Problem Set 1: Basic Image and Video Operations Using Python and OpenCV

Utku Acar
CS 523.V: Computer Vision
M.Sc in Computer Science Department
Ozyegin University
Vestel Electronics
Manisa, Turkey
utku.acar@ozu.edu.tr

April 9, 2023

1 Abstract

In this project, I implemented basic image and video operations using Python and OpenCV library. The project consisted of two problems. The first problem involved performing operations on a color image such as cropping, channel extraction, grayscale conversion, Sobel filtering, gradient magnitude and orientation computation, and Laplacian of Gaussian filtering. The second problem required live video stream processing, gradient magnitude computation, frame concatenation, and video recording for a certain duration.

Keywords: OpenCV, Python, image processing, video processing, Sobel filter, Laplacian of Gaussian filter, gradient magnitude, gradient orientation, fps

2 Introduction

The field of computer vision has been revolutionized by the OpenCV library, which provides a wide range of tools and algorithms for image and video processing. In this project, I utilized OpenCV with Python to perform basic operations on images and videos.

3 Problem 1: Basic Image Operations

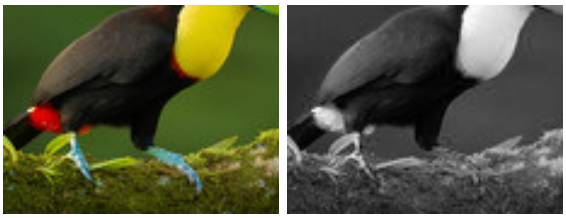
The first problem required us to perform basic operations on a color image. I started by downloading the image to my computer and then proceeded with the following steps:

1. Cropping out a region from the center of the image: I wrote a Python script to crop out a region from the center of the image with half the size of the input image. This means I took pixels from the center to a quarter of the size of the dimensions of the original image both height and width so I get half-sized cropped output. The resulting image was saved as a PNG file. You can see the result in Fig. 1 below.

(a) Original Image (b) Cropped Image

Figure 1: Comparison of Original and Cropped Images

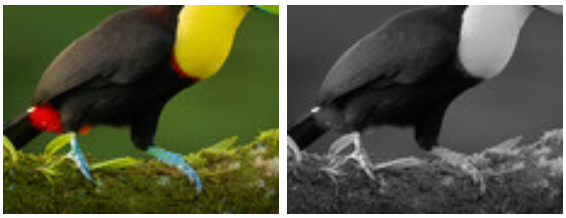
2. Extracting the red channel of the image and displaying it: I extracted the red channel of the image and displayed it using OpenCV.You can see the result in Fig. 2 below. In this figure, the brighter pixels are more likely to be Red in the Original RGB Image and darker pixels are less likely to be Red.



(a) Cropped Image (b) Cropped Red Channel

Figure 2: Comparison of Cropped and Cropped Red Channel of the Images

3. Converting the image to grayscale and displaying it: I converted the image to grayscale using OpenCV and displayed it. You can see the result in Fig. 3 below. The figure shows a pixel pattern that closely resembles the green channel of the original image.



(a) Cropped Image (b) Gray Scaled Image

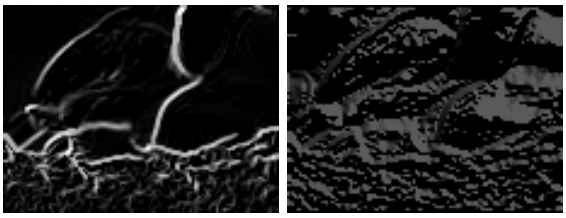
Figure 3: Comparison of Cropped and Cropped Gray Scaled Images

By comparing Fig. 2 with Fig. 3, we can observe that the Parrot's tail has a thick bright line in the red channel, while the grayscale image has a darker thick line at the same location. Additionally, since the upper point of that thick line is a white pixel, this spot appears white in both the red channel and grayscale image outputs.

4. Defining the Sobel filters in the x and y directions and applying them to the grayscale image: I used the Sobel filter to perform edge detection on the grayscale image. The Sobel filter is a discrete differentiation operator that calculates an approximation of the gradient of the image intensity function. I defined the Sobel filters for the x and y directions in my code and applied them to the grayscale image.

