VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

REPORT
Individual Lab 1

Computer Vision

Performed by:
Student's ID: 21127730
Student's name: Hoang Le Cat Thanh

Instructors:
Pham Minh Hoang

Ho Chi Minh City, March 2, 2024

Contents

1 Self evaluation

	Requirements	Completion rate
1	Convert a color image to the gray image	100%
2	Change the brightness of an image	100%
3	Change the contrast of an image	100%
4	Filter an image using average filter	100%
5	Filter an image using median filter	80%
6	Filter an image using Gaussian filter	100%
7	Detect edge of an image using Sobel of kernel size 3×3	100%
8	Detect edge of an image using Laplace of kernel size 3×3	100%

2 Algorithm for each function

2.1 Convert a color image to the gray image

The algorithm is carried out by calculating the grayscale value of each pixel from the RGB value. The value is calculated by applying the NTSC formula representing represents the average person's relative perception of the brightness of red, green, and blue light. The formula is shown below:

$$\mathbf{grayscale = 0.3 * R + 0.59 * G + 0.11 * B}$$

- The algorithm first checks if the input image is not empty, then it checks the number of channels of the image. If the input image only has 1 channel means that the image is already a grayscale image so there's nothing need to be done.
- Otherwise, each pixel will be iterated using a pointer *pixelPtr* to the pixel values in the input image and a pointer *dstPixelPtr* to access the pixel value in the destination image.
- The value after calculated will be assign to the output image through the pointer.

2.2 Change the brightness of an image

In order to change the image's brightness, the function performs an addition to the given brightness factor. If the value is positive, the image brightness will be enhanced and vice versa, the image will be darker if the factor value is negative.

- The algorithm first checks if the input image is not empty and then set the output image the same type as the input.
- Using two pointers to access and iterate over each of the two images pixel value accordingly.
- For each channel, the brightness is adjusted by adding the value of brightness factor. The result is then clipped to be in the range [0, 255] to ensure valid pixel values.

2.3 Change the contrast of an image

In order to change the image's contrast, the function performs an multiplication to the given contrast factor. If the value is greater than 1, the image contrast will be enhanced and the image will be darker in case the factor value is less than 1.

- The algorithm first checks if the input image is not empty and then set the output image the same type as the input.
- Using two pointers to access and iterate over each of the two images pixel value accordingly.
- For each channel, the brightness is adjusted by multiplying the value of contrast factor. The result is then clipped to be in the range [0, 255] to ensure valid pixel values.

2.4 Filter an image using average filter

Filtering an image using the average blurring kernel means that the value of the pixel will be assigned as the average value of the total pixels' value within the kernel

- Before performing the algorithm, the kernel size has to be checked to make sure that the value is valid.
- The pixels are then iterated excluding the border pixels to ensure that the kernel fits entirely within the image and calculate the sum of pixel values for each channel. Then the average value for each channel is calculated separately and assign to the according one in the destination image.

2.5 Filter an image using median filter

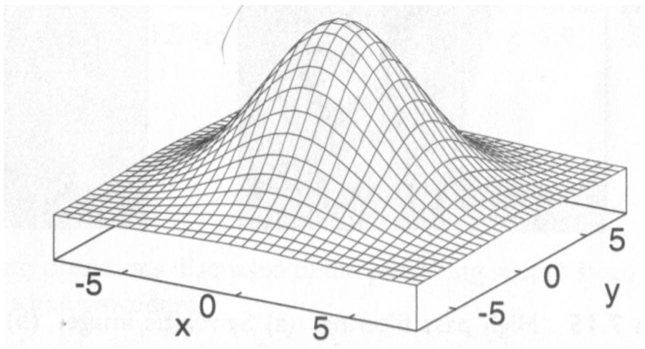
Median filter is applied by storing all the pixel value within the kernel then sort that list and take out the middle value as the new value.

- The input image is transformed into grayscale as the function is currently unable to perform on images with 3 channels.
- Before performing the algorithm, the kernel size has to be checked to make sure that the value is valid.
- The pixels are then iterated excluding the border pixels to ensure that the kernel fits entirely within the image and take out all the value within the kernel to store in a vector.
- The vector is then sorted and the value in the middle is the value to be assigned.

