CENG466

Fundamentals of Image Proccesing

Fall 2019-2020

Take Home Exam 1

Yunus Emre Gök
Computer Engineering
Middle East Technical University
2166460

I. Image Interpolation

Interpolation is making assumptions for unknown values of a function or pixel values for our subject by manipulating known values related to the location of values we want to estimate. These manipulations are different mathematical approaches to get an ideal value like nearest neighborhood, linear or cubic interpolation.

A. Bilinear Interpolation

In linear interpolation at 1D, an unknown point p between two known points a and b (nearest neighbors) can be estimated by making a linear function y = mx + n *with* a and b. The distances from the p will give weights of a, b and get a approximate value for p.
In 2D, Bilinear interpolation uses nearest four nearest neighbors' weights on the location of unknown value.
To find estimated value, the equation (1) can be used.

$$P(x,y) = ax + by + cxy + d \qquad (1)$$

where the coefficient can be determined from the neighbors.
To illustrate, a point P at coordinate (x, y) and neighbors are Q11(x1, y1), Q12(x1, y2), Q21(x2, y1), Q22(x2, y2).

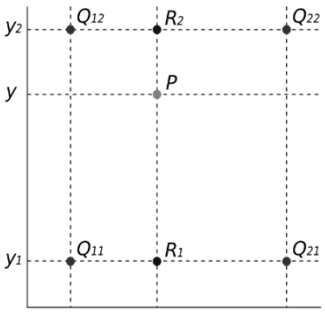

Figure 1: Four neighbors of point p-[1]

Since they all are pixels, the distance between them on both horizontally and vertically is 1. The weights of the neighbors on a point p can be found by,

$$\begin{aligned} P_1(x,y) &= Q(x_1,y_1) * (1-(x-x_1)) * (1-(y-y_1)) \\ P_2(x,y) &= Q(x_1,y_2) * (1-(x-x_1)) * (1-(y-y_2)) \\ P_3(x,y) &= Q(x_2,y_1) * (1-(x-x_2)) * (1-(y-y_1)) \\ P_4(x,y) &= Q(x_2,y_2) * (1-(x-x_2)) * (1-(y-y_2)) \end{aligned} \qquad (2)$$

Hence the estimated value on P is the total weight of the neighbors,

$$P(x,y) = P_1(x,y) + P_2(x,y) + P_3(x,y) + P_4(x,y)$$

B. Bicubic Interpolation

In 1D, from the values of two points and their derivatives, a third degree polynomial like

$$\begin{aligned} f(x) &= ax^3 + bx^2 + cx + d \text{ where} \\ f'(x) &= 3ax^2 + 2bx + c \end{aligned} \qquad (3)$$

can be written to use the estimate the values between them.
In 2D, Bicubic interpolation uses nearest sixteen nearest neighbors' weights on the location of unknown value. The additional twelve neighbors to bilinear interpolation is for the find the derivatives.
To find estimated value, the equation (4) can be used.

$$\mathbf{P(x,y)} = \sum_{i=0}^{3}\sum_{j=0}^{3} \boldsymbol{a_{ij}\, x^i y^j} \qquad (4)$$

The sixteen coefficients are determined from the sixteen equations with sixteen unknowns that can be written using the sixteen nearest neighbors of point (x, y)[2].
Since, the all of sixteen points have contribution on the estimation value the image smoother.

C. Resize Function From Matlab[3]

From Image Processing Toolbox, "imresize" function changes the size of an image depends on the input arguments.

1) Description

- B = imresize(A, scale) returns resized image of A in given scale using the bicubic interpolation by default. When scale is in the interval [1.0] B is smaller than A.
- B = imresize(A,[row, col]) scales image A and returns image B with given sizes. It is useful for nonlinear scaling and able to change height/width ratio.
- imresize(___ , method) specifies the interpolation method used

2) Input Formats

- A is the source image to be resized.
- B is the output image.
- Row and col are specific size for the resizing process.
- Method is the interpolation method which can be 'nearest', 'bilinear' or 'bicubic'.

3) imresize(A,[row,col],'bilinear')

In the the1_partA, this function is used. It takes 'AX_shrinked.jpg' and the size of 'AX.jpg' as arguments and resized it with bilinear interpolation.
In this function, it creates an image to given sizes. For every x and y value it scales with the ratio of shrinked size to original size. to fit in the shrinked image. Than it finds the nearest 4 neighbors and their distances to the x, y values. With these approach, the weights on that point and the estimated value can be calculated.

4) imresize(A, [row,col],'bicubic')

In the the1_partA, this function is used. It takes 'AX_shrinked.jpg' and the size of 'AX.jpg' as arguments and resized it with bicubic interpolation.
In this function, it creates an image to given sizes. For every x and y value it scales with the ratio of shrinked size to original size. to fit in the shrinked image. Than it finds the nearest 16 neighbors. With every neighbor it calculates the coefficients in the equation(4). After finding coefficients, the estimation value on x, y can be found.

