Faculty of Computer Science

Fall Semester 2023

CS486

Image Processing Documentation

Name	ID
Mariam Mamdouh	212219
Aly Ashraf	210267
Huda Fouad	210487
Farah Ashraf	213465

Image Processing Project

Average Filter: Average filter works by computing the average in a 2D matrix(the Image) then putting the Resulted value inside a new image. This is implemented in this project by getting a 3x3 kernel then averaging them by going around the pictures and taking the values in the kernel and putting it in the center of every pixel in order.

Median Filter: Median filter is implemented by by passing a kernel on an image then sorting the values,after that it gets the median value (the value in the middle of all values).This is implemented in this project by passing a 3x3 kernel on the image then saving the values in an np array,after that it is sorted in the array then it chooses the median value

Adaptive median filter: Adaptive median filter works by comparing each pixel in the image to the surrounding pixels. If the pixel value is significantly different from the surrounding pixels, then it is considered to be noise. The filter then replaces the noisy pixel with the median value of the surrounding pixels,it is implemented in this project by adding a range of S(size of kernel),smax(the maximum size) then. Adding a padded image (pad) as a np array with the size of the kernel,then the values of the image is added to a matrix to get flattened after that sorting was used to get Zmed,Zmax,Zmin.Furthermore,A1 is computed by doing zmed - zmin
And A2 is computed by doing Zmed-Zmax
After that an if condition of if a is greater than 0 and a2 is smaller than 0
If it satisfied we put b1 = to zxy(the value o the pixel) - zmin
And if b1 is greater than 0 and b2 is smaller than zero we put the values in the new computed image else the newimg value of that pixel will become the median.
However, if the a1>0 and a2<0 condition not satisfy the kernel then it will enter another condition if the st not equal smax then the st will reset back to 0 and s will increase by 1 incase that condition does not apply then the last condition will be that the img pixel will take the value of the zmed

Guassian filter: A Gaussian filter is a type of linear filter that is used to smooth images and remove noise. It works by convolving the image with a Gaussian kernel, which is a matrix of weights that is based on the Gaussian distribution. The Gaussian distribution is a bell-shaped curve that is centered

around the mean value of the image. The weights in the Gaussian kernel are highest at the center of the kernel and decrease as you move away from the center. This means that the pixels in the center of the image will have a greater influence on the output of the filter than the pixels at the edges.It was implemented in this project by taking a 3x3 kernel and putting the the following values in the kernel (1,2,1),(2,4,2),(1,2,1)then convoluting it onto the neighborhood after that it was divided by the summation of the previous filter then adding in to the center of the kernel.

Sharpening Spatial filters:

This filter is specifically created to emphasize fine details or improve the appearance of blurred details. It achieves this by using a mask with positive or negative weights. By applying the first derivative, we can extract more information from the image, and applying the second derivative enhances the sharpness by increasing the response to fine details.

1. *Laplacian Operator.*

   A spatial filter for detecting edges is the Laplacian operator. It calculates an image's second-order derivative and highlights areas with unexpected shifts in intensity. It is commonly used to detect edges and fine details in an image. For edge enhancement, the Laplacian operator can be applied directly to the image or combined with other filters.

   La plician equation without scaling = $g(x,y) = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$

   La plician equation with scaling = $g(x,y) = \left(\frac{\left((g(x,y)) - \min g(x,y)\right)}{(\max g(x,y) - \min g(x,y))} \right) * 255$

   Firstly, we will copy the original image to the center of the new image then create an output image with zeros of the same dimensions as the original image then iterate over the pixel of the new image but with excluding the padd region then calculate the la plician equation as mentioned before then assign the Laplacian value to the corresponding pixel in the output image

2. *Unsharp Masking and Highboost Filtering.*

   Unsharp masking is an image sharpening technique in which the original image initially becomes blurred, and then the blurred version is subtracted from the original. The edges and details in the image are improved by this subtraction.

   Highboost filtering is a variant of unsharp masking that allows for control over the amount of sharpening. It is achieved by combining the original image with a weighted version of the high-frequency components obtained from the unsharp masking operation.

   When weighted version of the high-frequency components which is constant in the project which is k = 1 the Unsharp masking is done, and when k > 1 the Highboost filtering is done.

   $g_{mask}(x,y) = f(x,y) - \overline{f}(x,y)$

$$g(x,y) = f(x,y) + g_{mask}(x,y)$$

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Both of the them work on the same code but only the different the value of the constant value first blurs the image using average filtering, then subtracts the blurred image from the original image to create an unsharp mask then add the unsharp mask to the output image multiply by the constant number.

