IMAGE PROCESSING COURSE (EE6553)

Assignment 1

OCTOBER 10, 2020

SAEED KAZEMI
Student Number 3713280

DUE: OCT 8, 2020

A. Abstract

In this assignment, I worked on a MATLAB program for transformation and Spatial operations on an image. This project contained four functions and an m-file that call all functions. In section B, I talk over the techniques used for this assignment. In the next section, the output results are discussed based on images that are shown in part D, and finally, we can see codes written in the appendix.

B. Technical discussion

In this section, the techniques used in this program are discussed. In contrast manipulation, I used two loops for manipulation of each pixel based on its value. For Log transformation, I used an element-wise operation to calculate the amount of log for all of the pixels. Then they Multiplied Constant C, which has been defined as input argument.

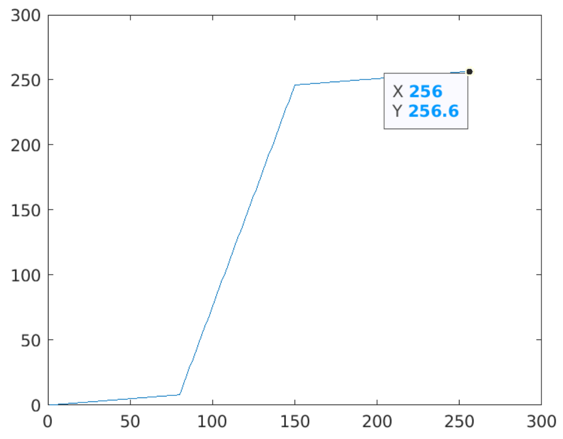
For the spatial operations task, I used two loops for computing. Also in this function, the border of the input image was ignored based on the input mask H. This mask could be a matrix with arbitrary size and value. Because of ignoring, the size of the output image is less than the input image. The below equation shows the differences between these two images.

*Size output image = Size input image – [2 * round (size mask / 2)]*

To show the histogram of the image, a function was written. This function returned a normalized vector.

C. Discussion of results

In this part, the result and pictures in the next section are discussed. As you can see, the image A is a low contrast image, and the image B shows the histogram of the image A. As the image B shows all of the pixels have a value between 90 and about 140. By the linear transformation, the image C is produced. If we plot the histogram of this image we notice that the range of pixels is distributed between 1 and 255 (image D). So, the output image has a perfect contrast.

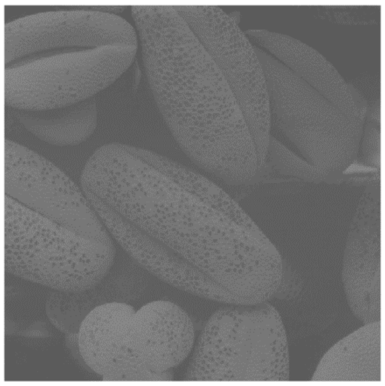


For the input image of logarithm transformation, the image E and I are used. These two pictures are so dark and the viewer could not notice their details. If we look at their histograms (image F and J), we can realize that all of their pixels are take values less than about 25. If the log transformation function is used the output image is a lighter image as the image G and K show. the histogram plots also show that the range of pixel values is distributed from 0 to 255 (image H and L). Therefore, more details are in the output image, and the contract increases.

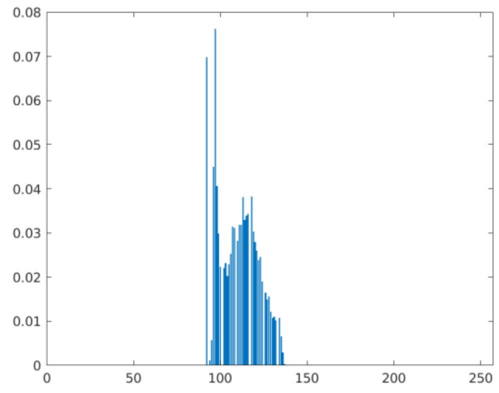
To perform a spatial operation, I used a mask for detecting the horizontal edges. This mask is a matrix 3 by 3 and these values `[4 4 4; 0 0 0; -4 -4 -4]`. The output image is image N. In this Image when the intensity of pixels is changed in the horizontal direction, the output is a large value or a white color. Otherwise, in a plain area, the output is zero or dark color. For finding the edge in the vertical direction, we could use this mask `[4 0 -4; 4 0 -4; 4 0 -4]`.

To perform zooming operation using pixel replication and bilinear interpolation I used the image O as the input image. This image is a low-resolution image and by zooming, I zoomed two times by two algorithms pixel replication and bilinear interpolation. As it shows in images P, Q and R, the replication algorithm change image to squares plain (the image P and Q) even by the low pass filter H. While the bilinear interpolation tried to produce a smoother image and blur (the image R).

