

UKRAINIAN CATHOLIC UNIVERSITY

FACULTY OF APPLIED SCIENCES

BUSINESS ANALYTICS & COMPUTER SCIENCE PROGRAMMES

Convolution and Image Processing

Linear Algebra final project report

Authors:

Anna-Maria PASHUK

Diana BILETSKA

Andrii TURKO

May 2021



APPLIED
SCIENCES
FACULTY ●

Abstract

We will discuss image processing algorithm of linear algebra. Our goal is to learn and understand theory to implement different methods of linear algebra for image transformations, filtering and test them by comparing them with existing python libraries. So, after implementing our features, we analyze the built-in libraries for the same problems to compare the speed of processing. The user will download the image, then he can choose from a list of possible transformations for this image, such as blurring, rotation, scaling, applying filters, such as black and white, cyan, yellow, and so on. We choose 3 basic strategies, such as edge detection, matrix transformations and color transformations. The implementation of the project you can find here [Google Colab](#)

1 Introduction

We had chosen image processing as our research topic for the project, because we were interested in wider understanding and learning of this topic. In the modern world people can not imagine their being without pictures and visualization, due to the fact that information provided with the help of image, processed better, deeper, and easily than just the words. After we made a research on this topic, we have understood that image processing is one of the most popular and fast-growing area nowadays. It is used almost in every part of our everyday life, such as medicine, production, education, or entertainment. Even though we chose our topic due to interest, while we were doing research, we understood the importance of image processing as well.

2 Explanation of the general topic and problem area

The importance of Image processing is presented in two directions. The first is improving visual information to make it more convenient for people's understanding. And the second is processing the image data for machine perception. Image processing is usually needed as manipulation with an image for receiving aesthetic appearance or to provide useful information. However, working with images is a method of translating a photograph between the human visual system and a digital device. Human's eye cannot perceive the world around its same way as digital detectors and devices often show a picture with some effects like denoise or pixalization. There are many noticeable differences between humans and digital receptors and also processing steps to achieve the final vision. For a good result image processing should be made with the scientific methods, so others can use an algorithm with any other picture and confirm their results. Now we want to explain exactly what our topic is about. Image processing is when we make some actions with a picture, to change it somehow or receive some needed information from it. In this process our input information is always a picture, or some features we receive from it and an output information can be anything due to the action we do with an image.

In our case this actions are color transformation, matrix transformation and edge detection. The color processing is important due to the fact that colors is a powerful tool, which simplifies object detection and extraction. With the help of color transformation human can see much more details and analyze the picture deeply. The idea of a color system is to facilitate the specification of colors in some standard, generally accepted by human eye way. Basically, a color model is a coordinate system and a subspace within

that system, where each color is represented by a point. Talking about digital image processing, the color system most commonly used is the RGB model (red, green, blue) for color monitors and video cameras; CMY (cyan, magenta, yellow) and CMYK (cyan, magenta, black) models for color printing, which closely matches how people describe and interpret color.

Matrix transformation allow us to perform rotation, scaling, mirroring and a lot more actions on the image. A matrix transformation is a linear transformation that is determined by a matrix along with bases for the vector spaces. Transform is used for transforming an image into a domain where there is more relevant data and less needless information

Edge detection is extremely important part of image processing. This is the type of filter used to select edge points in an image. Sudden changes in the image occur when the edge of the image along the contour intersects the brightness of the image. It helps to define essential information in the picture and can be used in object recognition. Accurate edge detection is necessary for a number of image analysis and recognition techniques.

3 Our project structure

We have decided to create a prototype for an image processing application. Now all interaction will be inside of the google colab workspace, but later it can be used as an application or website. We have divided all image effects and processes into three parts and in each user can enter in colab which concrete type of effect they want to try in this part.

4 Colour processing

The matrix of image color depends on the color system. We will use an RGB system, where for each pixel we can see the amount of red, green and blue. We use our multiplication matrix to provide color transformation. We chose to implement black and white, yellow, magenta and cyan filters. For this we make a unique transformation matrix based on the theory we studied and receive the different color pictures.

