Image Processing Program: A Python Implementation of Concepts Learned in Linear Algebra

By
Asif Tauhid (st4787)
Komal Neupane (kn2200)

To
Prof. Joachim Lebovits

Project Link for Google Colab

Project Link for Github

Introduction

This project is an image manipulation and editing application in Python that leverages linear algebra concepts for different transformations on an image. Linear algebra forms the backbone of various image transformations such as cropping, resizing, rotating, and zooming. In this report, we explore how these concepts are applied and implemented in the code to achieve the desired features. This Project provides a practical demonstration of how mathematical concepts are directly applied in computer science and image processing.

Program Overview

The program is designed using python language, leveraging minimal Python libraries to handle image input/output while focusing on manual implementations of image transformations. This approach highlights the application of linear algebra in manipulating the array-based data structures that represent images.

Digital images are represented as matrices where each pixel contains vector values representing color intensities. Transforming an image involves manipulating its matrix representation.

Key Features and Functions

1. Image Loading

The program begins by loading an image from a specified path in Google Drive. This step uses the imageio library, which simplifies reading the image into a numpy array. Each pixel in an image is represented as a combination of RGB (red, green, blue) values, stored in a matrix form (a two-dimensional array).

2. Cropping

Feature Description: Cropping an image extracts a specific region from the image, determined by starting and ending rows and columns.

Linear Algebra Implementation: This operation involves using array slicing on the image matrix to select the desired subset. This is a direct implementation of linear algebra's concept of subsetting a matrix. The code returns the portion of the image matrix that falls within the specified bounds.

3. Resizing

Feature Description: The resizing function changes the dimensions of the image, effectively altering the resolution. This is implemented by computing the ratio of new dimensions to old dimensions and then selecting pixels at intervals determined by these ratios. The function iterates over the desired output size and samples pixels from positions scaled according to these ratios. This operation can be seen as a practical application of matrix transformation, where the original matrix is transformed into a new matrix of the desired size through a process similar to matrix interpolation.

Linear Algebra Implementation: This transformation is achieved by calculating the height and width ratios based on the original and new dimensions. Each pixel in the resized image is mapped to the corresponding coordinates in the original image using these ratios. Nearest-neighbor interpolation is used to approximate the pixel values in the resized image.

The resizing function employs nested loops, where each iteration calculates the corresponding coordinates in the original image using the height and width ratios. This is a form of scaling in linear algebra.

4. Rotating

Feature Description: Rotating an image involves rotating the entire image around its center by a specified angle. The function rearranges the matrix elements to rotate the image, which is equivalent to performing a linear transformation on the matrix representing the image.

Linear Algebra Implementation: Rotation is performed using a rotation matrix, a fundamental linear algebra concept. The rotation matrix is applied to each pixel's coordinates to determine the new positions of the pixels after rotation. The function first converts the rotation angle from degrees to radians. Then, a 2D rotation matrix is constructed using trigonometric functions. For each pixel in the new (rotated) image, the function calculates the original coordinates using the inverse of the rotation matrix. It uses these original coordinates to fetch the corresponding pixel value from the original image.

Nested loops iterate over each pixel in the rotated image. For each pixel, the function computes the original coordinates and then performs nearest-neighbor interpolation to map the original image pixel to the rotated image. This also makes use of the fundamental concept of transformations in linear algebra using matrices.

5. Zooming

Feature Description: Zooming allows the user to magnify or reduce the image using a zoom factor.

Math Implementation: Zooming is achieved through a scaling matrix that adjusts the image's matrix representation based on the zoom factor. A zoom factor greater than 1 magnifies the image, while a factor less than 1 reduces the image.

Nested loops iterate through each pixel in the zoomed image, computing the original coordinates based on the zoom factor and the scaling matrix. The calculated coordinates are then used to fetch pixel values from the original image, which are set in the zoomed image.

6. Displaying Images

After each transformation, the image is displayed in the notebook. This allows for immediate visualization after each of the matrix operations are performed on the image data.

Conclusion:

Linear algebra provides the mathematical foundation for operations on matrices, which are crucial to image-processing tasks. Each pixel of an image can be thought of as a vector, and the entire image is a matrix of vectors. Overall, this program demonstrates how linear algebra is not just a field of abstract mathematical theory but a practical toolkit for modern computer applications like image processing. By manipulating the matrices that represent digital images, we can perform complex transformations that are both foundational and essential for advanced image processing tasks, computer graphics, and many other applications in technology and science.