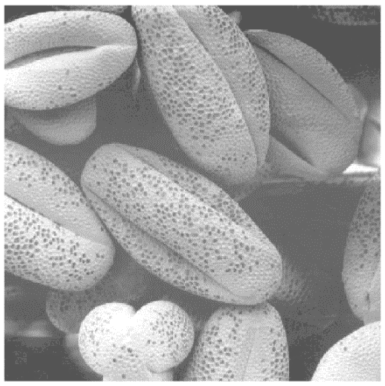
D. Image Results



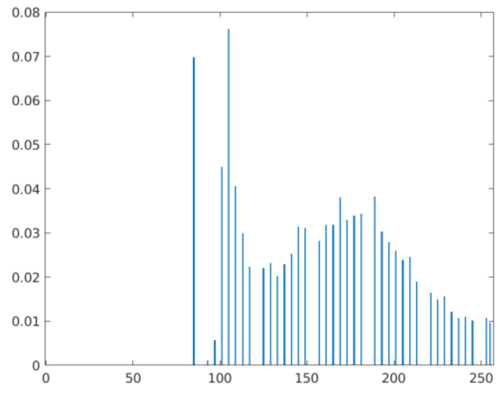
A) Input image for linear transformation



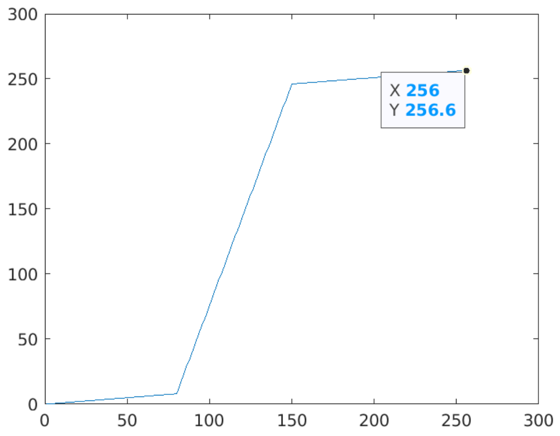
B) Histogram image A



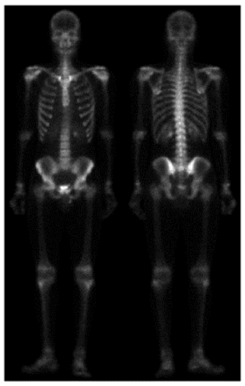
C) output image of the linear transformation



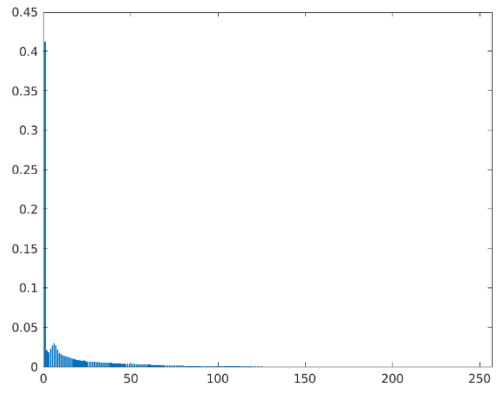
D) Histogram image C



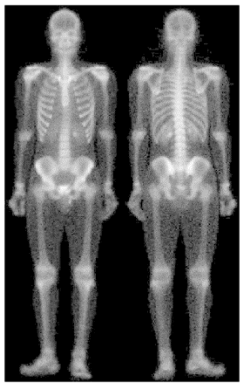
D2) the plot of the linear transformation



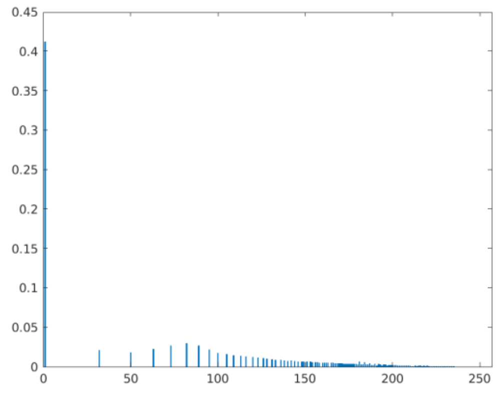
E) Input image for logarithm transformation



