Computer Vision HW2 Report

Ertürk Ocak - 150200048
Baturalp İnce - 150200738

07 November 2023

1 Introduction

In this Homework, we practiced and created functions to achieve image filtering with 3x3 and 5x5 windows, image upsampling with factor of 2 and then linearly interpolating it, image downsampling and rotating image by 45, 90, 135, 180 degrees.

2 Q1 - Image Filtering

In the first question, we needed to first convert RGB image to Grayscale just like the previous homework. Then for the window of 3x3 and 5x5 filtering operations we created according functions. Inside the filtering function, we first applied the zero padding to prevent changing the size of the image. We created 4 functions in total which do zero padding to RGB, 3x3 filtering to RGB, 5x5 filtering to RGB, zero padding to grayscaled, 3x3 filtering to grayscaled and 5x5 filtering to grayscaled.

For Zero padding on RGB, we added the stride to the image which resulted with the padding but also we needed to add the RGB channel data to the new array that we created, it is very similar with the previous homework. Because we need this channel data when we will do the 3x3 and 5x5 filtering. For zero padding on Grayscaled, it was very similar with previous homework. We just created two different functions to separate these tasks to be more clear and understandable when checking the code.

Figure 1: Grayscaled

Figure 2: 3x3 Filtering Grayscaled

Figure 3: 5x5 Filtering Grayscaled

As it could be seen from Figure 1, 2 and 3. 3x3 Filtering blurred the original grayscale image a bit. 5x5 filtering put more blur on the image. If images are not visible please check project folder because we can put this size to Latex.

For Grayscaled images, we did 3x3 filtering with one implementation and we did 5x5 filtering with a different implementation. That's why we created two different functions and wanted to practice different kinds of implementations to improve us. In 3x3 implementation we used a mask matrix of 1/9 coefficient and convoluted with the 3x3 window of the respective pixel. In 5x5 implementation, instead of mask, we accessed nearby pixels of the center pixel by two for loops and as a trade off we think that the performance is slower in this implementation.

Figure 4: RGB

Figure 5: 3x3 Filtering RGB

Figure 6: 5x5 Filtering RGB

In RGB, as it could be seen on Figure 4, 5 and 6, blur can be seen similar to grayscale version. Difference is that this filtering operations change the color of the image because while we were taking the mean around the pixels for interpolation, for each channel we took the mean around as well, this resulted to R, G and B values to change.

For RGB images, implementation is very similar to Grayscaled functions, only difference is that we calculated means for each channel of the each RGB pixel separately. For 3x3, for each channel we check the neighbouring pixels and calculate mean with a mask. For 5x5, we did not use mask matrix like 3x3, we dynamically iterated with two extra for loops to calculate mean of the window for each pixel and for each channel.

3 Q2 - Image Upsampling

In this question, we needed to create functions to do the upsampling of an image. We first did the grayscale conversion from the RGB image like the previous parts. Then we upsampled the grayscaled image by factor of 2. This resulted that every second pixels from each height and width will be replaced with 0 (black) pixels which will be replaced by the mean of the known pixels around with interpolation. In interpolation function, we first zero pad the image so that we will not go past beyond limits of the image size or not change the size of image. Then, we iterate over each second pixel because we want to only reach to unknown black pixels. At each iteration we double check if the pixel is black. If it is black, we will start controlling the pixels surrounding the black pixel but we only take known pixels into consideration that's why we have a counter variable. Then we get the mean around of black pixel and assign that mean to the black pixel. Resulting pictures and comments are on next pages.

Figure 7: Grayscaled

Figure 8: Upsampled Grayscale

Figure 9: Upsampled Interpolated Grayscale

On the Figures 7,8 and 9, it could be seen that when the image is upsampled every second pixel for width and height is black. As a result image appear to be dark. In the Figure 9, when it is interpolated those black pixels replaced by the mean of the surrounding known pixels. Therefore, the image looks more like the grayscale image and there are no full black pixels. But since we use only 3x3 filtering, it is still a bit dark compared to bigger filter boxes.

4 Q3 - Image Downsampling

On the downsampling part, we first linearly interpolated the image with 3x3 filter so that when we downsample it is smoother. On the downsampling function, we halved the width and height of the image, this operation is done as: - We take every second pixel of the linearly interpolated image and assign it to downsampled empty image.

Figure 10: Grayscaled

Figure 11: Downsampled Grayscale

In the downsampled image, it is more blurry compared to grayscale, downscaled images width and length of the picture is half of grayscale. Please check images inside project folder, because difference is more clear and visible on those images.

5 Q4 - Image Rotation

We grayscale and 3x3 filter the image as like the previous parts. We created a rotated_grayscale_img function which receive degree such as 45, 90, 135 and 180 as a parameter. In the first line of the function it converts the degree to radian. Then calculates the new width and height according to the degree, input image size and width. Then we calculated mid width and mid height for new and old images because we want to rotate images with respect to the center. Then we iterate over each pixel. According to the given formula for finding the position of the pixel belonging to the old image with the given pixel of new image is calculated. In this calculation, we also have to consider the mid x old and mid y old. So that we subtract those from the position of the pixel of the new image. After finding the position of the pixel of the old image we have to add mid x old to the mid y old and vice versa because of the subtraction before. We add x to the y and y to the x because of the counter clockwise rotation. But at the end we need to check if the position calculated is actually in the borders of the original image and it has an intense value. For each degree requested, we printed the images but for 45 and 135 degree rotations we also linearly interpolated the images after rotation since it was asked like this in the question. Resulting Images and descriptions are on the next pages.

Figure 12: Grayscaled

Figure 13: 45 Degree Rotation

Figure 14: 45 Degree Rotation Interpolated

Figure 15: 90 Degree Rotation

Figure 16: 135 Degree Rotation

Figure 17: 135 Degree Rotation Interpolation

Figure 18: 180 Degree Rotation

Rotations happened as expected. Since we use the correct calculation for new height and new width, rotated images fit to that size. Differences between 45 degree rotation and 45 degree rotation interpolated and also the 135 degree rotation and 135 degree rotation interpolated could be seen. Interpolated ones are more blurry.