4.1 Black and white



We have chosen a black and white filter as the first colour change. As far, as in our image matrix we have RGB coordinates (Red, Green, Blue), we have to rewrite them into grayscale. For this, we have used the luminosity method, which we liked the best. Colours in grayscale have the same coordinates in RGB system, so we have to find this coordinate. In luminosity method it is the following: $0.21 \cdot \text{Red} + 0.72 \cdot \text{Green} + 0.07 \cdot \text{Blue}$. After that we formed a filter matrix :

$$\begin{pmatrix} 0.21 & 0.21 & 0.21 \\ 0.72 & 0.72 & 0.72 \\ 0.07 & 0.07 & 0.07 \end{pmatrix}$$

For the matrix of our image we multiplied every pixel, which is an array like [R, G, B], with a filter matrix.

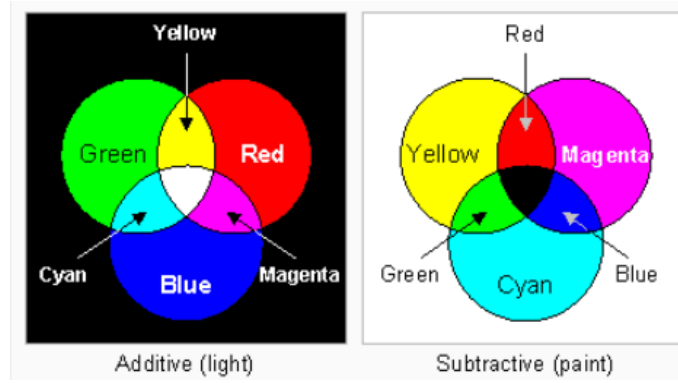
4.2 Matrix multiplication

Matrix multiplication is one of the basic operations in linear algebra. For most methods we want to use in our project we need matrix multiplication function. So we implement it in Python. Also we create an auxiliary function "check rows size" to check whether the sizes of our matrices are appropriate for multiplying. So our main function uses the typical logic for matrix multiplication:


$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}$$


4.3 Colour filters


We have decided to find out more about colour transformations and implement some of them. We have three variants of colour to apply as a filter for the image: "yellow", "magenta" and "cyan". So, the user enters the colour and we show him/her an image with this colour as dominant. First, we form a filter matrix for each colour and then apply the same algorithm as in black and white transformation. To build these filter matrices we have had to understand the theory and logic behind colours. The thing is, that there are two absolutely different ways to mix the colour: Additive and Subtractive. An additive colour diagram shows the result of mixing component lights. Subtractive diagram shows spectral power distribution, so actually, we will get by mixing inks or dyes in painting. In our case, we need to use additive mixing to build filter matrices because we work in RGB system. (not in CMY) So, to get a yellow filter we need to take away all blue shadows from the picture, which means multiplying B meaning in our matrix on 0. The same logic is true for creating magenta and cyan filters.



Our solution can be checked in Paint program, where we can enter RGB coordinates:

| Manual Color Input | | | |
|---|-------|--------------------------------------|-------------------------------------|
|  | Hex | <input type="text" value="#00ffff"/> | <input type="button" value="Copy"/> |
| | Red | <input type="text" value="0"/> | |
| | Green | <input type="text" value="255"/> | |
| | Blue | <input type="text" value="255"/> | |

| Manual Color Input | | | |
|---|-------|--------------------------------------|-------------------------------------|
|  | Hex | <input type="text" value="#ff00ff"/> | <input type="button" value="Copy"/> |
| | Red | <input type="text" value="255"/> | |
| | Green | <input type="text" value="0"/> | |
| | Blue | <input type="text" value="255"/> | |

| Manual Color Input | | | |
|---|-------|--------------------------------------|-------------------------------------|
|  | Hex | <input type="text" value="#ffff00"/> | <input type="button" value="Copy"/> |
| | Red | <input type="text" value="255"/> | |
| | Green | <input type="text" value="255"/> | |
| | Blue | <input type="text" value="0"/> | |

