Triple Threshold Statistical Detection filter for removing high density random-valued impulse noise in images using Python

MID MINOR SYNOPSIS

UNDER THE GUIDANCE OF: Dr. Archana Pandey

SUBMITTED BY: Samarth Agarwal (17102200)

Ananya Agarwal (17102250)

Certificate

This is to certify that the work titles **"Triple Threshold Statistical Detection filter for removing high density random-valued impulse noise in images using Python"** submitted by **SAMARTH AGARWAL (17102200) AND ANANYA AGARWAL (17102250)** in partial fulfillment for the award of degree of Bachelors of Technology of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of the Supervisor:.................

Name of Supervisor: Dr. Archana Pandey

Designation:ASSISTANT PROFESSOR (SENIOR GRADE)

Date: ....

Acknowledgement

We would like to acknowledge our parents and all the teachers, who supported us both morally and technically, especially our supervisor Dr. Archana Pandey who helped us at every step in the making of our project, helped us by clarifying our queries related to our project and technical problems.

Also, our special thanks to all class fellows who helped us in clarification of any issue as well as implementation and in documentation.

SAMARTH AGARWAL (17102200)

Signature of Student:..........................

ANANYA AGARWAL (17102250)

Signature of Student:..........................

ABSTRACT

In this project we have implemented a noise detection algorithm which detects noisy pixels in images corrupted by random-valued impulse noise of high levels up to 80% noise density,we have used an algorithm for detecting random-valued impulse noise (RVIN) in pictures.In random-valued impulse noise, noisy pixels are randomly present between 0 and 255, and so, it is difficult to detect the noise and restore the image.Three levels of adaptive thresholds, have been used so as to eliminate the misdetection of noise-free pixels as noisy pixels and vice versa. A noise signature has been calculated for each pixel of the image and compared with the first threshold to identify noise and then the central pixel is compared with the second and third levels of thresholds.

ABOUT PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.

The entire project has been executed on Google Colaboratory.

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

ALGORITHM USED

The algorithm for detecting RVIN is as follows:

Step I: Take a 5×5 window A. Then, calculate mean (μ A) and standard deviation (σ A) of all pixels of matrix A except the Central pixel.

Step II: Calculate p ij as absolute differences of μ A with all pixels of matrix A except CP and obtain 24 such values.

Step III: Calculate μ p and σ p of all the above values (i.e., p ij) and define first threshold T1 as

$T1=\mu p+\sigma p$

Step IV: Now, calculate q ij as absolute difference of CP with rest of all pixels of A and obtain 24 values.

Step V: Calculate μ q of all the above values (i.e., q ij) and define the NS as

$NS=\mu q$

Step VI: Now, check if $NS \geq T1$ and $(0 \leq CP \leq m)$ or $(255 - m \leq CP \leq 255)$, then the CP is noisy.

Step VII: But, if $NS < T1$, define second level of thresholds

$T2max=\mu A+0.5\times\sigma A$

Step VIII: Now check if $(CP \leq T2\ min$ or $CP \geq T2max)$

and $(0 \leq CP \leq m)$ or $(255 - m \leq CP \leq 255)$, then the CP is noisy.

Step IX: If both conditions are not satisfied, define third thresholds

$T3max=Q3$

where Q1 and Q3 are the first and third quartiles of the set of all pixels of A except CP.

Step X: Now, check if $(CP \leq T3\ min$ or $CP \geq T3max)$

and $(0 \leq CP \leq m)$ or $(255 - m \leq CP \leq 255)$, then the CP is noisy.

Otherwise, the CP is noise-free.