2.6 Filter an image using Gaussian filter

Unlike the previous two filter, Gaussian filter has to be calculated before applying. Each value in the kernel is calculated using the formula with the sigma value is chosen based on the kernel size, the larger kernel size, the broader distribution.

$$h(i,j) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{i^2+j^2}{2\sigma^2}}$$



- The filter kernel is calculated based on the formula and then normalize to ensure that the sum of all elements in the kernel is equal to 1.
- The function then using a pointer to iterate over the image and takes out the pixel within the kernel. These pixels will apply convolution with the kernel to get the desired value and assigned to the output image using another pointer.

2.7 Image edge detection using Sobel 3x3 kernel

Sobel edge detection uses two 3x3 kernels to calculate horizontal and vertical derivatives and the edge image is calculated from the magnitude value of the two derivatives. The kernels are also normalized in order to perform convolution. The two kernel are represented below.

$$W_x = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad W_y = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Before performing the algorithm, the image has to be in grayscale in order for the operator to work on.
- Using a pointer to iterate over the image and takes out the pixels within the kernel. These pixels will convolve sequentially with the horizontal and vertical kernels to get the derivatives and calculate the total sum accordingly.
- The value assigned to the output image is the magnitude value of the two sum.

2.8 Image edge detection using Laplace 3x3 kernel

Laplacian filter computes the second derivatives of an image, measuring the rate at which the first derivatives change. The filter used for convolution is defined as

$$Laplace = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Before performing the algorithm, the image has to be in grayscale in order for the operator to work on.

- Using a pointer to iterate over the image and takes out the pixels within the kernel. These pixels will convolve with the kernel to get the second derivative values.
- The total value in each kernel will be the output value for the pixel. The absolute value of the result is taken to emphasize edges.

3 Result and comaprison

The average runtime (milisecond) for both methods

Functions	Self-implemented	OpenCV
Convert a color image to the gray image	4	5
Change the brightness of an image	16	11
Change the contrast of an image	23	45
Filter an image using average filter (k = 5)	254	46
Filter an image using median filter (k = 5)	4387	444
Filter an image using Gaussian filter (k = 5)	331	35
Detect edge of an image using Sobel of kernel size 3 × 3	69	308
Detect edge of an image using Laplace of kernel size 3 × 3	72	154

The input image used for comparison is shown below

3.1 Convert a color image to the gray image

My Love Story
with Yamada-kun
at Lv999

(a) Self-implement

My Love Story
with Yamada-kun
at Lv999

(b) OpenCV

When testing with the given input image, a self-implemented algorithm and OpenCV produced nearly identical results. However, the self-implemented algorithm had a slightly shorter average runtime with the same input image.

