University of Gondar

Institute of technology

Department of Computer Engineering

Digital signal processing project 1

Title: Red eye removal from reddish colored eye image

Group members id no

1.Dawit Degu……...01926/14

2.Kidus Markos……01316/14

Submitted to: Mr. Belayneh

Table of contents

Contents page no

Acknowledgment

We would like to express our sincere gratitude to everyone who contributed to the successful completion of this report. First and foremost, we would like to thank Mr. Belayneh for his valuable guidance and support throughout the entire project. His expertise and insightful suggestions were instrumental in shaping our work, and his supervision played a pivotal role in ensuring the quality of our project. We are deeply appreciative of his patience and encouragement.

We also wish to extend our heartfelt thanks to our senior colleagues for their dedication and selfless assistance. Their guidance, valuable feedback, and willingness to share their knowledge helped us navigate the challenges we faced during the design, debugging, and implementation phases of the project. Their support made a significant difference in our understanding and progress.

Finally, we would like to acknowledge the collective efforts and motivation of all those who inspired us throughout this journey. The encouragement and belief in our ability to succeed fueled our determination and kept us focused on achieving our goal. We are truly grateful for their constant support, which made this accomplishment possible.

Abstract

This paper presents an enhanced method for effectively removing the red-eye effect from digital images, specifically targeting the reddish parts of the eye caused by flash photography. The approach utilizes MATLAB's image processing toolbox to accurately detect and correct the unwanted red coloration, providing a more natural and visually appealing result. Initially, the input image is read and converted to a double-precision format for more precise computation. The RGB color channels are then separated, with a particular focus on the red channel, as red eye typically involves an elevated intensity in this region.

A set of criteria is applied to create a mask identifying the red-eye regions. These criteria include ensuring that the red intensity is significantly greater than the green and blue intensities, with additional conditions to avoid false positives caused by background redness or other image features. The threshold for red intensity is adjusted to suit various image conditions, and morphological operations such as area opening and closing are used to refine the mask, eliminating small artifacts and filling in gaps. This refined mask ensures that only eye regions with excessive redness are targeted.

Once the red-eye regions are identified, the red values are replaced with a more natural eye color, created by blending the green and blue channels to achieve a realistic tone. This blend effectively mitigates the intense red hue, restoring the eye's natural appearance. The corrected image is then reconstructed, and the results are displayed alongside the original image for comparison. The proposed method is highly efficient and can be automated for batch processing of multiple images, making it an ideal solution for red-eye correction in portrait photography. The approach significantly enhances image quality and contributes to more realistic and visually pleasing photos, especially in cases where red-eye artifacts are prevalent due to flash lighting.

This method is demonstrated to be robust across various lighting conditions, offering an effective tool for automated red-eye correction with minimal manual intervention.

Chapter 1

Introduction

This paper presents an enhanced method for removing red-eye effects from images, specifically targeting the reddish parts of the eye in digital photographs. The approach leverages MATLAB's image processing capabilities to detect and correct the red coloration often caused by flash photography. The process begins by reading and converting the input image into a more precise numerical format. The red, green, and blue color channels are extracted and analyzed to identify areas where the red intensity is significantly higher than the green and blue channels. A threshold is applied to ensure that only the reddish regions, typically corresponding to the eye's red-eye effect, are targeted. Morphological operations such as noise removal and hole-closing refine the mask, ensuring accuracy in detecting red-eye regions.

The red areas identified by the mask are then replaced with a more natural eye color, achieved by blending the green and blue channels. This blending creates a more realistic eye appearance, removing the unnatural red tint. The final output is an image with the red-eye effect corrected, providing a more aesthetically pleasing and natural result. The method is effective for automated red-eye correction, improving the visual quality of portraits affected by red-eye.

Chapter 2

Methodology

This documentation provides a comprehensive explanation of the methodology, mathematical equations, and step-by-step details of the MATLAB code for red-eye removal. The goal is to detect and correct red-eye artifacts in images caused by flash photography.

1. Problem Statement

Red eye occurs when the camera flash reflects off the retina, causing the eyes to appear red. The objective is to:

1. Detect red-eye regions in an image.
2. Replace the red pixels with a natural eye color.

2. Methodology Overview

The methodology consists of the following steps:

1. **Image Preprocessing**: Convert the image to a suitable format for computation.
2. **Red-Eye Detection**: Create a binary mask to identify red-eye regions based on color thresholds.
3. **Mask Refinement**: Use morphological operations to improve the mask.
4. **Color Correction**: Replace the red regions with a blend of green and blue channels.
5. **Image Reconstruction**: Combine the corrected channels to produce the final image.

