Programming Assignment 1

W. Daniel Hiromoto

10-01-2024

Contents

Programming Assignment 1

Code can be found at Github

Program 1 - Affine Map

We have the following affine map:

$$\begin{bmatrix} x1 \\ x2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

This affine map can be converted into homogenous coordnates in 3 Dimensions to include the translation:

$$\begin{bmatrix} 1 & 1 & 4 \\ 1 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This can then be decomposed into the following transformation matricies:

$$\text{Shear: } \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ Rotation: } \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ Scale: } \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ Translation: } \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In order to apply these transformation to an image in python, we'll first import a few external libraries, namely, `numpy`, `imageio`, and `matplotlib.pyplot`

After loading an image of a black square on a white background, we need to itterate over every pixel of this image. Before we can do so, we need to understand how images are stored once read by the imageio library.

```
i1 = iio.imread('blackSquare.png')
```

Figure 1: 50 x 50 Black Square on White Background

Image Data Structure

Instead of every pixel value having a coordinate, it is placed in an array, with the “Y-Coordinates” Stored first followed by the “X-Coordinates”. This data structure will be a 3D array, as the color data (RGB) of every pixel is stored in an array of length 3. For simplicity sake, I will only be working with images that do not have an alpha/transparency channel:

```
[
    [[255, 255, 255], [255, 255, 255], ..., [255, 255, 255]],
    [[0, 0, 0],       [0, 0, 0],       ..., [0, 0, 0]],
    [[255, 255, 255], [255, 255, 255], ..., [255, 255, 255]],
]
```

Data Structure of images when read

The “Coordinate” values will corresspond to indexes in the image array. I.e. `i1[y, x, :]` will get the pixel data at coordinate x,y. Finally, the origin begins at the top left of the image, and the positive Y coordinates go to the bottom of the image and positive X coordinates go the right of the image.

Transformation Algorithm

The algorithm for applying these transformations will go as follow:

1. Define the size of the output image
2. take the inverse of a given transformation matrix
3. itterate over every pixel in a defined output image
4. for every pixel in our output image, find it’s equivalent location in the original image using the inverse transform
5. implement the Bilinear Interpolation algorithm to determine the value of the pixel in the output image