CENG466 THE1

Fall 2022-2023

1st Ali Kömürcü
Department Of Computer Engineering
Middle East Technical University
Ankara, TR
ali.komurcu@metu.edu.tr

2nd Muhammad Haseeb Motan
Department Of Computer Engineering
Middle East Technical University
Ankara, TR
e239807@metu.edu.tr

***Abstract*—This document is a report for the first assignment of our image processing course. We were assigned two tasks, namely, affine transformation with linear/cubic interpolation and histogram equalization of the given images.**

***Index Terms*—affine, rotation, contrast, histogram**

I. Introduction

We have researched how rotation and interpolation is handled in image processing, and how to enhance image with the use of histogram methods.

According to the assignment, our task is mainly handling Affine Transformations and Histogram Equalization.

A. Affine Transformation

First of all, images are represented as matrices. And, in order to rotate an image in display screens, we need to use a transformation method, **Affine Transformations** in this case. We used Affine Transformation Matrix to rotate the image. We used the matrix

$$A = \begin{bmatrix} \cos(\theta) & sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

and the θ is the angle that we want to rotate the image. If we want to rotate the image, we need to make some work to make image look smooth, namely **interpolation**.

And there are various types of interpolation:

1) Linear Interpolation: Linear interpolation is a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points [1].

To formulate, $p = xp_1 + (1-x)p_2$. Where p is the point that is the result of interpolation and interpolation occurs between $p_1 \& p_2$ points, x is the distance between $p and p_2$. Therefore, in this case, linear interpolation is nothing but a weighted sum of two points.

2) Bilinear Interpolation: To bilinearly interpolate a pixel value in 8-neighbourhood system, firstly linearly interpolate in x or y axis, and then linearly interpolate in the other (x or y) axis with using the neighbour pixels.

Note that, the distance between pixels and the centre pixel here can be thought as *city block distance*.

3) Cubic Interpolation: As it may be inferred from the name, cubic interpolation uses a larger neighborhood i.e. 16 neighboring pixels. A system of 16 linear equations with 16 unknowns is formed. Once the unknowns are found, we may use the function to find out the brightness value / color of the new interpolated pixel.

B. Histogram Equalization

One of the techniques used to enhance images with poor contrast is Histogram equalization. Images with a wide range of color are defined as images with high contrast. Therefore, this technique aims to equalize the frequency vs. brightness value histogram of the image.

II. Questions

A. Affine Transformations

- In order to rotate an image we multiply the image by affine matrix, which is defined in Introduction section in eq. (1).

$$Rotated\_Image = A \times Image \quad (2)$$

- After this multiplication, due to nature of the most displays, the coordinate of the pixels are distributed along new orientation. Since we can not represent a perfect diagonal line in pixel space, our image turns out to be include some point pixel mismatches. Therefore, some empty pixels occur in the resulting image matrix (see Figure 1).

Fig. 1.

- To get rid of this black points, we need to implement interpolation. In this assignment we used bi-linear and bi-cubic interpolation.
- Since the assignment allowed us to use libraries, we chose to use scikit-image which provides us with a function

named "rotate". The function takes as argument the image as a numpy array and the angle by which to rotate it.
- In addition to these, the function also takes as argument the interpolation technique it would use to smooth the image. The parameter for the interpolation choice is named "order". We used order values 1 and 3 which correspond to Bi-linear and Bi-cubic respectively. It finally returns the image as a numpy array.
- Besides, we derived our own algorithm which makes rotation with desired angle, and performs bi-linear interpolation on the image.

B. Histogram Equalization

First, the original histogram is generated. Then, we get use those values to generate a cumulative frequency distribution of the image against brightness values. We divide that by the number of pixels in the image to obtain a cumulative *density* function (CDF) and then multiply it with $L - 1$, where L is the number of possible brightness values for a pixel. This distribution that we obtained shows us what brightness value that the already existing ones will be mapped to. For example, if the CDF plot passes through $(50, 10)$, every pixel in the original image which has a brightness value of 50 should now instead be 10.

1) Bonus Question: We used a method in *opencv* called Contrast Limited Adaptive Histogram Equalization(CLAHE). In this method, image is divided into small blocks that are called *tiles*. Then histogram equalization on each tile is applied.It will be amplified if noise is present. And a contrast limit is applied to each tile in order to avoid noise. If there is a tile that has noise greater than the limit, then those pixels are cut and uniformly distributed to other tiles before applying histogram equalization. Bilinear interpolation is applied to get rid of the artifacts in the tile borders [2].

III. Problems faced

A. Mapping of Pixel Values

One problem we faced when we used the library scikit-image was that this library scales the integer brightness values in the image to floating point numbers between 0 and 1. We noticed this when the function was returning completely black images. For this reason, we first used a function to import the image as a floating point image and converted it back to an integer image after processing it.

References

[1] Wikipedia contributors. (2022, August 18). Linear interpolation. Wikipedia. https://en.wikipedia.org/wiki/Linear_interpolation

[2] Histograms - 2: Histogram equalization. OpenCV. (n.d.). Retrieved November 5, 2022, from https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html