Digital Image Processing

Final assignment and course assessment Report

|  |  |
| --- | --- |
| ID NO | Name |
| 27 | Mohanto Tushar chandra |

***1. Application Design by Matlab***

(1**) Histogram and achieve histogram equalization** :The relevance and impact of probability distributions on image processing are the subject of this study.It may be characterized as a probability distribution function of brightness for a certain area, which might be a whole picture. To generate a histogram, the probability density function of the brightness is frequently calculated by counting how many times each brightness occurs in the picture region. The brightness average is defined as the sample mean of the brightness of pixels in a certain region. The frequency is shown by the histogram. The histogram has a wide range of uses in image processing. It could, for starters, be used for picture analysis. Second, the functions of an image's brightness and contrast, as well as the final two uses of equalizing and thresholding. Normalizing a histogram is one technique to convert the intensities of discrete distributions to the probability of discrete distribution functions. The technique to equalize the histogram is to control the image's contrast by altering their intensity distribution functions. The major goal of this procedure is to give the cumulative probability function a linear trend (CDF).A method of segmentation is to divide a section of the picture into constituent areas or objects.

(2) **Use swarm optimization and cuckoo search algorithm to achieve gray image contrast enhancement:**

we report on the investigation of two different metaheuristic based algorithms for Gray Image (GI) enhancement. First, we investigated the Particle Swarm Optimization (PSO) algorithm under certain parameter settings for the GI enhancement task, and followed with the Cuckoo Search (CS) algorithm for the same task. Then, we proposed an algo-rithmic procedure for computing a new set of objective measures for quantifying the performance of any image enhancement algorithm. Comparative analyses were conducted alongside classical approaches such as the Linear Contrast Stretching (LCS) and the Histogram Equalization (HS) techniques.

**Use of Particle Swarm Optimization (PSO) Algorithm**

We present here the use of Particle Swarm Optimization (PSO) algorithm [14] as modified for the enhancement pro- cess. First, we state the necessary functions used in PSO as

vn = wnvn + cnr1(gbest − xn) + cnr2(pbest(n) − xn)

t+1 t 1 t 2

xn = xn + vn

t+1 t t+1

where xn is the position of the nth particle, vn denotes the next velocity of an nth particle, wn is the inertial weighting, r1 and r2 are randomly generated numbers withinthe range 0 and 1, and cn and cn are the social and cognitive components of the kernel, while gbest and pbest are the global and personal best values of the entire particle popula- tion,and individual particle, respectively. Next, a population size, P , is set for the possible number of solutions (or particles) to be used by the algorithm. The dimensions of each particle d, is given as the number of parameters to be optimized (d = 4, in this case). Next, the initial random values were generated for a, b, c, and κ, and these values were used in (1) to transform the image Gi,j into Fi,j, while evaluating the fitness of Fi,j using (5) based on the values of a, b, c, and κ. At an initial time t, the fitness of each particle, n is stored as pbest (for n = 1, 2, . . . , P ), while the global best value in the population of all particles is stored as gbest. The subsequent steps taken by the algorithm are as follows:

1) For each particle, n = 1 to P, do

2) Compute the fitness value of each particle, n using (5), after transformation using (1)

3) Compare the pbest(t) and pbest(t+1), and do If pbest(t+1) > pbest(t)

Then, pbest(t+1) is made the current best value of the particle.

4) Return to 1, and do for all P

5) Obtain the gbest at t+1

6) If gbest(t+1) > gbest(t)

7) Then, gbest(t+1) is made the current global at t+1.

8) Thus, compute the next value of the velocity and the particles using (6)

9) Return to 1, until P.

10) The use of Cuckoo Search (CS) Algorithm

#Swarm Optimization Matlab code

%Rosenbrook Function is (1-x)^2 + 100(y-x^2)^2

%population initialisation  
global pop gbestfitness w c1 c2  
pop = 100; %population  
gbestfitness = 50;

w = 1;  
c1 = 1;  
c2 = 1;

for i = 1:pop  
pbestfitness(i) = 100;  
end

clc;  
position = rand(pop,2);  
velocity = rand(pop,2);

for k = 1:100

for i = 1:pop  
fitness(i) = (1 - position(i,1))^2 + 100\*(position(i,2) - position(i,1)^2)^2 ; %fitness function  
end

for i = 1:pop  
if fitness(i) < pbestfitness(i)  
pbestfitness(i) = fitness(i);  
pbest(i,:) = position(i,:);  
end  
  
if fitness(i) < gbestfitness  
gbestfitness = fitness(i);  
gbest = position(i,:);  
end  
end

for i = 1:pop  
velocity(i,:) = w \* velocity(i,:) + (c1 \* rand \* (pbest(i,:) - position(i,:))) + (c2 \* rand \* (gbest - position(i,:))); %velocity update  
position(i,:) = position(i) + velocity(i); %position update  
end

end

disp(gbest);

**The Cuckoo Search (CS) Algorithm using Levy flight**

[15] is applied here for image enhancement and the kernel function used for finding new solutions is given as: n t=1= xn + α ⊕ Levy(λ) (7)

11) Increment BE set by 1

12) End

where α > 0 is the step size related to the scale of the problem of interest, in most cases, α = 1 is normally used, and 1 < λ < 3 is the Levy distribution parameter, while xn and xn are the current and next solutions with dimension,d. In our work, the CS algorithm was used as follows:

1)Let the number of nests (or different solutions, similar to particles in PSO) be n, and the dimension of each particle be d (where d = 4). Let the probability of discovering an alien egg (or solution) in a nest be paThus, for a number of iteration, Niter, the rest of the process ensues as follows:

2)Set the lower and upper bounds for the parameter constraints based on dimension, d,

3)Obtain the random initial solutions (or nests),

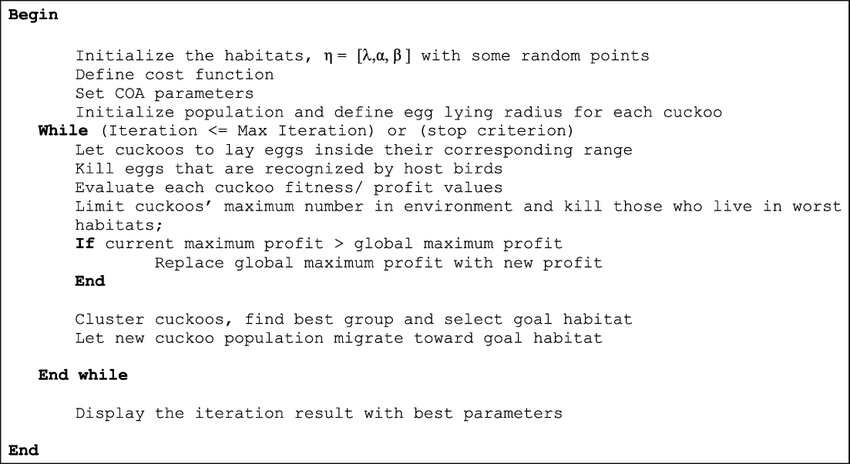
4)For each iteration, until Niter, do 5) Get a cuckoo randomly by Levy Flights

t+1 t 1 t 2

xn = xn + vn

6) Evaluate each solution (or nest) using (5), after trans- 7) Obtain the global best value among all nest as Maxfit (8) If Maxfit(t+1) > Maxfit(t) ( 9) Update the new global best (10) End if (11) Empty a fraction, pa, of the worst nests, (12) Update the new nests using (7) (13) Keep the best solutions (14) Return to 1, until Niter is completed.