PYTHON IMPLEMENTATION

DETECTION:

```
from google.colab import drive
drive.mount('/content/drive')

#Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
import math
from google.colab.patches import cv2_imshow
from skimage.util import random_noise
from PIL import Image
import random
import glob
from skimage.measure import compare_ssim,compare_mse,compare_psnr

mse=[]
psnr=[]
ssim=[]
images=[]
img_size=[]
img_shape=[]
noise_free_pixels=[]
li=[]
count=0

#Loading Image
for img in glob.glob("/content/drive/My Drive/Minor_6th_sem/images/Copy of IMG0025.bmp"):
  noise_free_pixels.clear
  li.clear
  count+=1
  print("count:",count)
  inp_img = cv2.imread(img)
  gray_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)
  img_size.append(gray_img.size)
```

```
    img_shape.append(gray_img.shape)


#Adding Noise
    noise_img = gray_img.copy()
    noise = random_noise(noise_img, mode='s&p', amount=0.5)
    noise = (255*noise).astype(np.uint8)
    m=4;
    for x in range(0,noise.shape[0]-1):
      for y in range(0, noise.shape[1]-1):
       if(noise[x][y]==0):
        noise[x][y]=random.randint(0,m)
       elif(noise[x][y]==255):
        noise[x][y]=random.randint(255-m,255)
    noise_img = Image.fromarray(noise)
    noise_img = np.asarray(noise_img)
    print("Noise added")

    #Padding
    #edge_padding for 3 extra edges
    padded_img=np.pad(noise_img,3,mode='edge')
    print("Padding done")

    #Detection
    noisy_pixels = padded_img.copy()
    #LEVEL 1:
    no_of_noisy_pixels=0;
    no_of_noise_free_pixels=0;
    i=0;
    for x in range(3,padded_img.shape[0]-3):
      for y in range(3, padded_img.shape[1]-3):
       #print('pixel',i);
       i+=1;
       cp=padded_img[x][y];
       #Step 1:
       mean_a=0;
       standard_deviation_a=0;
       for m in range(x-2,x+3):
        for n in range(y-2,y+3):
         if(((x==m) and (y==n))):
          mean_a=mean_a+0;
         else:
          mean_a=mean_a+padded_img[m,n];
```

```
  mean_a=mean_a/24;
  for m in range(x-2,x+3):
   for n in range(y-2,y+3):
    if(((x==m) and (y==n))):
     standard_deviation_a=standard_deviation_a+0;
    else:
     standard_deviation_a=standard_deviation_a+pow((padded_img[m,n]-mean_a),2);
  standard_deviation_a=standard_deviation_a/24;
  standard_deviation_a=math.sqrt(standard_deviation_a);

#**************************************************************************************
*******#

  #Step 2 and 3:
  mean_p=0;
  standard_deviation_p=0;
  for m in range(x-2,x+3):
   for n in range(y-2,y+3):
    if(((x==m) and (y==n))):
     mean_p=mean_p+0;
    else:
     mean_p=mean_p+abs((mean_a-padded_img[m,n]));
  mean_p=mean_p/24;
  for m in range(x-2,x+3):
   for n in range(y-2,y+3):
    if(((x==m) and (y==n))):
     standard_deviation_p=standard_deviation_p+0;
    else:
     standard_deviation_p=standard_deviation_p+pow((padded_img[m,n]-mean_p),2);
  standard_deviation_p=standard_deviation_p/24;
  standard_deviation_p=math.sqrt(standard_deviation_p);
  T1=mean_p+standard_deviation_p;

#**************************************************************************************
************#

  #Step 4 and 5:
  mean_q=0;
  for m in range(x-2,x+3):
   for n in range(y-2,y+3):
    if(((x==m) and (y==n))):
```

```
            mean_q=mean_q+0;

          else:
            mean_q=mean_q+abs((cp-padded_img[m,n]));
        mean_q=mean_q/24;
        NS=mean_q;

#****************************************************************************************************#

        #Step 6:

        if((NS>T1) and ((cp<=m) or (cp>=255-m))):
          #print(cp,'is noisy');
          noisy_pixels[x][y]=0;
          no_of_noisy_pixels+=1;

    #Level 2:

        #Step 7:
        else:
          T2_min=mean_a-(0.5*standard_deviation_a);
          T2_max=mean_a+(0.5*standard_deviation_a);

        #Step 8:
          if((cp<=T2_min or cp>=T2_max) and (cp<=m or cp>=255-m)):
            #print(cp,'is noisy');
            noisy_pixels[x][y]=0;
            no_of_noisy_pixels+=1;
        #Step 9:
          else:

    #Level 3:

            for m in range(x-2,x+3):
              for n in range(y-2,y+3):
                if((x==m) and (y==n)):
                  continue;
                else:
                  li.append(padded_img[m][n]);
            li.sort();
            N=24;
            Q1=int((N+1)/4);
```

```
        Q3=int((3*(N+1))/4);

        T3_min=li[Q1];
        T3_max=li[Q3];
        #Step 10:
        if((cp<=T3_min or cp>=T3_max) and (cp<=m or cp>=255-m)):
          #print(cp,'is noisy');
          noisy_pixels[x][y]=0;
          no_of_noisy_pixels+=1;
        else:
          #print(cp,'is noise-free');
          no_of_noise_free_pixels+=1;

print("Detection done")
print("no_of_noisy_pixels :",no_of_noisy_pixels)
print("no_of_noise_free_pixels :",no_of_noise_free_pixels)
```

Results (Detection)

```
[54]
pixel 0
4 is noisy
pixel 1
251 is noisy
pixel 2
254 is noisy
pixel 3
252 is noise-free
pixel 4
254 is noisy
pixel 5
253 is noise-free
pixel 6
253 is noise-free
pixel 7
254 is noisy
pixel 8
251 is noisy
pixel 9
251 is noisy
pixel 10
252 is noise-free
pixel 11
251 is noisy
pixel 12
254 is noisy
pixel 13
253 is noisy
pixel 14
252 is noise-free
pixel 15
251 is noisy
```

```
pixel 65521
[54] 0 is noisy
pixel 65522
79 is noise-free
pixel 65523
0 is noisy
pixel 65524
255 is noisy
pixel 65525
104 is noise-free
pixel 65526
0 is noisy
pixel 65527
255 is noisy
pixel 65528
60 is noise-free
pixel 65529
255 is noisy
pixel 65530
85 is noise-free
pixel 65531
85 is noisy
pixel 65532
86 is noisy
pixel 65533
89 is noisy
pixel 65534
89 is noisy
pixel 65535
75 is noisy
```

Number of noisy and noise free pixels

```
[55] no_of_noisy_pixels
45300

[56] no_of_noise_free_pixels
20236
```

References

[1]Singh, N., Oorkavalan, U. Triple Threshold Statistical Detection filter for removing high density random-valued impulse noise in images. *J Image Video Proc.* 2018, 22 (2018),https://doi.org/10.1186/s13640-018-0263-0

[2] T1 - Gupta, Vikas,Chaurasia, Vijayshri,Shandilya, MadhuRandom-valued impulse noise removal using adaptive dual threshold median filter,2014/10/16,VL-26,https://dl.acm.org/doi/abs/10.1016/j.jvcir.2014.10.004

[3] Neeti Singh, Thirusangu Thilagavathy, Ramasubramanian T. Lakshmipriya, Oorkavalan Umamaheswari, "Some studies on detection and filtering algorithms for the removal of random valued impulse noise", *Image Processing IET*, vol. 11, no. 11, pp. 953-963, 201

[4] Amit Saha,*Doing Math with Python*, 1st ed. Reading, Paula L. Fleming,2015,[E-book] Available:it-ebooks

[5]Allen B. Downey,*Think Stats: Probability and Statistics for Programmers*,2011,,[E-book] Available:Green Tea Press