   I then obtained the gradient magnitude and orientation using these gradients. The gradient magnitude is the magnitude of the gradient vector, which is calculated as the square root of the sum of the squares of the gradients in the x and y directions. The gradient orientation is the direction of the gradient vector, which is calculated as the arctan of the ratio of the gradient in the y direction to the gradient in the x direction.

   You can see both the Gradient Magnitude and Orientation in Fig. 4 below.



(a) Gradient Magnitude of the Image (b) Gradient Orientation of the Image

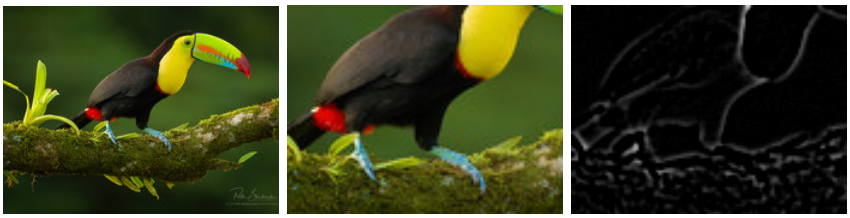
Figure 4: Comparison of Gradient Magnitude and Gradient Orientation of the Image

5. Obtaining Laplacian of Gaussian image for different sigma values and displaying the results: I used the Laplacian of Gaussian (LoG) operator

to perform blob detection on the input image. The LoG operator is a combination of the Laplacian operator and a Gaussian filter. It is used for detecting blobs in an image by convolving the image with the LoG operator.

To obtain the LoG image, I convolved the input image with the LoG operator for different sigma values. Sigma is the standard deviation of the Gaussian filter used in the LoG operator. I chose sigma values ranging from 1 to 4 and displayed the results. As the value of sigma increases, the size of the blobs detected by the LoG operator also increases. However, increasing sigma too much can cause the image to become too blurred, resulting in distorted output.

In Fig. 5 below, you can see the results of applying the LoG operator for different sigma values to the input image. The output images become increasingly distorted as the value of sigma increases, but the blobs detected by the LoG operator also become larger.



(a) Original Image (b) Cropped Image (c) LOG with $\Sigma = 1$

(d) LOG with $\Sigma = 2$ (e) LOG with $\Sigma = 3$ (f) LOG with $\Sigma = 4$

Figure 5: Comparison of Images with Different Sigma Values

4 Problem 2: Basic Video Operations

The second problem required me to perform live video stream processing. I wrote a Python script to take a live video stream from the webcam of my computer, convert it to grayscale, and compute the gradient magnitude using the Sobel filter. To calculate the magnitude response in real-time, I used the clip method to restrict the pixel range to [0,255]. Without this method, the magnitude response values could go out of this range, resulting in incorrect or distorted visualization of the image, and in my case, there was a blank white

screen in the location of the gradient response. Later on, I found the method "clip" and fixed this problem. [1, 2]

I also displayed the input frame and the gradient magnitude side by side on the screen and recorded 15 seconds of the video stream. The video stream stopped when I hit "q" on my keyboard, and I displayed the fps on the video frames. By concatenating the input frame and the gradient magnitude image, I was able to show both images side by side. You can see the result in Fig. 6 below.

Figure 6: Video stream and Gradient Magnitude side by side

You can also find the output video which includes both the video stream and gradient magnitude response of it side by side in the "output" folder.

5 Conclusion

In conclusion, I successfully implemented basic image and video operations using Python and OpenCV. I was able to perform cropping, channel extraction, grayscale conversion, Sobel filtering, gradient magnitude, and orientation computation, Laplacian of Gaussian filtering, and live video stream processing. The techniques and algorithms used in this project can be applied to various computer vision tasks.

References

[1] "OpenCV documentation," https://docs.opencv.org/4.5.4/.

[2] J. Howse and J. Minichino, "Learning OpenCV 4 Computer Vision with Python 3," 3rd ed., Packt Publishing, 2018.