keeping them, and tossing the rest.

To detect if an object is round, we must compute the perimeter and the area of the object and compute its thinness ratio. If the ratio is close to 1 then the object tends to be round. If this is done correctly, the result will look like this:



III. The third and last step is take all the pixels that have their corresponding pixel in the masked image as white, compute the average of its blue and green channel and assign the average to that pixel
This should achieve the desired effect and would result in an image that looks like this: