INTRODUCTION

This experiment guides students through the basic functions of MATLAB regarding image processing. It helps students to build an intuitive understanding of various types of noise and denoising algorithms， leading to further study on image processing technics.

RESULTS&EXPLAINATIONS

**TASK1**

**The way to complete this task is simple: write out the expressions for MSE and PSNR using the functions available in MATLAB.**

**The code for task 1 is:**

**function [PSNR] = psnr(a,b)**

**A = double(a);**

**B = double(b);**

**C = A-B;**

**MSE = sum(C(:).\*C(:)) / numel(A);**

**PSNR = 10\*log10(255^2/MSE);%For the peak value use 255**

**end**

**Input the image uint8 data a, b into the function and convert them into high precision value format for calculation.**

**Square the difference between the true value and the predicted value then summe and average it to get MSE. Use MSE to get PSNR.**

**Note that the right conversion of the data format should be used, to get the result in numeric form.**

**TASK2**

**(a)**

**The task is to down sample the 512\*512 image to get a 256\*256 image. To implement this, split the 512\*512 image matrix into 2\*2 matrices, average each 2\*2 matrix, merge the averaged values together to form the desired 256\*256 matrix.**

**The essential codes are listed below**

for i=1:256

for j=1:256

subA=A(((1+2\*(i-1)):i\*2),((1+2\*(j-1)):j\*2));

B(i,j)=mean(subA(:));

end

end

**It is achieved using nested for loops, where each time a 2\*2 matrix is selected sequentially from the original graph and averaged, and the resulting averages are finally combined.**

**Comparing the new arrival image with the original image, it can be seen that the new image is a quarter of the size of the original.**

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

Figure 1-Generated 256\*256 image I\_1(right) and the original image I\_0(left), equally zoomed

At the same time, details are lost due to the pixel losses.

手机截图图人的脸

描述已自动生成

Figure 2-Generated 256\*256 image I\_1(right) and the original image I\_0(left), unequally zoomed in until the patterns are likely evenly displayed.

**(b)**

**The nearest neighbor interpolation algorithm selects the value of the nearest point to the sought data point to produce a segmented constant interpolated value.**

**It takes the closest of the four pixel points around a point as the pixel value of the corresponding point in the output image. This is similar to the case of the machine learning KNN (K - Nearest Neighbor) algorithm with K = 1.**

The essential codes are listed below

for i=1:row\_n

for j=1:col\_n

m=round(i/2);

k=round(j/2);

if m<1

m=1;

end

if k<1

k=1;

end

if m>row

m=row;

end

end

B(i,j)=A(m,k);

end

end

end

**This is achieved by using nested for and if statements, for each pixel, the nearest pixel point around it is selected and that point is filled with the original pixel into a blank uint8 file (B) of size 512\*512 to obtain the interpolated new image.**

**The figure below shows three images, from it is observable that nn algorithm produces an image that is the same size as the original one, four times the number of pixels of I\_0.**

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

Figure 3-The 256\*256 image I\_1(right), the nn interpolated image I\_1’(middle) and the original image I\_0(left), equally zoomed

**The blurring of the image generated by the nn algorithm after a round of down sampling can be clearly observed here in relation to the original image. It can be seen that in the case of losing most of the pixels, there is a serious loss of information.** **Traces of the squared-off processing of the images are also evident.**

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 4-The 256\*256 image I\_1(right), the nn interpolated image I\_1’(middle) and the original image I\_0(left), unequally zoomed in until the patterns are likely evenly displayed

**(c)**

Use MATLAB’s build-in imresize function to implement bilinear interpolation and bicubic interpolation. Calculate the PSNR of the image obtained by each algorithm with the original image using the function constructed in task1

|  |  |  |  |
| --- | --- | --- | --- |
| image | nearest | bilinear | bicubic |
| PSNR (dB) | 31.5456 | 32.7110 | 34.2666 |

The PSNRs obtained by the three algorithms are between 30 and 40 dB, which usually indicates good image quality (with perceptible but acceptable distortion). However, the distortions can be clearly seen with the naked eye, as shown below.