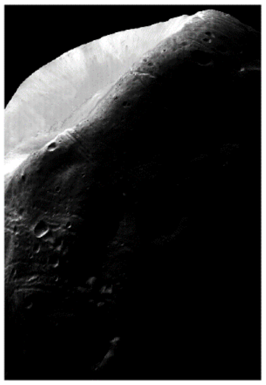
F) Histogram image E



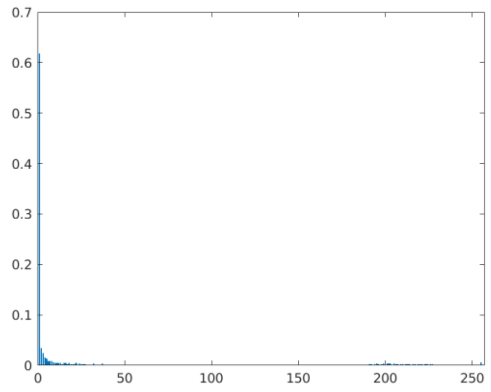
G) output image of logarithm transformation



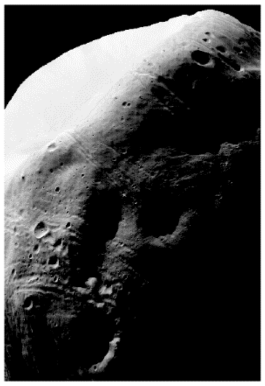
H) Histogram image G



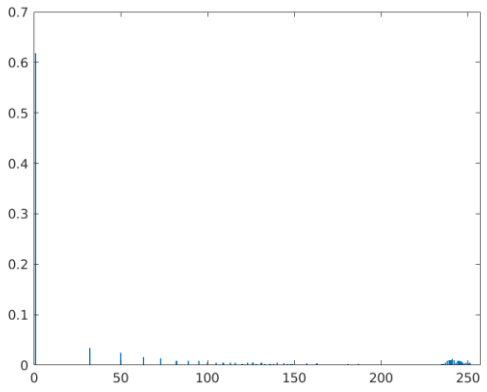
I) Input image for logarithm transformation



J) Histogram image I

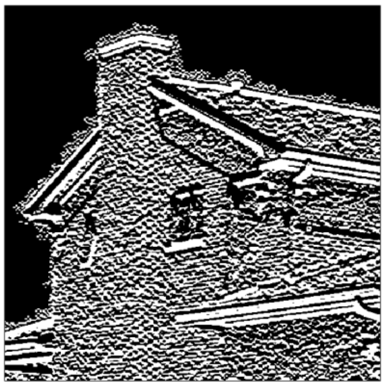


K) Output image of logarithm transformation



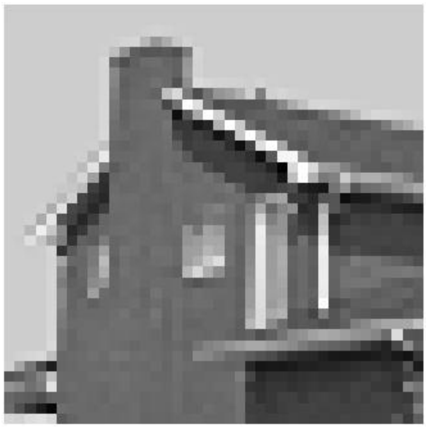
L) Histogram image K

M) Input image for spatial function by using a [H] mask

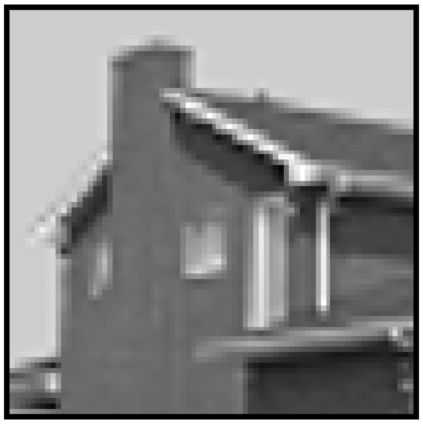


N) The output image of the edge detector with [H] mask

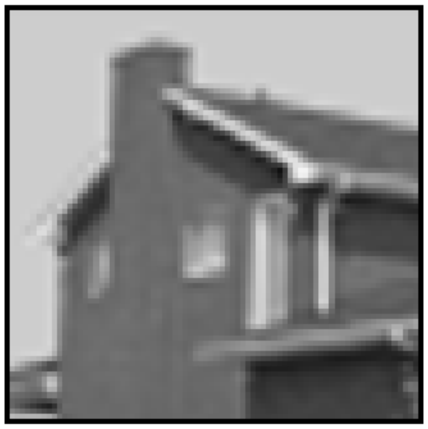
O) Low-resolution image for the input of zooming function



P) Zooming two times by replication pixel algorithm of image O



Q) Zooming two times by replication pixels algorithm of image O and the low pass filter H =[1, 1; 1,1]



