Image Processing Final Project

(Description)

Team Members:

|  |  |
| --- | --- |
| Member | ID |
| Sherif Mohammed | 213485 |
| Mohamed Emad | 213411 |
| Yousef Amr | 212473 |

General Description:

This is an image processing project that aims to apply filters on images using different algorithms.

Gui:

a simple Gui Using (Tkinter) the user interface consist of some buttons one is for selecting the input image and each of the other buttons represents a filter (an algorithm) to be applied on that image.

Filters:

1. Smoothing Spatial filters

* Median Filter
* Adaptive Filter (Max)
* Averaging filter
* Gaussian filter

Using This Rule here we generated the 5x5 kernel for each cell in the kernel.

A picture containing font, handwriting, white, calligraphy

Description automatically generated

Then multiplying each 5X5 window in the original image by the generated kernel

1. Sharpening Spatial filters:

* Laplacian Operator

We tried 3x3 Laplacian operator but we found that the result from using 5X5 kernel is better than 3x3 also used 5x5 Laplacian operator

Laplacian operator= [ 0 , 0 , -1 , 0 , 0 ]

[ 0 , -1 , -2 , -1 , 0 ]

[ -1 , -2 , 16 , -2 , -1]

[ 0 , -1 , -2 , -1 , 0 ]

[ 0 , 0 , -1 , 0 , 0 ]

* Unsharp Masking and Highboost Filtering

According to this Equation :

A picture containing font, text, typography, calligraphy

Description automatically generated

we got the blurred image first by applying gaussian filter because it produced better result than the other smoothing filter then we calculated the mask form subtracting the blurred image from the input image then applying the equation and the user can enter the value of k to choose between highboost filter or unshrp masking.

* Roberts Cross-Gradient Operators
* Sobel Operators

1. Noise filters:

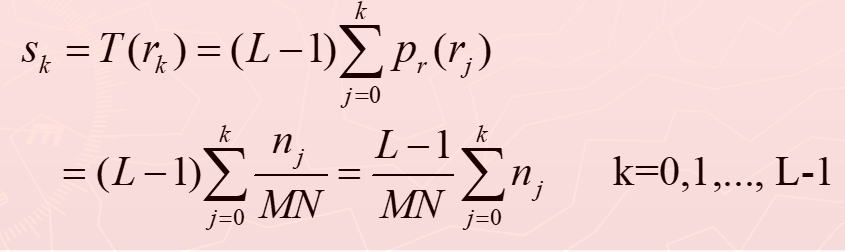
* Impulse noise (salt and pepper)
* Gaussian noise
* Uniform noise

1. Transform /Frequency Domain Filters

* Histogram Equalization

Calculate the recurrence of each intensity value the the probability by multiplying it by mxn then we used this rule

Note :Include drawing



And round the result then mapping the new intensity values

* Histogram Specification

Using the same rule in equalization but additionally we calculate the same thing for the image used for matching the we mapped the new intensity values

Note :Include drawing

* Fourier transform (forward and Inverse)

In Forward Fourier transform we used this equation

A black text on a white background

Description automatically generated with low confidence

But there was a problem with the calculation time because of the imaginary values so we used vectors to speed it up and also, we used multi-threading to apply the equation on different parts on the original image at the same time and then we shifted the low frequency components to the center to get shifted Fourier spectrum.

In Inverse Fourier Transform We used This equation

A picture containing font, text, white, line

Description automatically generated

But the result of it is not 100% Accurate even we got the input image (which is the result of forward transform) in its imaginary values and then undo the shifting we done in the forward transform but it looks reasonably like the input image

* Interpolation (nearest neighbor)

The user should enter a scale factor for resizing the photo the we Calculate the new size of the output image based on the scale factor the user provided and loop on the output image and get the value of the output image from the input image [current pos of output image \*scale factor ] for both x and y coordinates.