3.2 Change the brightness of an image

My Love Story
with Yamada-kun
at Lv999

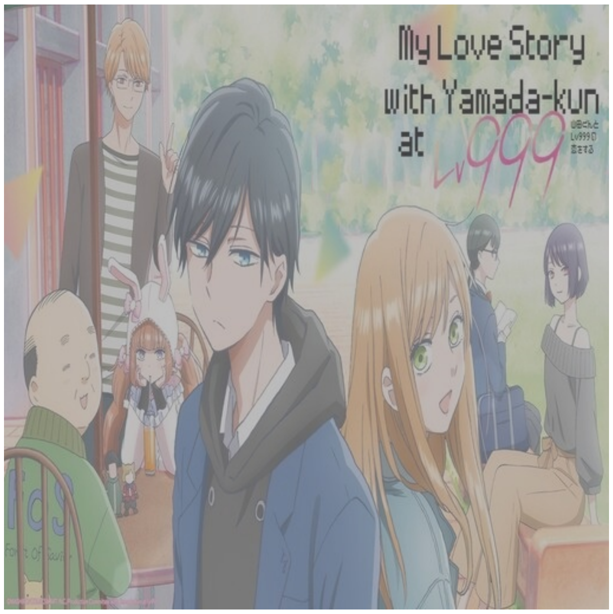
(a) Self-implement

My Love Story
with Yamada-kun
at Lv999

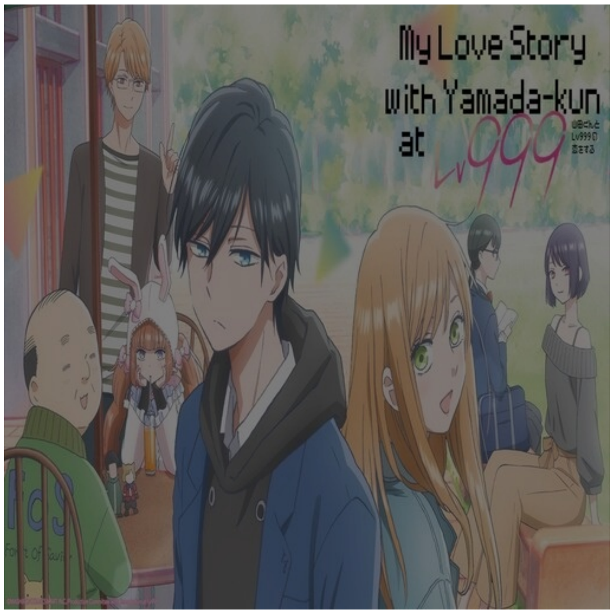
(b) OpenCV

The same result images produced with the function used to adjust the brightness of an image. Though yield the same outcomes, OpenCV significantly reduced the processing time, thus given higher efficiency.

3.3 Change the contrast of an image



(a) Self-implement



(b) OpenCV

With the same contrast factor, OpenCV produced better output image with contrast value change significantly. Due to that, the average runtime for this method is longer than when using self-implemented algorithm.

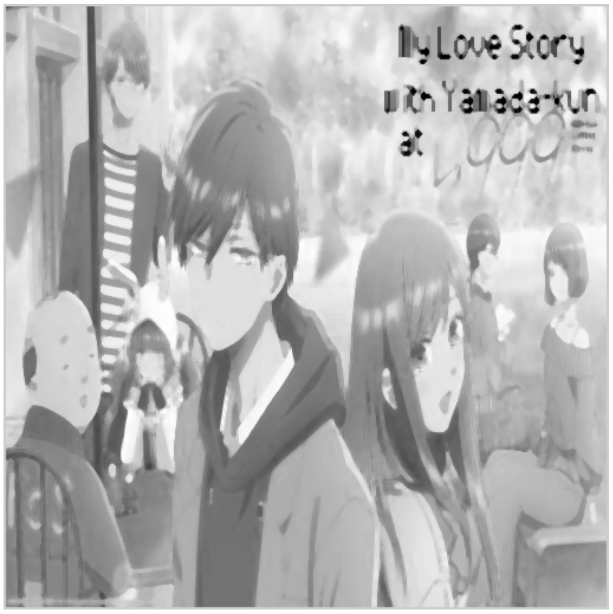
3.4 Filter an image using average filter

My Love Story
with Yamada-kun
at Lv999
(a) Self-implement

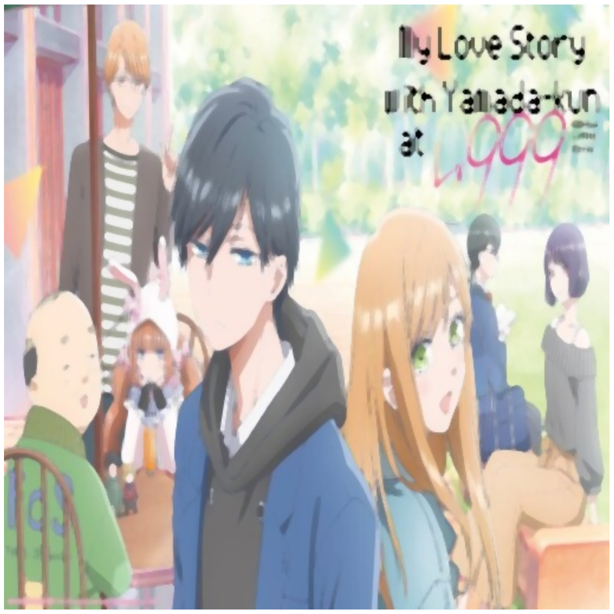
My Love Story
with Yamada-kun
at Lv999
(b) OpenCV

It can be clearly seen that when performing more complicated function, using OpenCV gives better result in both terms of output image and average runtime. Since the self-implemented algorithm uses nested loop to iterate while excluding border pixels, those pixels are ignored while carrying out the function with significantly longer time to process.