人不同表情的照片

描述已自动生成

Figure 5-Images obtained using nn(left), bilinear(down), bicubic(right) algorithms and the original image (up central), equally zoomed

**图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成**

Figure 6-Images obtained using nn(left), bilinear(down), bicubic(right) algorithms and the original image (up central), equally zoomed in until the patterns are likely evenly displayed

**The higher the PSNR, the higher the fidelity the processed image is to the original image. This is the same as what presented in figure 6, where the bicubic algorithm with the highest PSNR has the best image detail preservation. The bilinear and nn algorithm, with similar PSNRs, are difficult to judge for the better.**

**For bilinear interpolation, the implementation quality is generally not as good as the nearest neighbor algorithm. However, its effectiveness depends on the content of the image, for example in this image its PSNR is higher than that of nearest neighbor interpolation. Bilinear interpolated images tend to have blurred edges but are also smooth and look good overall.**

**The nearest neighbor interpolation algorithm, which is essentially copying pixels, yields images with sharper edges, but most images are blocky and jagged.**

**Bicubic is considered to be the best interpolation method for enlarged images.**

**TASK3**

**(a)**

**Gaussian white noise is noise where the instantaneous values obey a Gaussian distribution, and the power spectral density is uniformly distributed. Gaussian white noise can be achieved adding a matrix of normal distribution to the picture.**

**The essential code is listed below**

**function[im\_wn] =gwn(n)**

**A = double(n);**

**B = A + (0+5\*randn(size(n)));**

**im\_wn = uint8(B);**

**imwrite(im\_wn,'im\_wn.bmp')**

**end**

**Use randn function in MATLAB to add a normal distributed matrix with a mean of 0, standard deviation of 5(variance of 25).**

**As can be seen below the picture, image with Gaussian white noise seems to be covered with a light, uniform shadow, this is because the white noise tends to be constant in the power spectrum, the noise is frequency rich and has components throughout the spectrum.**

女人戴着帽子

描述已自动生成

Figure 7-Image with Gaussian white noise(left) and the original image(right)

Zooming in to the details below, the effect of the noise is clearly observable. The left image is a notch duller and blurrier than the right. As the noise obeys a normal distribution, the larger the variance parameter, the more blurred the image.

人的脸

中度可信度描述已自动生成

Figure 8-Details of image with Gaussian white noise(left) and the original image(right)

**(b)**

**The idea of image averaging for noise removal is indicated as follows:**

**An image with noise can be expressed as captured scene** **and noise .**

**The process of denoising is the process of approximating** **from the known .**

**For multiple images taken of the same scene,**  **are the same and**  **are random and uncorrelated with each other.**

**The mean value of image images of the same scene**

**The expectation of the averaged image**

**This indicates that the expectation of the mean of multiple images of the same scene is the noise-free scene, but there will be some perturbations, and the standard deviation of these perturbations determines the intensity of the noise. The essence of image denoising is to reduce the standard deviation in the spatial domain.**

**The variance of the averaged image**

**This indicated that Noise can be reduced by increasing the value of K, in another word, increasing the number of average images.**

|  |  |  |  |
| --- | --- | --- | --- |
| **image** | **im\_wn10** | **im\_wn100** | **im\_wn1000** |
| **PSNR (dB)** | **33.7092** | **34.0723** | **34.1282** |

**The images were implemented using a while loop to repeat the function from task3a, adding Gaussian white noise to get new matrices, accumulate these matrices and take the average.**

**The essential codes are listed below:**

function[C] =im\_wn(m,A)

n = 1;

B = double(zeros(512,512));

while n <= m

B = B + (A + 5\*randn(size(A)));

n=n+1;

end

C=B/m;

end

**The figure below presents the generated images versus the original image. Along with the PSNRs, it illustrated that increasing the number of images alone has little effect and is less and less effective as the number of images averaged grow.**

**This is because as the increases, the standard deviation changes less and less**

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

Figure 9-Details of Original image lenna512(right), im\_wn10(middle right), im\_wn100(middle left) and im\_wn1000(left)

**(c)**

**女人的照片上写着字

描述已自动生成**

Figure 10-im\_SP(left) and original image(right)

**This is achieved by randomly modifying the element values in the matrix of the original image(A) to 255 (adding white pixels, salt) and 0 (adding black pixels, pepper) adding the same number of black and white dots.**

**Essential codes:**

n = numel(A(:,:,1));

% 15% pixels altered

m = fix(0.15\*n);

idx = randperm(n, m);

% among all altered pixels,

% 50% percent of then are now white pixels, others are black pixels

k = fix(0.5\*m);

idx1 = idx(1:k);

idx2 = idx(k+1:end);

idx1 = idx1' + n.\*(0:size(A,3)-1);

idx1 = idx1(:);

idx2 = idx2' + n.\*(0:size(A,3)-1);

idx2 = idx2(:);

A(idx1) = 255;

A(idx2) = 0;