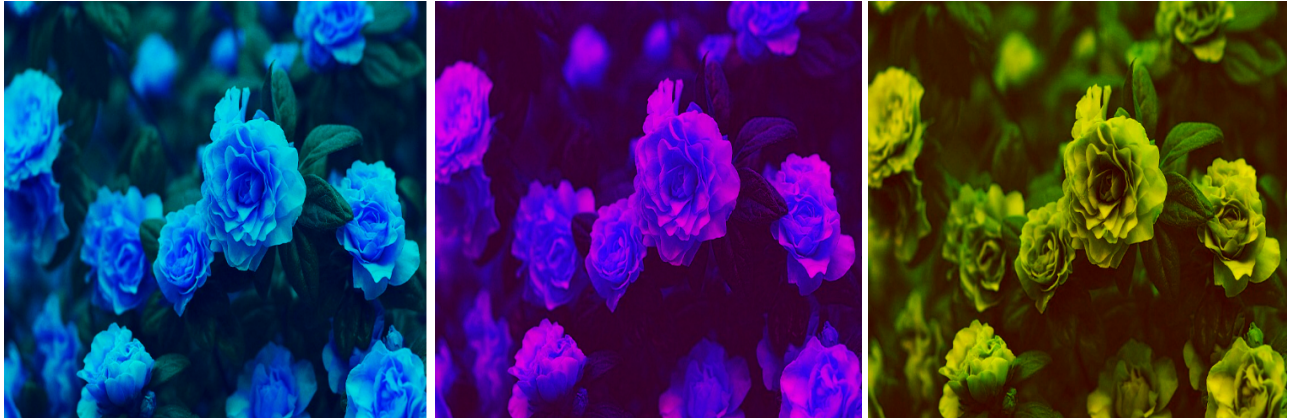
As a result, our filter matrices are:

$$filter_{cyan} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

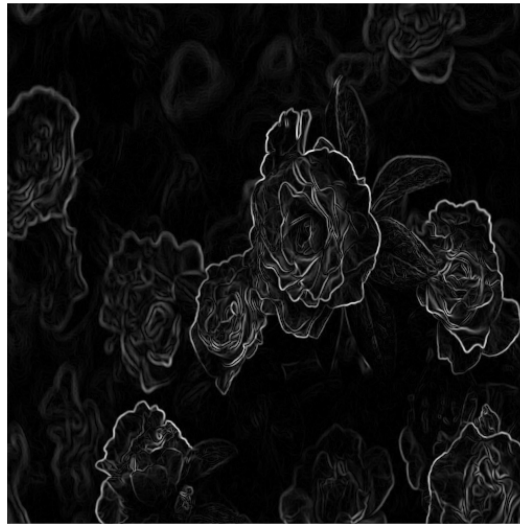
$$filter_{magenta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$filter_{yellow} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The results of our colour transformations:



4.4 Edge detection



The detection algorithm uses three image processing methods: edge detection, line detection and angle detection. Edge detection converts an image into a bitmap image, where each pixel is classified as belonging or not belonging to an edge. After identifying the edges, the image is processed with a change to distinguish lines. Contours are closed edges and shapes of the image. The approach we used for edge detection is convolution based on the Sobel vertical and horizontal matrix. The convolution operation is the process of adding each element of the image to its neighbor, with the operation weighted by the kernel. Python provides some methods to do it, but we decided to implement it ourselves and already have done it. So, all that remains is to test our function, which we will do by comparing time for our method and python function.

Edge detection is an image processing algorithm for finding the contour of objects in the image. It works by finding changes in light. Edge detection can help to make image partition and get data in areas such as image processing, machine vision, and computer vision.

As we are using different ways of changing images, we reached the conclusion that edge detection is also an interesting method to show how linear algebra is used in image processing. Firstly, we put a black and white filter on our image. After that we create two sobel filters, vertical and horizontal, convolve them separately with the matrix of our bw image and then combine the results to get detected edges and to show them on a new image.

