Yuyang Tian CS5330-Proj1

Project description

Task 1 - display image

The main function, a `Mat` is initialized to store a image, use `imshow` function to show a picture in a window. `waitkey` function waits users for hitting a key in the keyboard, if the key is 's', the image is saved to the host, or any other keys, the window will be closed and program exits.

Task2 - display live video

To display a video, the `VideoCapture` class is used to open the camera and capture frames.

A video consists of a sequence of images, where each image is read into a `Mat` object. If the image is not empty, it is displayed in a window. This process runs in a loop, creating a live video stream.

The `waitKey` function listens for user input. If the key pressed is 's', the current frame is saved. If 'q' is pressed, the program exits.

Task3 - grayscale

If the last user input is 'g', the video switches to a grayscale version. The `isGrey` variable determines the frame's color mode. When the `isGrey` is true, the `frame` Mat is converted to grayscale and displayed in the window.

However, since the mode depends on the most recent selection, pressing any other key will set `isGrey` to `false`, reverting the video to its original color mode, which is BGR.

Transformation within RGB space like adding and removing the alpha channel, conversion to grayscale using

$$RGB(A)\ to\ Gray: Y = 0.299 * R + 0.587 * G + 0.114 * B$$

Task4 - alternative greyscale

I created custom weights for the RGB channels as follows:

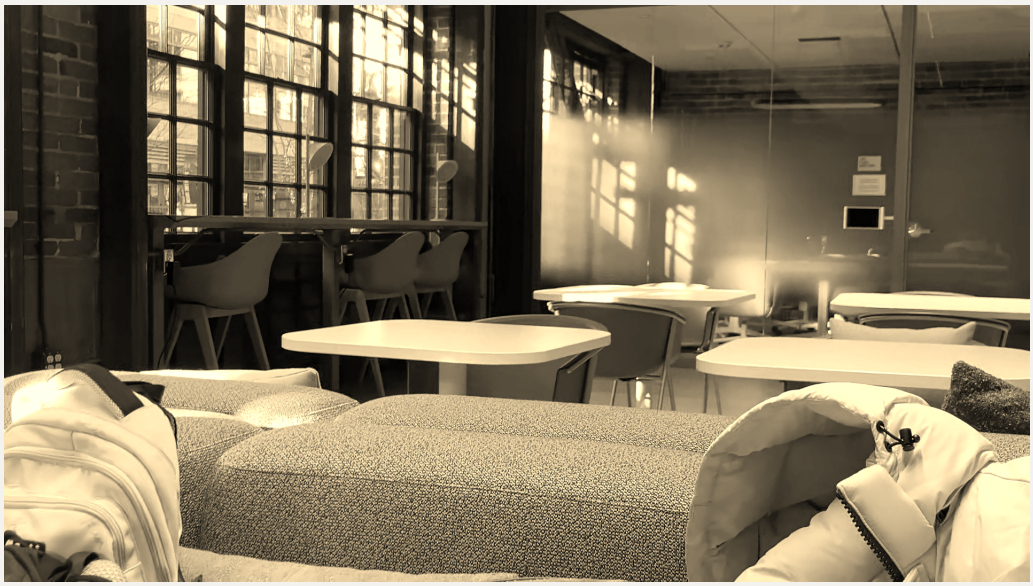
$$RGB(A)\ to\ Gray: Y = 0.005 * R + 0.8 * G + 0.195 * B$$

This alternative grayscale image appears darker in shaded areas (like chairs) and brighter in well-lit regions(windows), providing a different visual effect.

- Alternative:

Task5 - sepiatone filter with extension

I applied matrix transformation manually for each pixel to ensure transforming the original RGB values. After calculating the new values for each channel, we ensure they are within the range `[0,255]` by using `std::min()` to clip the values to 255 if they exceeds that range.



For the vignetting extension, since we want the image become darker towards edges, for each pixels, the distance to the center of the image is calculated using the Euclidean distance. And the vignetting effect is achieved by scaling the color values of each pixel. The further a pixel is from the center, the smaller the scaling factor.
$vignetteFactor = 1 - \frac{distance}{maxDistance}$

Task6 - Blur filter

- Version1:
    - Time that blurred 20 times using version1: 1.2279 seconds
- Version2:
    - Time that blurred 20 times using version1: 0.4816 seconds
    - Before:

- After:

Ane the reason why the version2 is faster:

1. Separable Filtering:
    - Split the 5x5 filter into two 1x5 operations
    - Reduces operations from 25 to 10 per pixel
2. Uses `ptr()` instead of `at()` for direct memory access

Reflection

Acknowledgement