3. *Roberts Cross-Gradient Operators*

Edge detection filters called Roberts Cross-Gradient operators determine the gradient's strength and direction at every single pixel in an image. These operators compute the intensity differences between neighboring pixels in both vertical and horizontal directions using a 2x2 kernel. The gradient magnitude and direction can be used to identify edges in the image.

$M(x,y) \cong |G_X| + |G_Y| \cong |Z9 - Z5| + |Z8 - Z6|$

Firstly, we copy the original image to the center of the new image by assigning the corresponding pixel values then iterate over the pixels of the new image excluding the padding region and for each pixel, calculate the differences between the current pixel and its diagonal neighbors in both the x (horizontal) and y (vertical) directions. These differences represent the cross-gradients. And it represents three outputs the first apply the filter on x direction and the second output apply the filter on y direction and the last output represents the sum of the x and y direction.

4. *Sobel Operators.*

Several gradient-based edge detection filters use Sobel operators. They consist of two 3x3 convolution kernels, one for estimating the gradient along the horizontal axis and the other along the vertical axis. To detect edges in the image, the Sobel operators compute the gradient magnitude and direction at each pixel. They are especially effective in detecting edges with different directions.

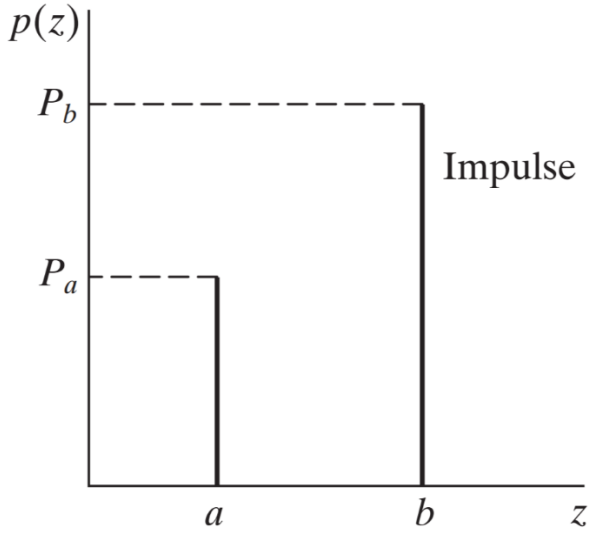
$$M(x,y) \cong |G_X| + |G_Y|$$
$$\cong (|Z7 + 2 * Z8 + Z3 - Z1 + 2 * Z2 + Z3|)$$
$$+ (|Z3 + 2 * Z6 + Z9 - Z1 + 2 * Z4 + Z7|)$$

Firstly, we copy the original image to the center of the new image by assigning the corresponding pixel values then iterate over the pixels of the new image excluding the padding region and for each pixel, convolve the surrounding 3x3 pixel neighborhood with the Sobel kernels which is horizontal kerel is [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]] and the vertical kernel is [[-1, -2, -1], [0, 0, 0], [1, 2, 1]]. Then compute the horizontal gradient by summing the products of the kernel and pixel values in the x direction. Also compute the vertical gradient by summing the products of the kernel and pixel values in the y direction. And it represents three outputs the first apply the filter on horizontal gradient and the second output apply the filter on vertical gradient and the last output represents the combined gradient information.

Noise Filters

➢ *Impulse noise (salt and pepper)*

Salt-and-pepper noise, also known as impulse noise, is a form of noise sometimes seen on digital images. This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels. This type of noise has the following distribution.



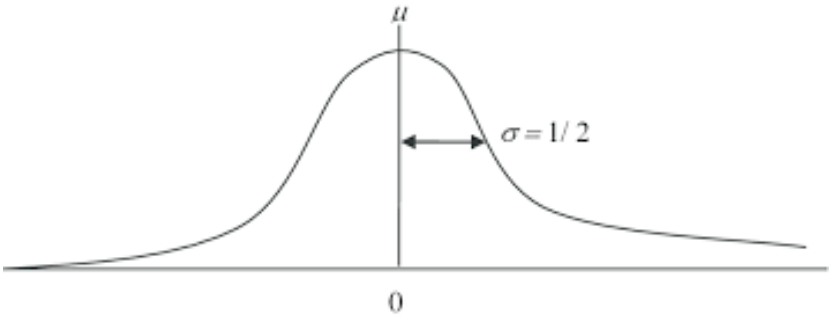
The following steps are the followed steps to create Impulse noise in the image.

i. A variable holding the image's path.
ii. A variable holding the probability of the pepper noise, and another for salt noise.
iii. Two for loops then loop over the rows and columns of the image to assign new intensities for the pixels.
iv. A variable holding a random number generator is created inside the loop to create a random number each round.
v. A comparison technique is used for assigning a zero intensity for the pixel in case the random number generated is less than the pepper's probability.
vi. Another one is used to assign a 255 intensity for the pixel in case the random number generated is more than the salt's probability.
vii. If not both cases then the pixel's intensity stays as it is.
viii. Then the original image and the noisy image are both displayed.