4.5 Convolution

This method clearly shows all edges found on the image and works correctly. Also it is easy for a person, who sees the code for the first time, to get a proper idea. But there is risk, that algorithm won't give clear edges for images with bad quality.

As we started implementing this algorithm, we needed to learn what convolution is and to write a function for it in Python.

Sobel Edge detection need two 3x3 convolution kernels matrices. G_x is a horizontal kernel and G_y is vertical

The measurements of image can be displayed in both orientation, like vertical or horizontal, so the Kernels are applied to picture separately. The kernel matrix is also called as mask. When we apply this mask on the image it detects horizontal or vertical edges. It calculates the difference of intensities of pixels by taking the derivative.

$$sobel_{vertical} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

$$sobel_{horizontal} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Due to the fact that the middle column or row consists of zeros the algorithm does not include the real values of image, but check the difference between left and right (for vertical) or up and down (for horizontal) values around the edge.

The center values of the first and third column or row is 2 and -2 because it give more weight to the values of pixels near the edge. This values help increase intensity of the edges and it become more visible than on the original image.

This kernel matrices will find edges in vertical and horizontal direction. When we convolve such masks onto an image it would prominent horizontal and then vertical edges in the image.

5 Image transformation: Blur, Rotation, Mirror, Scaling

Pixels change their position inside the image, or more precisely, when each pixel in the matrix is built on the basis of another pixel of the matrix, but without changing its color. In terms of linear algebra, filters are applied to each pixel of the matrix using a transformation matrix. Such functions can rotate, zoom or blur. As with the edges and lines detection we made our own algorithm for matrix multiplication, which allows us to rotate or zoom images. The remaining work is a blur function and there are a lot of types of it and ways to implement it. The approach which fits us best is to make a Kernel matrix and do averaging. In this part we want to show different types of image transformation. For each of this processes we have to build transformation matrix and multiply it with our image matrix. As far, as we already know transformation matrices for Rotation, Mirroring and Scaling, the main difficulty is that we have to write our own function "multiplication" and "transform". "Transform" function includes multiplication of our image with transformation matrix and adding black background for proper image output. In Blur the main difficulty is to build the correct Blur function that will work properly with our

multiplication and transformation algorithm. The theoretical part behind the algorithm with stress on the LA methods used: Blur can be done as simple averaging (which is also called as mean and Homogenic filter), such that image will be blurred evenly. This is reached by taking the average value of the around pixels and changing each element on the average of its neighbours. For this we take Kernel matrix:

$$K = \frac{1}{m * n} \begin{pmatrix} 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & \dots \end{pmatrix}$$

To make circled, linear or another blur, we have to build a Kernel matrix based on values of function, which is dependent on our x and y coordinate. How our blur works. Firstly, the user chooses the degree of blur, which corresponds to the privacy level. This coefficient detects amount of pixels, which will have one common averaged colour. For example, this coefficient is 9, that means that we take 3x3 square of pixels, count mean of their rgb values, and fill these 9 pixels with received colour.

6 Conclusions

In this project, we reached our goal: implemented interesting image transformations on own without using any python functions for linear algebra approaches. We have visually checked the correctness of our work. Also, we have compared processing time of our functions with python implementations. Although our function works a little slower, it's main advantage is in simplicity and intelligibility.

References

- [1] http://vision.stanford.edu/teaching/cs131_fall1718/files/02_notes.pdf
- [2] <https://forum.patagames.com/posts/t501-What-Is-Transformation-Matrix-and-How-to-Use>
- [3] <https://scholarworks.uark.edu/cgi/viewcontent.cgi?article=1067context=csceuht>
- [4] <https://www.youtube.com/watch?v=am36dePheDc&list=PLkDaE6sCZn6G129AoE31iwdVwSG-KnDzFi>
DeepLearningAI
- [5] <https://www.semanticscholar.org/paper/Sobel-Edge-Detection-Algorithm-Gupta-Mazumdar>