Summarily, the optimum values of a, b, c, and κ, computed by the CS algorithm will typically produce the best enhanced image at the end of the iteration.

pseudu code for cuckoo optimization algorithm

**We propose here an algorithmic procedure for measur**- ing the performance of a GI enhancement algorithm. The procedure is as follows: Let the original and enhanced gray image be Gi,j and Fi,j, respectively, for i R and j C. Then, the algorithm uses a (3 3) window size to compute the local variance, σG and σF , of both Gi,j and Fi,j , respectively. Next, it uses Otsu’s algorithm to compute an optimum threshold value, TG from σG. Finally, to compute the measurement metrics, let the count of the Detailed and Background Variance of both the original and enhanced image be denoted as DO and BO, and DE and BE, respectively. The algorithm computes these metrics as follows:

1) For i = 1 to R, do

2) For j = 1 to C, do

3) If σG(i, j) TG

4) Increment DO set by 1

5) Else

6) IncrementBO set by 1

7) End

8) If σF (i, j) TG

9) Increment DE set by 1

10) Else

11) Increment BE set by 1

12) End

13) End

14) End

At the end of Line 14, the algorithm computes the overall Detailed and Background Variance of both the original and enhanced image by adding all the counts in DO and BO, and DE and BE, respectively. In addition, by using a Sobel detector, the number of edges denoted as NO and NE respectively for the original and enhanced image are also considered for evaluating the algorithm’s performance.

3) **Add ‘Gaussian’ and ‘Pepper&Salt’ noise to the images**:

Salt-and-pepper noise is a sparsely occurring white and black pixels sometimes seen on images. Median filter or a morphological filter methods considered as a common reduction methods of this type noise of noise [4, 5].

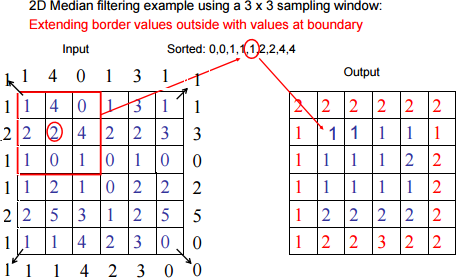
Image noise may be defined as any change in the image signal, caused by external disturbance. Digital images are often corrupted by impulse noise also known as Salt and Pepper noise due to transmission errors [10]. Accordingly it is important to detect noisy pixels (using estimated calculations) and recover an efficient value for each, which known as image filters [6, 9]. The most commonly used filters are the Standard Median Filter (SMF), Adaptive Median Filter (AMF) [5], Decision Based Algorithm (DBA) [8], Progressive Switching Median Filter (PSMF) [12], and Detail preserving filter (DPF) [13]. The filtering algorithm varies from one algorithm to another by the approximation accuracy for the noisy pixel from its surrounding pixels [6, 9]. Median Filter (MF) is used widely because of its effective noise suppression capability [7]. Despite its main disadvantages of modifying both noisy and non-noisy pixels thus removing some fine details of the image. Image de-noising is the process of finding and recover unusual values in digital image, that represents unwanted information which spoils image quality.

The Salt & Pepper noise is generally caused by defect of camera sensor, software failure, or hardware failure in image capturing or transmission. Due to this situation, Salt & Pepper noise model, only a proportion of all the image pixels are corrupted whereas other pixels are non-noisy [12]. A standard Salt & Pepper noise value may be either minimum (0) or maximum (255). The typical intensity value for pepper noise is close to 0 and for salt noise is close to 255. Furthermore, the unaffected pixels remain unchanged.



Is a nonlinear method widely used to remove ‘salt and pepper’ type noise. The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels within predefined window size. The median is calculated by first sorting all the pixel values from the window, and then replacing the pixel being considered with the middle (median) pixel value. Then the window slides, pixel by pixel over the entire image.

The following (Figure 2) example shows the application of a median filter to a simple 2 dimensional signal. A window size of 3x3 is used, with one entry immediately preceding and following each entry.



# Salt and Pepper Noise Effects

First part of this paper we will introduce an experimental method to investigate the affects a Salt and Pepper noise affecting RGB color image. The proposed method can implemented applying the following steps:

Acquire the source RGB image.

1.Add Salt and Pepper noise to the image varying the noise ratio.

2.Construct a RGB columns matrix by reshaping the color image.