3.5 Filter an image using median filter



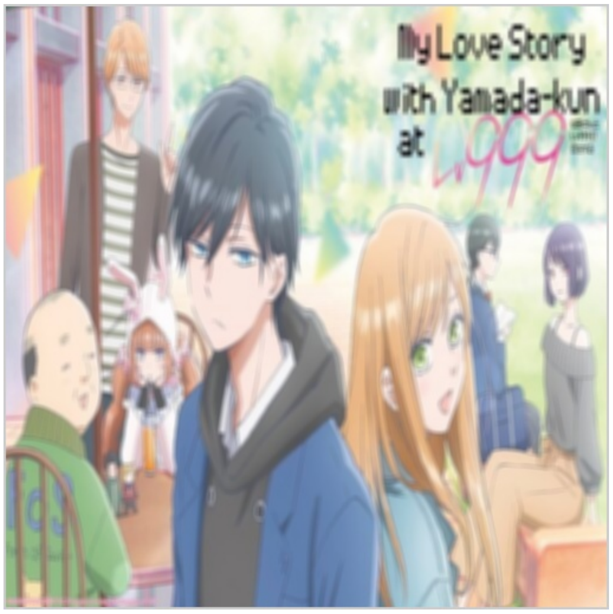
(a) Self-implement



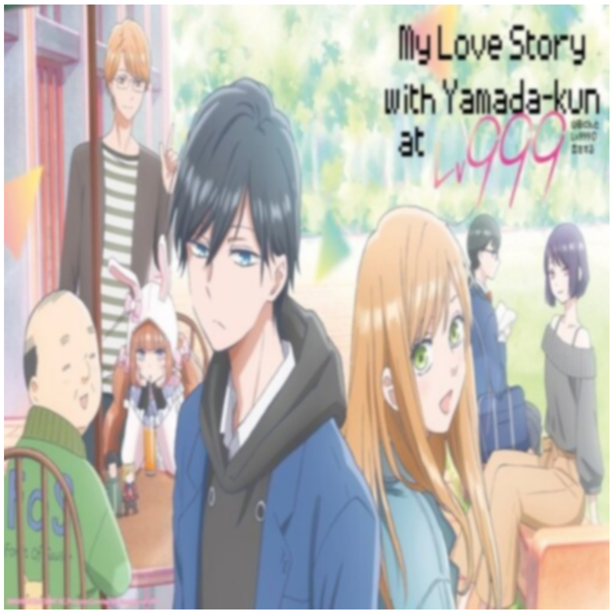
(b) OpenCV

Comparing to the method of using a widely-used library in computer vision, OpenCV, the algorithm performed by self-implemented code falls short behind due to the failure to handle colored image. Therefore, input image needs to be convert into grayscale before applying the algorithm, leading to much longer runtime and lower the efficiency.

