Ertuğrul Aypek, 2171270

Answer to Question 1 – Image Interpolation

Concept of interpolation: Interpolation is widely used in image processing especially on resizing purposes. When we are resizing an image, since the dimensions of input and output images are different, there is no 1-1 correspondence between them. Thus, we need to guess what will be the intensity value at each pixel of the new image by utilizing the input data. Image interpolation does this mapping from input image to output image with different image sizes.

Bicubic and bilinear interpolation: The name bilinear comes from two linear operations which are applied on x and y axis. When using bilinear interpolation, for each pixel (i,j) in output image, we calculate where that pixel corresponds in input image. We do that simply by multiplying (i,j) with rescaling factors at x and y axes. After calculating where (i,j) coordinate maps in input image, bilinear interpolation uses four neighboring pixels which form a 2x2 grid. Since the output image's pixel coordinate mapping to input image might be a floating point coordinate, we also weight the contribution of 2x2 pixels seperately by considering inverse proportion of distance. Hence, closer pixels to the mapped coordinate will affect the output intensity more.

With bicubic interpolation, we use a 4x4 grid of neighborhood instead of 2x2. However, weight distribution is not the same with bilinear interpolation which was inverse proportion of distance. Since bicubic interpolation works on a larger grid, the outputs of it is smoother than bilinear interpolation.

As a result, in bilinear interpolation we calculate 4 weights for a 2x2 grid and in bicubic interpolation we calculate 16 weights for a 4x4 grid.

Matlab's imresize function:

Imresize is a useful function which is used to shrink or expand image resolution.

There are several parameters that the function uses:

I – input image aimed to be resized: This can be a grayscale or an RGB image or image with any number of channels.

Scale – coeffienct of resizing: This value is used to determine the resized image's resolution and must be positive since a resolution can not be negative. Since we can both shrink and expand images, if this value is greater than 1, than the image gets larger; otherwise, the image gets smaller in resolution.

[rows, columns] – Resized image's resolution: Other than scale, this parameter (array with to elements) is just another way of expressing the output resolution. In the output, we will get an image with dimensions [rows,columns] while preserving the number of channels for images with depth greater than one.

Method – interpolation method to be used: This parameter determines with which interpolation algorithm we scale our image. And possible methods are:

- ‘nearest’ which considers only the nearest pixel while assigning a pixel value in the output image.
- ‘bilinear’ which considers a weighted 2x2 grid while assigning a pixel value to the output image.
- ‘bicubic which considers a weighted 4x4 grid while assigning a pixel value to the output image.

How it works: Imresize function first extracts the point where a pixel in output image maps on the input image. After getting that point, according to the used method, it generates an nxn grid. Before matrix multiplication of two nxn grids, one is the kernel and the other is a portion of input image, the kernel’s weights are determined according to the method’s algorithm. For example bilinear interpolation uses inverse proportion method which assigns a higher weight to the elements of the kernel which are more closer to the mapped point. Then two grids are multiplied and one of the pixel values of output image is determined.

Results:

A1_bilinear_euclidean_distance: 5.8520e+03

A1_bicubic_euclidean_distance: 6.0862e+03

A2_bilinear_euclidean_distance: 2.8569e+04

A2_bicubic_euclidean_distance: 2.9940e+04

A3_bilinear_euclidean_distance: 5.3166e+03

A3_bicubic_euclidean_distance: 5.4198e+03

Comment on findings:

As it can be clearly seen from the results, with bicubic interpolation we have more Euclidean distance compared to bilinear interpolation. However, there is conflict with (Gonzalez, p.88) which claims the following:

“Generally, bicubic interpolation does a better job of preserving fine detail than its bilinear counterpart.”

However our cases are not in “generally” part.This is because the resolution of our input images are not at a high level and the shrinking coefficients are not at low level. Due to these facts, while calculating a pixel intensity by applying bicubic interpolation on a 4x4 grid, we consider a variety of objects. As a result, each pixel in expanded includes the noise and smoothness of a large region which leads to a larger distance. On the contrary, since bilinear interpolation works on a 2x2 grid, there are less noise; thus, less distance.

Bilinear interpolation method beats bicubic interpolation method here in our examples. However, I think that the opposite results will be out if we use an image with more resolution and less shrinking coefficient.

Examining our results, the largest distance comes from A2.jpg which has 120x63 resolution on shrinked image and 8 as resizing coefficient. In addition to these, the objects in the image are very close to each other. Because of that we have more than one object in a single grid

Answer to Question 2 – Histogram Processing

Histogram matching is a nice algorithm which aims to generate a mapping function from pixel values to pixel values. The algorithm needs two images one is the base image whose histogram will be the target histogram and the other is the image whose histogram will be changed. So, the aim of the algorithm is to find transition matrix from reference image to target image.

Histogram matching is a kind of style transfer. After application of it, we get an image that preserves its objects but with a different color tones, i.e different texture.

There are two main methods when applying histogram matching on RGB images. One of them is to divide the image into single channels, match each channel's histogram seperately and concatanate the resulting channels to form an RGB image again. The other method is to convert each image into grayscale and apply single histogram matching. However, with second method we lose color information, so I used the first one.

Since B1 and B3 have a larger variety of colors compared to B2 and B4, B2_histmatch_output and B4_histmatch_output look like to contain more noise.

Answer to Question 3 – Noise Elimination

I observed that there are more noise in one of the channels of C1 and C2 compared to the other channels.

As it is stated in (Gonzalez, p.473-476), converting the image from RGB to HSI format to denoise color images is useful and by doing that we also spread the noise in one channel to entire HSI format. Furthermore, applying averaging filters gives good results according to the same source.

Thus, I decided to use a 5x5 averaging filter to reduce the noise in the images. By applying this filter, the noise in images becomes less visible, i.e we get a smoother image.

In order to detect edges, I used Sobel edge detection algorithm which uses two different kernels for two axes of the image. Sobel algorithm gives higher responses to the input grids where there is a rough pixel intensity value change. After applying sobel edge detection algorithm to each channels seperately, I applied an elementwise maximum operation on Sobel outputs on x and y axes. The final result is grayscale version of the edge detected image. Other than Sobel operation, I applied average smoothing operation with 5x5 kernels both before and after the Sobel edge detection phase.