CENG-466 THE-I REPORT

1st Mert Kaan YILMAZ
Computer Engineering
Middle East Technical University
Ankara, Turkey
e2381093@ceng.metu.edu.tr

2nd Ayberk Gökmen
Computer Engineering
Middle East Technical University
Ankara, Turkey
e2380442@ceng.metu.edu.tr

I. Introduction

In this assignment we have used several different libraries. We used pillow for image processing, matplotlib for drawing histograms, and numpy to extract histogram data.

II. Affine Transformation

In this part our task is to rotate a given image. Therefore we used pillow library to use already defined methods.
For all the transformations, we started by reading the image. Then, we called rotate_image function that we have written to do the operations. In this function, we call rotate method for image with the given parameters degree and interpolation_type (linear or cubic).

A. Degrees with 90 and Its Multiplies

When the angle is 90 degree or its multiplies, rotate method checks if image's width and height is equal. If it is, we return the transposed image directly, because we do not have to do interpolation for square images. On the other hand, if the image is not a square, we do matrix calculations as below.

B. Degrees that are not 90 and Its Multiplies

For this degree values, rotate methods is not able to do transpose operations directly, since the pixel positions does not match after they are transposed. Therefore matrix calculations are required. At this point, function checks if the expand and center fields are available as arguments. These fields are optional and the output image changes whether you use it or not. We agreed to set these parameters as None. The first reason is that we want to rotate our image around its center, therefore we set the center argument as None. The second reason is a little bit about the implementation. When you set the expand parameter as None, the matrix operation cut the corners out. Otherwise it expands the image size and does not let those corners being cut.
After these checks, required transform and rotation matrices get calculated and multiplied with image's matrix.

```
2229        matrix = [
2230            round(math.cos(angle), 15),
2231            round(math.sin(angle), 15),
2232            0.0,
2233            round(-math.sin(angle), 15),
2234            round(math.cos(angle), 15),
2235            0.0,
2236        ]
```

Fig. 1. Affine transform matrix defined in the library

III. Analysis of The Affine Transformation

As we said earlier, if the image gets rotated with 90 degree or its multiplies, we don't need to do interpolation, since transposing the image is enough. If the angle is not like that, we should interpolate it to get the image. This extra operations affect the image a little. In examples below, there is no difference between cubic and linear transformed images if you rotate them with 90 degrees. On the other hand, there are small differences between images in terms of sharpness/smoothness when you do interpolation. The linearly transformed image looks a little bit smoother than the cubic transformed one.

Fig. 2. 90 Degree Rotation. Linear on the left, bicubic on the right

Fig. 3. 45 Degree Rotation. Linear on the left, bicubic on the right

IV. Histogram Equalization

As it was done in the first part, we read the image and save it. Afterwards, we try to extract the histogram of that image by using histogram method in numpy library. This method gives us the axis values of the relevant histogram. Then, we use that data to draw the original histogram and save it as original_histogram in Outputs folder. For the next step, we have written histogram_equalization function to get the equalized histogram of the image. To get that output, we used ImageOps.equalize method. It basically, generate the histogram of that image and equalize it and then converts it back to an image, which is the equalized image. Then, we extract that equalized histogram of that equalized image, and save it as an image in the Outputs folder.
Lastly, we generate equalized image by using write_image function with equalized histogram and save it as enhanced_image in the Outputs folder.

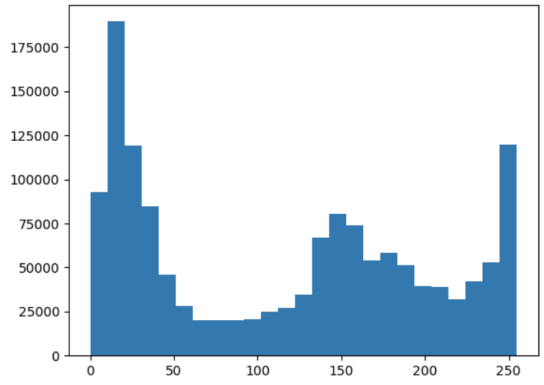

Fig. 4. Original Histogram

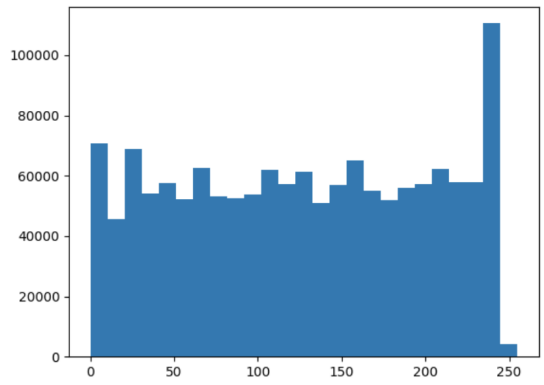

Fig. 5. Enhanced Histogram

Fig. 6. Enhanced vs Original Images

V. Analysis of The Histogram Equalization

As we can easily see from the images, the enhanced one looks much more brighter, but when this image enhancement operation is being done, the brighter parts of the image does not get more brighter. It only affects the darker parts of the image and makes it much more visible.

VI. Requirements and Dependencies

Required libraries:

- numpy
- PIL
- matplotlib

Installattion: pip install numpy PIL matplotlib

There is no required dependency.