- ***Gaussian noise***

Gaussian noise is a statistical noise with a Gaussian (normal) distribution. It means that the noise values are distributed in a normal gaussian way. This algorithm depends on the mean, standard deviation and normal distribution of the original image.

This type of noise has the following distribution.



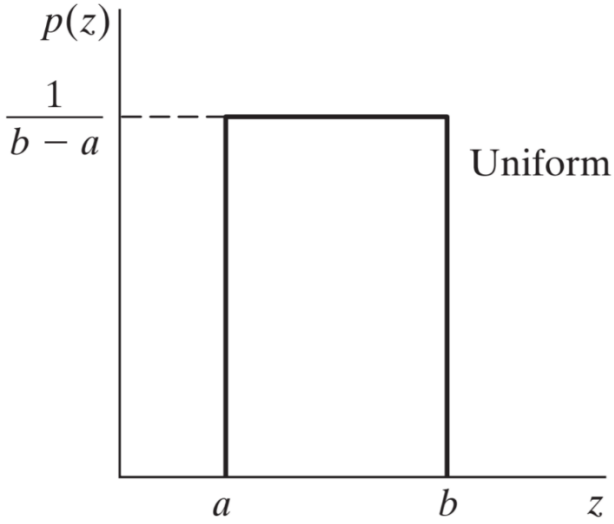
The following steps are the followed steps to create Gaussian noise in the image.

i. A variable holding the image's path.
ii. Two variables, one to hold global mean, and the other for global standard deviation.
iii. Two for loops to calculate the global mean, another two to calculate the global standard deviation.
iv. A variable holding the integer rounded value of the images size over the kernel's size.
v. A variable holding a normal value randomly selected based on the global mean, global standard deviation, and the new size.
vi. Two for loops to fill in the noise list values into the center of kernels.
vii. Then both original and noisy image are displayed.

- ***Uniform noise***

The noise caused by quantizing the pixels of a sensed image to a number of discrete levels is known as quantization noise. It has an approximately uniform distribution. In simple words, the noise is equally distributed or the intensity values of noise have equal probabilities.

This type of noise has the following distribution.



The following steps are the followed steps to create Uniform noise in the image.

i. A variable holding the image's path.
i. A list created to hold kernel values.
ii. Four for loops then loop over first the rows and columns of the image, second along a kernel size 3 x 3 to collect the kernel values.
iii. Then the kernel's pixels' intensities are appended into the list.
iv. Then the numpy function of minimum value and maximum value in the list capture both values into two variables.
v. Then a mathematical function takes place. It is the difference of max and min values added over the min value then multiplied by a random value, generated by a random number generator, all added to the kernel's center intensity.
vi. Then the list is cleared to hold the new kernels' intensities.
vii. Then the original image and the noisy image are both displayed.

Noise healing filters

Noise healing filters are filters responsible for noise healing and getting the optimal outcome. In simple words, a denoising technique. A few filters were applied to the previously mentioned noise types.

They are the following:

i. *Arithmetic mean or averaging filter*: it's the summation of a kernel's intensities over the kernel size. Then the value captured is assigned as the kernel's center intensity. Repeated along the image with a constant kernel size.

ii. *Geometric mean filter*: it's the sum of multiplication of a kernel's intensities power 1/kernel size. Then the value captured is assigned as the kernel's center intensity. Repeated along the image with a constant kernel size.

iii. *Median filter*: it's a list holding the kernel's intensities. Then sorted and the middle value in the kernel's size is assigned as the kernel's center intensity. Repeated along the image with a constant kernel size.