R) Zooming two times by bilinear algorithm of image O

E. Appendix (codes)

I. M-file

```
close all;
```

```
clc


%% Piecewise linear transformation
A = imread("image.tif");
B = LinearTrans(A,...  %inputImage
                0.2,...%FirstSlope
                100,...%FirstPoint
                10,...%SecondSlop
                120,...%SecondPoint
                0.2);  %ThirdSlop

C = LinearTrans(1:256,...  %inputImage
                0.1,...%FirstSlope
                80,...%FirstPoint
                3.4,...%SecondSlop
                150,...%SecondPoint
                0.1);  %ThirdSlop
plot(1:256,C) %Show the LinearTrans function

B = uint8(B);
figure(1)
imshow(A)
OutputBarA = ImageHist(A);
figure(2)
bar(OutputBarA)
figure(3)
imshow(B)
OutputBarB = ImageHist(B);
figure(4)
bar(OutputBarB)


%% Log transformation


c =  0;
B = LogTrans(A,...     %inputImage
                c);  %Scale
B = uint8(B);
figure(1)
imshow(A)
OutputBarA = ImageHist(A);
figure(2)
bar(OutputBarA)
figure(3)
imshow(B)
OutputBarB = ImageHist(B);
figure(4)
bar(OutputBarB)
```

```
  %% spatial operation upon an image using a [H] mask
  H = [4  4  4;
       0  0  0;
      -4 -4 -4];
  B = Filt(A,...      %inputImage
               H);    %Scale
  B = uint8(B);
  figure(1)
  imshow(B)


%% zooming operation using pixel replication
A = imread("Lowhouse.png");
%A = imread("highhouse.png");


A = rgb2gray(A);
B = zeros(2*size(A,1),size(A,2));
B(1:2:end,1:end) = A;
B(2:2:end,1:end) = A;
B1 = zeros(2*size(A,1),2*size(A,2));
B1(1:end,1:2:end) = B;
B1(1:end,2:2:end) = B;
figure(1)
imshow(A)
truesize
B1 = uint8(B1);
figure(2)
imshow(B1)
truesize
H = [0  0  0;
     0  1  1;
     0  1  1];


C = Filt(B1,...       %inputImage
             H);    %Scale
C = uint8(C);
figure(3)
imshow(C)
truesize




%% zooming operation using bilinear interpolation
B = zeros(2*size(A,1),size(A,2));
B(1:2:end,1:end) = A;
B(2:2:end,1:end) = A;
B1 = zeros(2*size(A,1),2*size(A,2));
B1(1:end,1:2:end) = B;
B1(1:end,2:2:end) = B;
```

```
H = [.25 .5 .25;
.5 1 .5;
.25 .5 .25];


C = Filt(B1,... %inputImage
H); %Scale
C = uint8(C);
figure(3)
imshow(C)
truesize
```

II. Histogram Function

```
function OutputBar = ImageHist(inputImage)

    c = zeros(1,256);
    for m = 1:size(inputImage,1)
        for n = 1:size(inputImage,2)
            c(inputImage(m,n)+1) = c(inputImage(m,n)+1) + 1;
        end
    end
    OutputBar = c./(size(inputImage,1)*size(inputImage,2));


end
```

III. Filtering Function

```
function OutputImage = Filt(inputImage,H)
     inputImage = double(inputImage);
     %inputImage = ones(20,27);
     OutputImage = zeros(size(inputImage,1),size(inputImage,2));
     %H = [];
     [a,b] = size(H);

     X = fix(a/2)+1;
     Y = fix(b/2)+1;



    for i=X:size(inputImage,1)-X+1
```

```
        for j=Y:size(inputImage,2)+1-Y
            OutputImage(i,j) = sum((...
                inputImage(i-X+1:i+X-1,j-Y+1:j+Y-1).*H) ,'all')/...
                sum(H, 'all');
        end
    end
end
```

IV.Linear transformation Function

```
function OutputImage = LinearTrans(inputImage,FirstSlope,FirstPoint,...
                                    SecondSlop,SecondPoint,ThirdSlop)
    OutputImage = zeros(size(inputImage,1),size(inputImage,2));
    for i=1:size(inputImage,1)
        for j=1:size(inputImage,2)

            if inputImage(i,j) > SecondPoint

                OutputImage(i,j) = (ThirdSlop * ...
                    (inputImage(i,j)-SecondPoint))+((SecondSlop  * ...
                    (SecondPoint-FirstPoint))+(FirstSlope*FirstPoint));

            elseif inputImage(i,j) < FirstPoint
                OutputImage(i,j) = FirstSlope * inputImage(i,j);

            else
                OutputImage(i,j) = (SecondSlop  * ...
                    (inputImage(i,j)-
FirstPoint))+(FirstSlope*FirstPoint);

            end


        end
    end
end
```

V. Logarithm transformation Function

```
function OutputImage = LogTrans(inputImage,c)
    inputImage = double(inputImage);
    OutputImage = c .* log((inputImage + 1));
end
```