3. Detailed Mathematical Equations and Methodology

3.1 Image Preprocessing

The input image is read and converted to double precision for accurate computations:

```
I = imread('input_image.jpg');
I = im2double(I);
```

- **Purpose**: Convert pixel values to the range [0, 1] for precise arithmetic operations.
- **Mathematical Representation**:

$$I_{\text{double}}(x, y, c) = \frac{I_{\text{uint8}}(x, y, c)}{255}, \quad c \in \{1, 2, 3\}$$

where:

- I_{double} is the double-precision image.
- I_{uint8} is the original 8-bit image.
- (x, y) are pixel coordinates.
- c represents the RGB channels (1 = Red, 2 = Green, 3 = Blue).

3.2 Red-Eye Detection

The red-eyed regions are detected by analyzing the RGB channels. A binary mask is created based on the following conditions:

1. **Red Intensity Threshold**: Red channel values must exceed a predefined threshold.
2. **Red Dominance**: Red must be significantly higher than green and blue.
3. **Intensity Contrast**: The difference between red and the maximum of green/blue must exceed a threshold.

```
red_threshold = 0.55; % Adjust based on image
mask = (red > red_threshold) & (red > 1.4 * green) & (red > 1.4
* blue) & (red - max(green, blue) > 0.15);
```

- **Mathematical Explanation:**
  - **Red Intensity Threshold:**

$$\text{red}(x, y) > \text{red\_threshold}$$

  - **Red Dominance:**

$$\text{red}(x, y) > 1.4 \times \text{green}(x, y)$$

$$\text{red}(x, y) > 1.4 \times \text{blue}(x, y)$$

  - **Intensity Contrast:**

$$\text{red}(x, y) - \max(\text{green}(x, y), \text{blue}(x, y)) > 0.15$$

  - The mask is a binary image where:

$$\text{mask}(x, y) = \begin{cases} 1 & \text{if all conditions are satisfied,} \\ 0 & \text{otherwise.} \end{cases}$$

3.3 Mask Refinement

The initial mask may contain noise or small holes. Morphological operations are applied to refine the mask:

```
mask = bwareaopen(mask, 30); % Remove small regions
mask = imclose(mask, strel('disk', 3)); % Close small holes
```

- **Purpose**:
  - **Remove Small Regions**: `bwareaopen` removes connected components with fewer than 30 pixels.

$$\text{mask}(x,y) = \begin{cases} 0 & \text{if area of connected component} < 30, \\ \text{mask}(x,y) & \text{otherwise.} \end{cases}$$

  - **Close Small Holes**: `imclose` uses a disk-shaped structuring element of radius 3 to close small holes.

$$\text{mask} = \text{mask} \oplus \text{strel('disk', 3)}$$

where $\oplus$ denotes the morphological closing operation.

3.4 Color Correction

The red regions in the mask are replaced with a blend of green and blue channels to create a natural eye color:

```
corrected_red = 0.5 * green + 0.5 * blue; % Blend green & blue for realistic eye color
red(mask) = corrected_red(mask);
```

- **Mathematical Explanation**:
  - The corrected red channel is computed as a weighted average of green and blue channels:

$$\text{corrected\_red}(x,y) = 0.5 \times \text{green}(x,y) + 0.5 \times \text{blue}(x,y)$$

  - The red channel is updated only for pixels where the mask is 1:

$$\text{red}(x,y) = \begin{cases} \text{corrected\_red}(x,y) & \text{if mask}(x,y) = 1, \\ \text{red}(x,y) & \text{otherwise.} \end{cases}$$

3.5 Image Reconstruction

The corrected RGB channels are combined to produce the final image:

Purpose: Concatenates the red, green, and blue channels along the third dimension to form the final image.

- **Mathematical Representation:**

$$J(x, y, 1) = \text{red}(x, y), \quad J(x, y, 2) = \text{green}(x, y), \quad J(x, y, 3) = \text{blue}(x, y)$$

4. Results and Visualization

The original and corrected images are displayed side by side for comparison:

```
figure;
subplot(1,2,1); imshow(I); title('Original Image');
subplot(1,2,2); imshow(J); title('Red-Eye Corrected Image');
```

The corrected image is saved as output_image.jpg:

```
imwrite(J,'output_image.jpg');
```

5. Key Parameters

1. **Red Threshold (red_threshold)**: Controls the minimum red intensity for detection.
2. **Red Dominance Factor (1.4)**: Determines how much red must dominate green and blue.
3. **Intensity Contrast Threshold (0.15)**: Ensures sufficient contrast between red and other channels.
4. **Morphological Parameters**:
    - bwareaopen: Minimum area of connected components to retain.
    - strel('disk', 3): Size of the structuring element for closing holes.