D. Comparison

To compare the images, we can find the Euclidian distance between RGB values of images. The Euclidian distance of two RGB values is

$$d(rgb_1, rgb_2) = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} \quad (5)$$

	Original Image vs Bilinear Result	Original Image vs Bicubic Result	Bilinear vs Bicubic Result
A1	1.8243e+06	1.8837e+06	2.9561e+05
A2	1.3647e+07	1.4190e+07	2.3076e+06
A3	1.7963e+06	1.8304e+06	3.6517e+06

Table 1 .Total Euclidian Distance Between AX result bilinear.jpg and AX.jpg AX result bicubic.jpg and AX.jpg

E. Conclusion

In bilinear interpolation, using four neighbors makes a basic assumption on the values in between. Since there are only four coefficients to estimate the value, the calculations for the bilinear interpolation does not take to much time and useful for faster response.
In bicubic interpolation, using sixteen neighbors make more complicated assumption since estimation comes from both points and derivatives on the points. There are sixteen coefficients to calculate, the time consumption is more than bilinear interpolation.
On the other hand, As we can see on the table(1) in part D, the comparison between the original image and result images, Difference is higher between bicubic result and original image. This shows that bilinear result is close to image in pixel wise. However, bicubic interpolation makes more smooth and burry image.
To conclude, they both can be useful for the purpose of use. For example, in A3 since there is nearly no difference in look bilinear can be faster and better method and bicubic can be used for more smooth images.

II. Histogram Processing

A. Histogram

Image histogram shows the color distribution of an image. Basically value count of each color and it can give clue of the image like is it dark or bright. Colorful or monotone.

1) Algorithm

- *Histogram*

Pixel value of a gray scale image is in the range [0,255]. So we can go one by one for every pixel and increment the count of the value in assigned array.

- *Cumulative Probability*

If we divide the histogram values to total pixel count, we can find the probability of each value in the image.

The cumulative probability is the sum of the probability of previous and the current.

$$c(x) = p(x) + p(x-1) \qquad (6)$$
$$c(0) = p(0)$$

B. Histogram Equalization

Histogram equalization is getting close to a uniform distribution to enhance contrast by stretching the range of the image.

C. Histogram Matching

In histogram matching, we try to get close a target distribution like histogram equalization. However, this time target is another image or histogram.

1) Algorithm

Since we know how to equalize histogram, we can go form both target image and source to equalized histogram and turn the matched one.

$$T_1(r_k) = s_k = T_2(z_k)$$
$$z_k = T_2^{-1}T_1(r_k) \qquad (7)$$

- *Mapped*

As shown in equation (7) we need to find map to get matched image.

$$M(r_k) = T_2^{-1}T_1(r_k) \qquad (8)$$

To find map, we first find $T_1(r_k)$ and $T_2(z_k)$ which are cumulative distribution function of source and target like showed in figure 2.

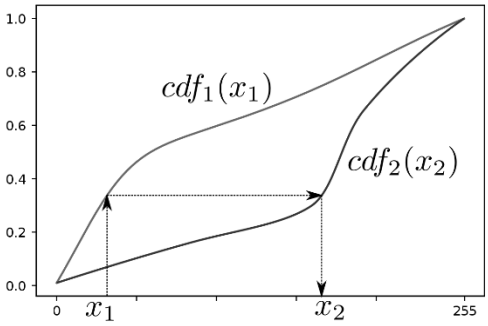

Figure 2: cdf functions of source and target

Since pixel values are discrete we need to find closest values and their index.

Since, RGB images have three pixel values, this process should be done for every red, green and blue value.

Giving the source map to match with the target histogram to be equalized.

D. Conclusion

Equalized image B1 with the reference B2 returns a dark and greenish image which close to color distribution of B2. Also enlarges the shadows that bare eye cannot detect.

Equalized image B2 with the reference B1 return a bright and disturbed image which close to color distribution of B1. Although the distribution, it can be useful to analyze the shapes left in shadows.

Equalized image B3 with the reference B4 return a dark and blueish image which close to color distribution of B1. This match separates the clouds and the sun so clouds can be detected easier than the original one.

Equalized image B4 with the reference B3 return a bright and disturbed image like B2 which close to color distribution of B3. Also this one can be useful to analyze the shapes left in shadows like buildings. This matching also gives a clue about distances since ground and the sky is brighter.

III. Noise Elimination

A. Convolution

The mathematical definition of convolution is an integral that expresses the amount of overlap of one function g as it is shifted over another function f. It therefore "blends" one function with another. [3]

For images, with a mask h results returns the intensity of h in the image.

$$y[m,n] = x[m,n] * h[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} h[m-i, n-j].x[m,n] \quad (8)$$

B. Algorithm

- In the convolution, mask and image faces reversely, and multiplication happens between the end of h and the beginning of the image. Therefore, we rotate the mask the face them directly.
- To calculate the first element of the image we extent image with the mask, and center the image.
- For every value they face, we multiply and assign to the location. After shifting to the next multiplication we add the values that overlap to make it cumulative.
- We apply this for every RGB value.

C. Denoising

- Find edges of vertical, horizontal, diagonal and sum them up.
- Make image blur with Gaussian filter to eliminate the noise
- Sum the blur image and the edges to maintain the shape.

This algorithm can delete the noise from the image. However, I could not apply this algorithm to my code.

D. Edge Detection

We can use sobel or laplacian operator for edge detection.

[1] (7th ed.) Gonzalez, Rafael C, and Richard E. Woods. Digital Image Processing. Upper Saddle River, N.J: Prentice Hall, 2002. Print

[2] https://en.wikipedia.org/wiki/Bilinear_interpolation.

[3] http://mathworld.wolfram.com/Convolution.htmlK. Elissa, “Title of paper if known,” unpublished.