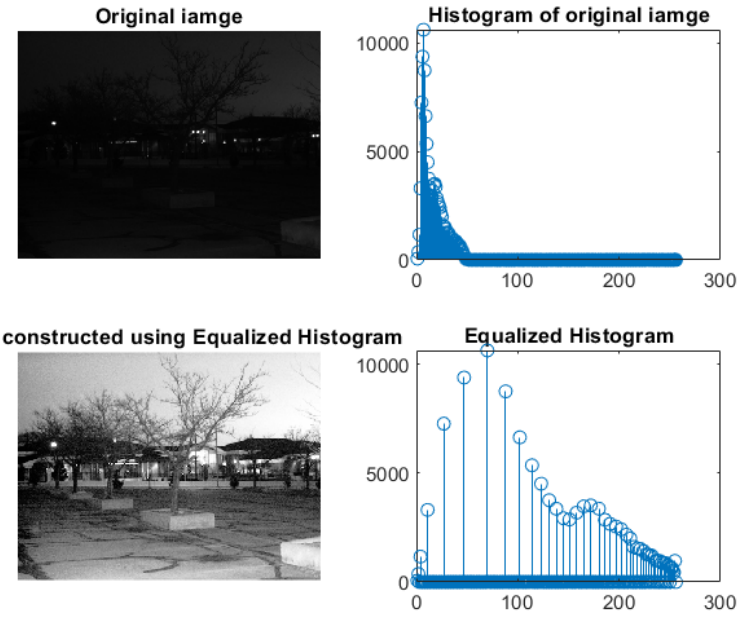
iv. *Midpoint filter*: it's a list holding the kernel's intensities. Then the minimum and maximum values in the kernel are captured. Then the sum of the min and max values is multiplied by ½. Then the final value is assigned as the kernel's center intensity. Repeated along the image with a constant kernel size.

v. *Harmonic mean filter*: it's the sum of 1/the kernel's intensities. Then the kernel's size is divided by the sum. The captured value is assigned as the kernel's center intensity. Repeated along the image with a constant kernel size.

NOTE: All algorithms are implemented through for loops for kernels along the image with all capturing lists or variables restarting with every new kernel to capture intensities of the kernel only not the whole image.

➢ *Histogram Equalization*

Definition: Histogram equalization is a computer image processing technique used to improve contrast in images. Through effectively spreading out the most frequent intensity values, also known as stretching out the intensity range of the image.



How is it calculated?

i. The frequency of the image's intensities is captured.

ii. Then each frequency is divided by the total number of pixels (also know as image size), to get the intensities' probabilities.

iii. Then the probabilities are normalized through multiplying the intensity range-1 by the sum of the current intensity probability and the previous probabilities. Note: integers only.

S_0 = (256-1(probability [0]))

S_1 = (256-1(probability [0] + probability [1]))

.

.

.

S_n = (256-1 (probability [0] + probability[1] +….. + probability[n]))

iv. Then the normalized values are assigned back to their original intensity positions.

➢ *Histogram Specification*

Histogram Specification is a generalized version of histogram equalization, a standard image processing operation. An equalized image has an equal number of pixels at all brightness levels, resulting in a straight horizontal line on the histogram graph. Simply it's the process of mapping two normalized intensity values of two images into one image. The process of transforming one image so that the cumulative distribution function (CDF) of values in each band matches the CDF of bands in another image.

How is it calculated?

i. The histogram equalization process is applied to both images.

Then the first image intensity values are mapped to the second image intensity values.

Interpolation

Interpolation is the process of using known data to estimate unknown values. It is a method of image resizing and resampling that involves determining the intensity values of new pixels in a resized image.

1-Nearest Neighbor Interpolation

This interpolation method takes the closest pixel in the original image and assigns the intensity of that pixel to the new pixel.
It is a method of image resizing and resampling that involves determining the intensity values of new pixels in a resized image based on the values of the nearest pixels in the original image.
In this method, when a new pixel needs to be inserted into the resized image, the value of the closest pixel in the original image is used. This means that the new pixel is assigned the value of the nearest pixel in the original image, without any interpolation between the surrounding pixels.

2-Bilinear Interpolation

This interpolation uses the four nearest neighbors to estimate the intensity at a given location.
Bilinear interpolation is a method of image resizing and resampling that involves determining the intensity values of new pixels in a resized image by taking a weighted average of the closest four pixels in the original image.
In this method, when a new pixel needs to be inserted into the resized image, the value of the pixel is determined by considering the values of the four closest pixels in the original image. The weights used for each pixel are proportional to the distance between the new pixel and the four closest pixels.

$$f(x,y) = (1-a)(1-b)\, g(l,k) + a(1-b)\, g(l+1,k) + b(1-a)\, g(l,k+1) + ab\, g(l+1,k+1)$$

$$l = ceil(x) \qquad a = x - l$$

$$k = ceil(y) \qquad b = y - k$$