Chapter 3

System design

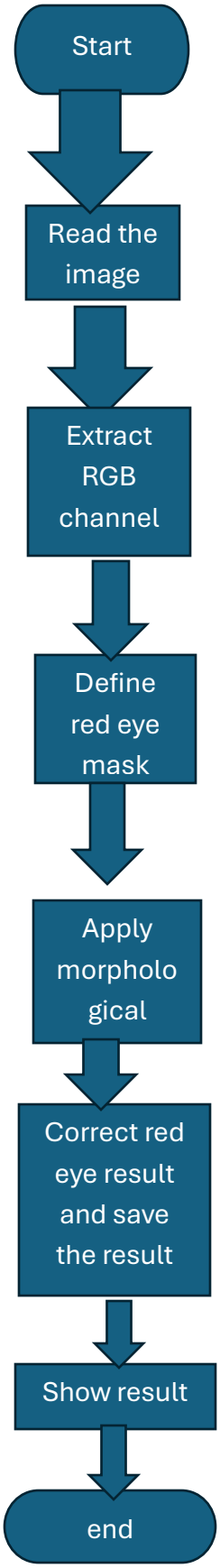
Overview of system architecture and modules

- The system design consists of multiple interconnected modules that work together to efficiently detect and remove the red-eye effect while maintaining natural eye colors. The primary components of the system are as follows:

1. **Input Module:**
    - This module is responsible for loading the input image. The system supports various image formats such as JPEG, PNG, and BMP.
    - The input image is read using MATLAB's imread function and converted into a format suitable for processing.
2. **Preprocessing Module:**
    - Converts the image to the YCbCr or HSV color space, where the red-eye effect is easier to isolate.
    - Performs image smoothing using median or Gaussian filters to reduce noise and improve segmentation accuracy.
3. **Red-Eye Detection Module:**
    - Uses color segmentation techniques to identify red-dominant regions in the image.
    - Applies thresholding to detect areas with a high red component and low blue/green values.
    - Uses MATLAB's regionprops function to analyze connected components and filter out false positives based on shape, size, and intensity characteristics.
    - Morphological operations such as dilation and erosion are applied to refine the detected red-eye regions.

4. **Correction Module:**
    - The identified red pixels are adjusted by reducing their red intensity while maintaining their original texture and brightness.
    - The color of the affected area is replaced with a natural eye color extracted from surrounding regions.
    - Adaptive color correction techniques ensure that the corrected region blends seamlessly with the rest of the eye.
    - If the natural eye color is unknown, an estimation algorithm uses the nearest unaffected pixels to determine the correct hue and saturation.
5. **Post-Processing Module:**
    - Performs final adjustments to enhance the output quality.
    - Uses blending techniques such as alpha blending to merge the corrected area with the surrounding regions smoothly.
    - Converts the image back to RGB format if necessary.
    - Ensures that the final output maintains a natural appearance and does not introduce artifacts.
6. **Output Module:**
    - Displays the corrected image to the user.
    - Saves the output image in the desired format with an option to compare it with the original image.
    - Provides an interface for further manual adjustments if required.

Flow chart


Start
Read the image
Extract RGB channel
Define red eye mask
Apply morpholo gical
Correct red eye result and save the result
Show result
end

Chapter 4

Result and discussion

Input Image:

The input image typically contains a person with a noticeable red eye, caused by the flash reflecting off the retina. In this image, the red component in the eyes will be significantly higher than the green and blue components, creating a reddish hue in the iris.

Red-Eye Mask:

The red-eye mask effectively identifies regions of the image where the red intensity is much higher than the green and blue components. This mask marks the areas where the red-eye effect is likely present, but it also avoids false positives (non-eye regions) by using thresholds and intensity contrasts.

Morphological Operations:

The morphological operations applied to the mask help refine its accuracy. Removing small regions (through area opening) and filling holes (through morphological closing) ensure that only the actual red-eye regions are corrected, improving the overall performance of the algorithm.

Red-Eye Correction:

By blending the green and blue channels to replace the red channel in the detected red-eye areas, the correction produces a more natural and visually appealing result.

Final Image:

The final red-eye corrected image (`output_image.jpg`) should look much more natural, with the red-eye effect removed while preserving the overall appearance of the face. The result should be a clear improvement over the original image, with the red eyes now appearing in a more natural color.

figure that shows the original and corrected image

Chapter 5

Conclusion

- This method for red-eye removal in MATLAB is an effective and efficient approach for dealing with one of the most common issues in portrait photography.
- The use of a well-defined red-eye mask based on the red channel's intensity, combined with morphological operations and color correction, allows for a robust solution that works in a wide range of scenarios.
- The final result is a more natural-looking image with the red-eye effect removed, improving the overall quality and realism of the photograph.
- By adjusting the `red threshold` and other parameters, the algorithm can be fine-tuned to work with different lighting conditions, camera settings, and types of images.

Chapter 6

References

https://www.youtube.com/watch?v=Ao_Lzwr0jZ0&t=164s

https://www.mathworks.com/matlabcentral/fileexchange/25157-image-segmentation-tutorial

https://www.mathworks.com/help/images/morphological-filtering.html

https://towardsdatascience.com/image-processing-with-python-color-correction-using-thresholding-61b8d87c66dc