**(d)**

**To the naked eye, the 5\*5 median filter removes all the salt and pepper noise but loses a lot of detail, the photo is less sharp and the image is blurred. the 3\*3 median filter leaves a little noise, but less detail is lost in the image.**

女人不同表情的照片

描述已自动生成

Figure 11-details of Original image(right); im\_SP(middle right); im\_median3, 3\*3 median filter applied(middle left); im\_median5, 5\*5 median filter applied(left)

|  |  |  |  |
| --- | --- | --- | --- |
| **images** | **im\_SP** | **im\_median3** | **im\_median5** |
| **PSNR (dB)** | **13.6780** | **31.1530** | **29.6128** |

**The PSNR data suggests that the 5\*5median filter may not have the same fidelity for the original image as the 3\*3median filter due to the loss of too much detail.**

**(e)**

**The idea of mean filtering is to replace the value of a pixel in an image with the mean of the values of the pixel points in one of that pixel's neighbors, so that the surrounding pixel values are close to the true value, thus eliminating isolated noise points.**

**This is achieved by constructing a matrix range of size 3\*3** (B(i:(i+2), j:(j+2)))**, sorting the pixel values in that range by their size, generating a monotonically ascending (or descending) sequence of two-dimensional data, and calculating the mean (in the case of 3\*3, 9). This operation is sequentially delivered to all pixels of the image.**

**The essential codes are list below:**

for i = 1: 512-3+1

for j = 1 : 512-3+1

C = B(i:(i+2), j:(j+2));

%convert C into a list of vectors for calculation

C = C(:);

Cm = mean(C);

B(i+1, j+1) = Cm;

end

end



Figure 12-the filtered image

**PSNR = 22.3508 dB**

**(f)**

**To compare the performance of various denoising algorithms requires consideration of not only their ability to remove noise, but also their ability to preserve the detail of original the image. In practical terms, other elements such as running time and memory consumption should also be taken into consideration.**

**The mean filter algorithm does not perform well with salt and pepper noise because the means takes into account the extreme values of the pixels caused by the noise points, "diluting" the noise, passing the change in pixel value by the salt and pepper noise evenly to all pixels, resulting a blurred, grainy image.**

**The median filter algorithm could "delete" the noise because taking the median could possibly excludes the extreme values. For the median denoising algorithm, a larger filter will result in better denoising, but more image detail could be lost.**

**For the interpolation algorithms, the nn algorithm gives a sharp image boundary but a strong jagged stem; the bilinear algorithm is constrained by the content of the image itself.**

**In terms of performance, the bicubic algorithm is the best. It yields more detailed images and have good denoising performance, however, the bicubic algorithm takes longer and is more resource intensive.**

**The PSNR provides a somewhat comprehensive picture of performance.** **In the series of experiments conducted before, the bicubic algorithm obtained the highest PSNR for the images. This is because bicubic interpolation uses the most complex algorithm. Each time it uses two polynomials in a rectangular grid divided by the image to calculate a weighted average of sixteen sampled points to interpolate three points. The interpolated surface is smoother than the corresponding surface obtained by bilinear interpolation or nearest neighbors interpolation.**

****SUMMARY****

In this experiment students learned how to use MATLAB to read, display and generate a variety of images. Students mastered basic image processing technics including calculating PSNR, resizing and removing noise. Students gained insight into important concepts such as Gaussian white noise, salt and pepper noise, interpolation algorithms, average denoising and median denoising. Students also develop skills on coding and mathematical to progress in the field of image processing.

APPENDIX

**Images**

**Lenna512.bmp---------original image**

**I\_1.bmp-----------------down sampled image for task2(a)**

**I\_1\_1.bmp--------------up sampled image using nearest neighbor algorithm for task1(b)**

**I\_1\_2.bmp--------------up sampled image using bilinear algorithm for task2(c)**

**I\_1\_3.bmp--------------up sampled image using bicubic algorithm for task2(c)**

**im\_wn.bmp-------------image with Gaussian white noise added for task3(a)**

**im\_10.bmp--------------image from averaged 10 images for task3(b)**

**im\_100.bmp------------image from averaged 100 images for task3(b)**

**im\_1000.bmp-----------image from averaged 100 images for task3(b)**

**im\_sp.bmp--------------image with salt and pepper noise added for task3(c)**

**im\_median3.bmp------image with 3\*3 median filter applied for task3(d)**

**im\_median5.bmp------image with 5\*5 median filter applied for task3(d)**

**im\_m3si.bmp-----------image with self-implemented 3\*3 mean filter applied for task3(e)**

**Scripts**

**Scripts are provided with MATLAB functions corresponded to the tasks’ requirements.**