Linear Algebra Midterm Project Report:

Maryam Soleimani Shaldehi

Fall 2023

Image Manipulation with Linear Algebra Introduction

The purpose of this project is to apply fundamental concepts of linear algebra to perform various image manipulation operations. Students will gain a better understanding of how matrices can be used to transform and manipulate images.

Prerequisites

Basic knowledge of linear algebra, including matrices, matrix multiplication, and transformations. Proficiency in a programming language (e.g., Python, MATLAB, or C++).

Project Overview

In this project, students will implement the following image manipulation operations using linear algebra techniques:

1. Image Resizing

Implement resizing of images using matrix operations. This includes both upscaling and downscal ing.

2. Image Rotation

Perform image rotation by applying rotation matrices. Students will learn how to transform pixel coordinates to achieve rotation.

3. Image Filtering

Apply convolution operations on the image using filter kernels. This includes operations like blur ring, sharpening, and edge detection.

4. Color Manipulation

Implement color transformations, such as grayscale conversion, color inversion, and color balance adjustments. Project Structure

Main Components

1. Image Class: Create a class or data structure to represent and manipulate images. This should include methods for loading images, performing operations, and saving results.

2. Matrix Transformation: Implement functions for matrix transformations needed for resizing and rotation. Students will need to apply scaling and rotation matrices.

3. Convolution: Develop functions for convolution operations with various filter kernels for image filtering.

4. Color Transformations: Implement methods to manipulate colors in the image, such as con verting to grayscale or adjusting color balance.

Workflow

1. Load an image from a file or generate one programmatically

2. Implement the desired image manipulation operations (resizing, rotation, filtering, and color ma nipulation)

3. Display or save the manipulated image

4. Document and explain the linear algebra concepts applied in each operation
Evaluation Criteria

Students will be evaluated based on the following criteria:

• Correct implementation of image manipulation operations

• Efficiency of the implemented algorithms

• Clarity and documentation of the code, including explanations of the linear algebra concepts used in each operation

• The visual quality of the manipulated images

In this project I have come up with this idea to convert the image to a numpy array which is a matrix representing the pixels of the original image and by using the features and operation of matrices we can implement the following functions:

apply_grayscale(input_image_path, output_image_path): This function converts an image to grayscale by calculating the grayscale values for each pixel using the formula Y = 0.2989 * R + 0.5870 * G + 0.1140 * B. The resulting grayscale image is saved and displayed.

inverse_image_colors(input_image_path, output_image_path): This function inverts the color values of an image by subtracting each channel value from 255. The inverted color image is saved and displayed.

color_balance_adjustment(input_image_path, output_image_path, red_factor, green_factor, blue_factor): This function applies color balance adjustments to an image by multiplying the RGB values of each pixel by the provided factors. The resulting balanced color image is saved and displayed.

rotate_image(input_image_path, output_image_path, angle): This function rotates an image by the specified angle in degrees. It calculates the rotation matrix and applies it to each pixel using an affine transformation. The resulting rotated image is saved and displayed.

downscale_image(input_image_path, output_image_path, scale_factor): This function downscales an image by the specified scale factor. It creates a new blank array to hold the resized image and performs bilinear interpolation to estimate pixel values from the original image. The downscaled image is saved and displayed.

upscale_image(input_image_path, output_image_path, scale_factor): This function upscales an image by the specified scale factor. It creates a new blank array to hold the upscaled image and performs bilinear interpolation to estimate pixel values based on adjusted original coordinates. The upscaled image is saved and displayed.

Notice: in the project I have downscaled the image by the factor of 0.25 and then upscaled the result image in downscale def by the factor 2 in upscale factor.

blur_imgage(input_image_path): This function applies iterative blurring to an image using a predefined blur kernel. It performs convolution with the blur kernel iteratively for a specified number of iterations to increase the blurring effect. The increased blurred image is displayed.

sharpen_imgage(input_image_path): This function applies sharpening to an image using a predefined sharpening kernel. It performs convolution with the sharpening kernel for each channel of the image to enhance edges and details. The sharpened image is displayed.

The edge_image function works the same as sharpen and blur by using an appropriate kernel.