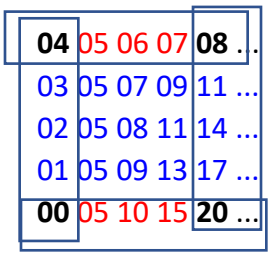
Lab3 Report Shuchen Zhang (szhan114)

Introduction: In this lab, we explore the image interpolation techniques including image resizing, image compression and image restoration. In part 1, we use two algorithms Zero Order Hold and Bilinear interpolation to downsample the image to create low-resolution image. In part 2, we explore the image decimation algorithms including decimation without anti-aliasing filter and decimation with an ideal low pass filter. We also apply image restoration using bilinear interpolation and compare the restored image with original one with mean square error method.

In part 1, we do the image interpolation of a specified 128x128 block and turn it into 512 x512 image. When we use Zero Order Hold algorithm, we simply replicate the each pixel value with the top left value of subblock to fill the corresponding 4x4 subblock in output image. In Bilinear interpolation algorithm, we also take the top left value of 4x4 subblock. But instead of replicate for each pixel with that value, we perform a computation of linear combination of neighbored copied pixels which make the output image much more smoother. The order of doing interpolation along row first or column first doesn't matter. The copied pixel value which is the top left value of 4x4 subblock from input image are fixed in the output image.

For instance if we do row first(red) and then columns (blue):



05: A2' 04:A2 and A1 08: B2

03: A1' 11: B1'

05: B2' 00: B1

The linear interpolation equation is given by : new_value[i] = ((1-(i/distance)) * A) + ((i/distance) * B) where A and B are the copied values from original image.

The four corner values are always fixed, and also the circled values are always fixed too because of they have fixed A and B value taken from the corner values. And these fixed values are regarded as A' and B', and the values of the middle region are calculated with the same equation new_value[i] = ((1-(i/distance)) * A') + ((i/distance) * B'). Then we just need to prove the middle region part. In the middle region, for each row, we have new_value[i]=((1-(i/distance)) * A1') + ((i/distance) * B1') where i is the index of columns with A1' as reference point. We have new_value[j]=((1-(j/distance)) * A2') + ((j/distance) * B2') where j is the index of row with A2' as reference point. A2' = new_value[i] = ((1-(i/distance)) * A2) + ((i/distance) * B2),

B2' = ((1-(i/distance)) * A2) + ((i/distance) * B2), A1' = = new_value[i] = ((1-(i/distance)) * A1) + ((i/distance) * B1), B1' = ((1-(i/distance)) * A1) + ((i/distance) * B1), As we plug in A1', B1' and A2', B2', we can get the result that new_value is the same for both horizontal and vertical calculation for any pixel in middle region. An example is given above with labels by the number matrix.

In part 2, we use two decimation method to subsample the image and restore it and compare the restored image with original one.

Mean square error for method A with decimator factor 2: 770.562

Mean square error for method A with decimator factor 8: 3116.24

Mean square error for method B with decimator factor 2: 31615.1

Mean square error for method B with decimator factor 8: 31609.4

Rank of images from best to worst according to the visual quality of interpolated output images: Method B with M=2, Method A with M=2, Method B with M=8, Method A with M=8.

Method B is the low-pass filter method and Method B is the decimation without anti-aliasing.

As subsampling rate increase, the MSE increases as well. Because the restore image from smaller image which is generated from larger subsampling rate will lose more information than those with smaller subsampling rate.

MSE is correlated with my subject evaluation. MSE compare the restored image with the original image and find the difference. The anti-aliasing low-pass filter perform better than non-anti-aliasing method.

The method B and D is equivalent to convolving the input image with an ideal interpolation sinc function then sub-sampling in spatial domain. Because the ideal low-pass filter is derived from computing the inverse fourier transform of H(f) and will have result in sinc(x) which is

$\sin(\pi x)/ \pi x$.

The distortion in letters are more apparent than the distortion of cats when subsampling in Method A because Method A takes 2x2 subblock from original image. Letters forms its shape from the combination of pixels. The neighbors of pixel you copy to the smaller image may have huge difference with the copied pixel. Letter will become much less recognizable in this way.

Comparing images obtained from method A and B, we can see that the main difference comes from the letters. The letters in method A is not quite complete compared with letters in method B which letters with complete shape.

Conclusion: In part 1, the bilinear interpolation have smoother blurring effect than zero order hold method. In part 2, decimation with low-pass filter performs better than the method without anti-aliasing.

Note: To run my code for part 2, set the flag value to 0 for non-anti-aliasing method and 1 for low-pass filter method. Set the deci value to either 2 or 8. When I implement argv, it keeps giving me floating exceptions. So I have to set the value in the on my own in the script. Sorry about the inconvenience.