3.Find the unique\_colors combination by applying the Matlab function unique.

4.Construct an array with accumulation to find the repetition of each color combination by applying the Matlab function accumarray.

5.Find the color combination with maximum repetition.

6.Find the color combination with minimum repetition.

7.Detect the locations of the pixels with maximum repletion.

8.Detect the locations of the pixels with minimum repletion.

## 4) **Use ‘prewitt’ and ‘sobel’ filters to extract image edges**:

## **Prewitt Operator:** The Prewitt operator was developed by Judith M. S. Prewitt. Prewitt operator is used for edge detection in an image. Prewitt operator detects both types of edges, these are:

* Horizontal edges or along the x-axis,
* Vertical Edges or along the y-axis.

Wherever there is a sudden change in pixel intensities, an edge is detected by the mask. Since the edge is defined as the change in pixel intensities, it can be calculated by using differentiation. Prewitt mask is a first-order derivate mask. In the graph representation of Prewitt-mask’s result, the edge is represented by the local maxima or local minima.

Both the first and second derivative masks follow these three properties:

* More weight means more edge detection.
* The opposite sign should be present in the mask. (+ and -)
* The Sum of the mask values must be equal to zero.

Prewitt operator provides us two masks one for detecting edges in the horizontal direction and another for detecting edges in a vertical direction.

*Prewitt Operator [X-axis] = [ -1 0 1; -1 0 1; -1 0 1]*

*Prewitt Operator [Y-axis] = [-1 -1 -1; 0 0 0; 1 1 1]*

*Steps:*

* Read the image.
* Convert into grayscale if it is colored.
* Convert into the double format.
* Define the mask or filter.
* Detect the edges along X-axis.
* Detect the edges along Y-axis.
* Combine the edges detected along the X and Y axes.
* Display all the images.

% MATLAB code for prewitt

% operator edge detection

k=imread("logo.png");

k=rgb2gray(k);

k1=double(k);

p\_msk=[-1 0 1; -1 0 1; -1 0 1];

kx=conv2(k1, p\_msk, 'same');

ky=conv2(k1, p\_msk', 'same');

ked=sqrt(kx.^2 + ky.^2);

% display the images.

imtool(k,[]);

% display the edge detection along x-axis.

imtool(abs(kx), []);

% display the edge detection along y-axis.

imtool(abs(ky),[]);

% display the full edge detection.

imtool(abs(ked),[]);

**

## **Sobel Operator**

It is named after Irwin Sobel and Gary Feldman. Like the Prewitt operator [Sobel operator](https://www.geeksforgeeks.org/image-edge-detection-operators-in-digital-image-processing/) is also used to detect two kinds of edges in an image:

* Vertical direction
* Horizontal direction

The difference between Sobel and Prewitt Operator is that in Sobel operator the coefficients of masks are adjustable according to our requirement provided they follow all properties of derivative masks.

*Sobel-X Operator = [-1 0 1; -2 0 2; -1 0 1]*

*Sobel-Y Operator = [-1 -2 -1; 0 0 0; 1 2 1]*

Source Code:

% MATLAB code for Sobel operator

% edge detection

 k=imread("logo.png");

 k=rgb2gray(k);

 k1=double(k);

 s\_msk=[-1 0 1; -2 0 2; -1 0 1];

 kx=conv2(k1, s\_msk, 'same');

 ky=conv2(k1, s\_msk', 'same');

 ked=sqrt(kx.^2 + ky.^2);

 %display the images.

 imtool(k,[]);

 %display the edge detection along x-axis.

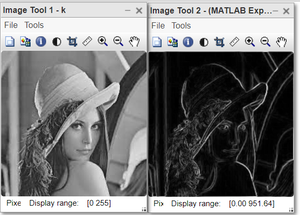
 imtool(abs(kx), []);

 %display the edge detection along y-axis.

 imtool(abs(ky),[]);

 %display the full edge detection.

 imtool(abs(ked),[]);



**2.Whole Course Report is in another page.**