3.6 Filter an image using Gaussian filter



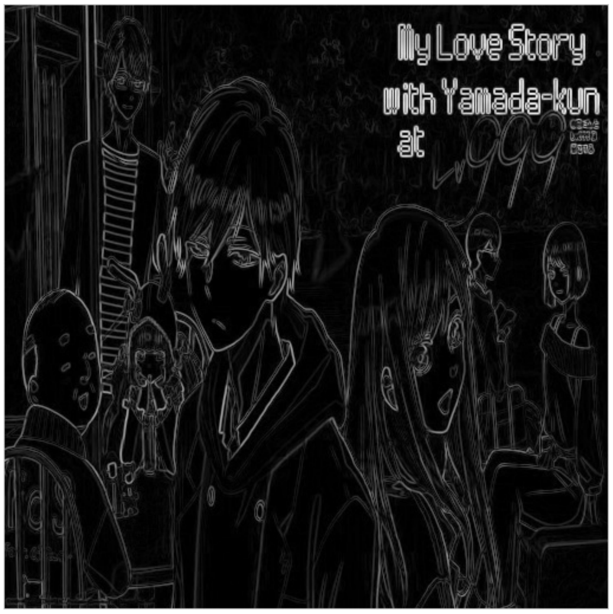
(a) Self-implement



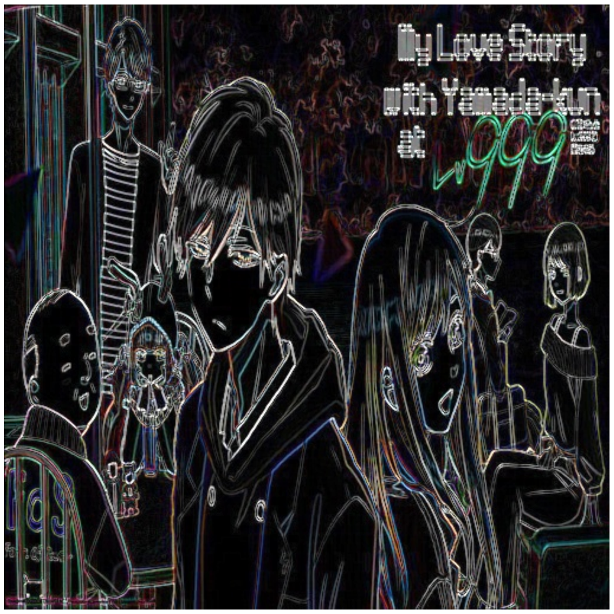
(b) OpenCV

Due to having to calculate the Gaussian kernel from scratch, together with the use of nested loop, the average time required to process the algorithm without the help of OpenCV library is clearly much longer than when applying by the library. Moreover, similar to the previous two filter, the self-implemented algorithm also ignore the border pixels so that the filter kernel can fit in the image.

3.7 Image edge detection using Sobel 3x3 kernel



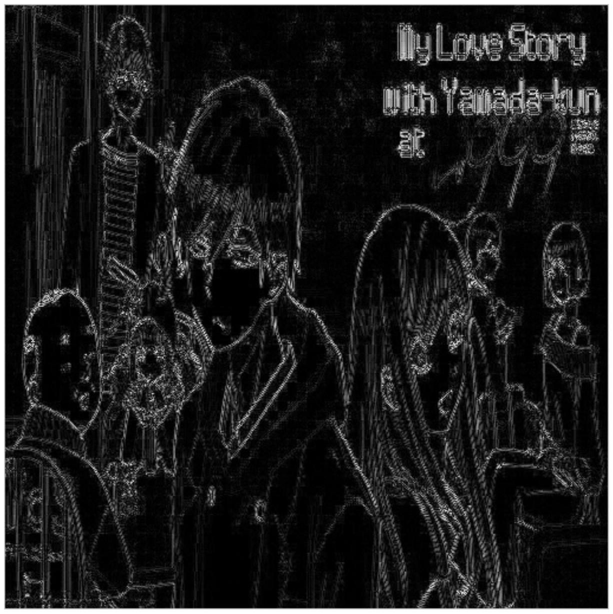
(a) Self-implement



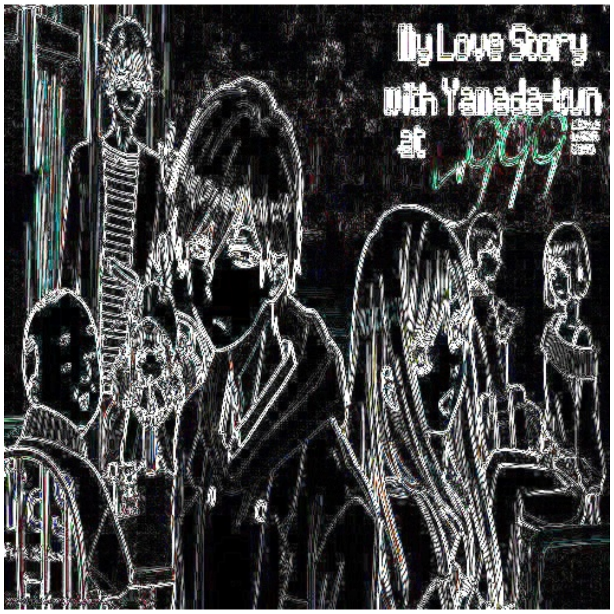
(b) OpenCV

With the complexity for the algorithm is O(n), the output image using the method utilized OpenCV excelled in edge clarity, producing images with sharp and well-defined edges. On the other hand, the self-implemented method, despite its advantage of requiring less average runtime, fell short in this aspect, resulting in images with less clear edges.

3.8 Image edge detection using Laplace 3x3 kernel



(a) Self-implement



(b) OpenCV

Similarly, the same case apply for Laplacian filter. OpenCV once again outperformed the self-implemented method with superior edge detection capabilities, capturing the edges of the input image with greater precision and clarity though ends up taking more time to finish the process.

4 References

[0] Example code
[1] Lecture slide
[2] Digital Image and Video Processing lecture slide
[3] Convert to grayscale
[4] Change image brightness
[5] Change image contrast
[6] Average and Gaussian filter
[7] Median filter
[8] Sobel edge detection
[9] Laplacian filter