IMAGE EFFECT APPLICATION

INTRODUCTION:

This software is designed to implement a diverse set of image effects like flip, contrast, rotation, sharpen etc. The effects are thoroughly applied individually using C++. The integration is facilitated through Java Native Interface (JNI), wherein a Java wrapper manages the interaction between the Java and C++ components. It also handles exceptions. Furthermore, an efficient logging mechanism serves as a database for each effect.

EFFECTS:

1. Sharpening:

- The sharpening process involves convolving the image with the sharpening kernel, adjusting pixel values based on the convolution result, and ensuring the final pixel values stay within the valid range (0 to 255).
- The code handles cases where the input image is empty to prevent processing invalid data.
- The resulting sharpened image is stored back into the original image vector.

2. Brightness:

- Brightness of the image can be changed by adjusting the brightness of the image pixels. It takes a 2D vector representing the image and a floating-point parameter for the amount of brightness adjustment.

3. Contrast:

- Contrast effect scales the RGB values of each pixel by a specific amount to enhance or reduce the contrast and maintains the pixel values within the valid range of 0 to 255.

4. Flip:

- Horizontal Flip: If the horizontal flag is set to 1, elements in each row are reversed.
- Vertical Flip: If the vertical flag is set to 1, the i^{th} row and the $n\text{-}i^{th}$ are swapped because we want to reverse the columns.

5. Gaussian Blur:

- This effect convolves the image with a Gaussian kernel of dynamically determined size based on the provided parameter value.
- The kernel is computed and applied to each pixel in the image which results in a blurred image maintaining the valid range (0 to 255).

6. Grayscale:

- This effect converts a color image represented by a 2D vector into grayscale.
- The conversion is done by iterating through each pixel in the image and applying the standard luminosity formula for grayscale conversion.
- The resulting value of grayscale intensity is assigned to RGB components of each pixel.

7. Hue Saturation:

- This effect modifies the hue and saturation of an image.
- It takes the original image vector, along with specified values for saturation (saturation Value) and hue (hue Value).
- The image's hue is adjusted by rotating the color space using a rotation matrix, and saturation is modified based on a grayscale representation of each pixel.
- Rotation matrix for adjusting hue is calculated and this matrix is applied to each pixel’s RGB values.

8. Invert:

- It flips the pixel values of each color channel in the image and creates a negative or inverted color representation which results in an inverted image.

9. Rotation:

- This effect rotates a 2D vector(image) by a specified angle (90, 180, 270 degrees).
- 90-degree rotation (amount == 1), it creates a new image vector and populates it with the rotated pixel values.
- 180-degree rotation (amount == 2), it swaps rows vertically, effectively flipping the image upside down.
- 270-degree rotation (amount == 3), it transposes the image (swaps rows with columns)

10. Sepia:

- The Sepia filter puts a warm, brownish filter on a photo and makes it look like an old picture. The filter works by changing theRGB components of each pixel using certain weightings.

→ temp.r = image[i][j].r * 0.393 + image[i][j].b * 0.189 + image[i][j].g * 0.769;

→ temp.g = image[i][j].r * 0.349 + image[i][j].b * 0.168 + image[i][j].g * 0.686;

→ temp.b = image[i][j].r * 0.272 + image[i][j].b * 0.131 + image[i][j].g * 0.534;

- The sepia tone effect is applied by calculating new values for the RGB components using specific coefficients for each channel and these coefficients determine the contribution of each original channel to the final sepia tone value.

INSTRUCTIONS TO RUN THE PROJECT:

1. Open the Terminal in the Image Frontend Folder and then run the command “npm start”. Then click on the link generated to start the front end.

2. Now open the terminal in the Image Backend Folder and run the following commands.

    1. “make clean”
    2. “./mvnw clean”
    3. “make”
    4. “./mvnw package”
    5. java -jar target/imageEffectApplication-0.0.1-SNAPSHOT.jar

3. Now both the frontend and backend have started. Now we can use the Image Editor.

CONTRIBUTONS:

1. Madhav Girdhar (IMT2022009) - Brightness, Grayscale and implemented the base effect interface for every effect that is used for calling the cpp functions of the effect.
2. Siddhesh Deshpande (IMT2022080) - Gaussian Blur, Rotation and implemented the base effect interface for every effect that is used for calling the cpp functions of the effect.
3. Dikshant (IMT2022549) - Hue Saturation, Contrast, Sepia and Comments.
4. Bijeet Basak (IMT2022510) - Flip, Invert, Sharpen and Comments.
5. Teerth (IMT2022586) and Krish Patel (IMT2022097) - Logging Service, Documentation containing the explanation of the effects.