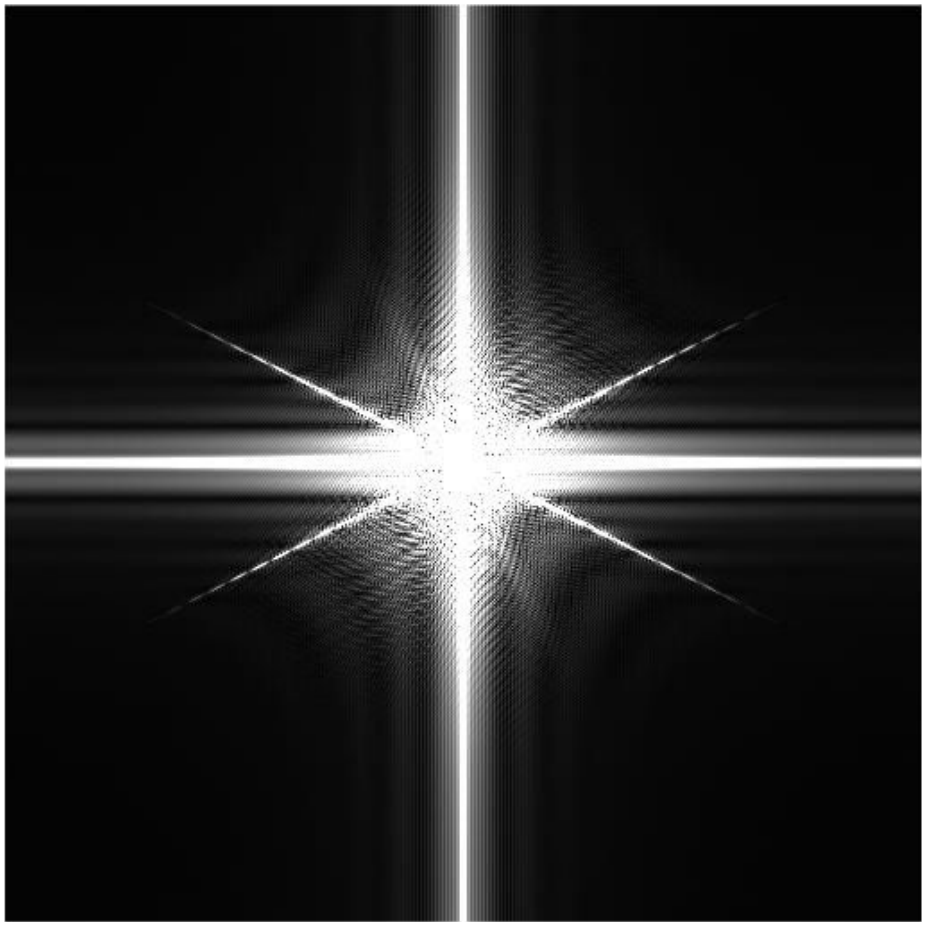
Name: Pankaj Bhootra **Matric No.:** A0144919W

EE4704 Project Report

Note: All the code for the different tasks in this project can also be found in various script files submitted along with this report (taskA.m, taskB.m, taskC.m, percentile.m) under the folder 'scripts'. All images attached in this report are also submitted as jpeg files under the folder 'images'.

A. Fourier Spectrum

Answer 1. The following Fourier spectrum is obtained with 2D DFT of *test1.bmp*.



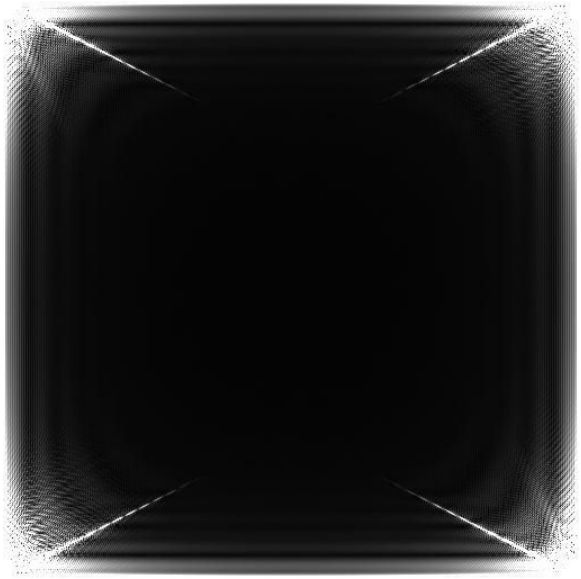
Fourier spectrum for *test1.bmp*

(code on next page)

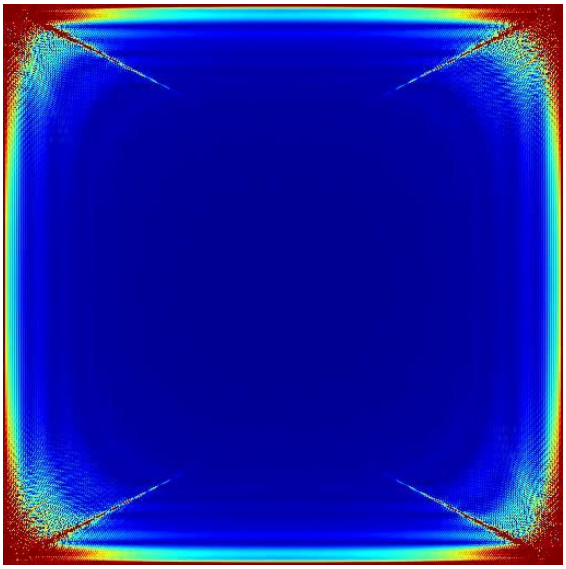
```
%% Question A1

% Load image test1.bmp towards task of Question A1
Img = double(imread('images\test1.bmp'));
[M, N] = size(Img);
Output = zeros(M, N);
SumVal = 0;

% Calculate 2D Discrete Fourier Transform
[nx, ny] = ndgrid([0:M-1]-(M-1)/2, [0:N-1]-(N-1)/2);
du = 1;
for u = [0:M-1]-(M-1)/2
    dv = 1;
   for v = [0:N-1]-(N-1)/2
       SumVal = sum(sum(Img.*exp(-1i*2*3.1416*(u*nx/M+v*ny/N))));
       Output(du,dv) = SumVal;
       dv = dv+1;
   end
   du = du+1;
end
% Calculate Spectrum and display it
img_A1 = uint8(abs(Output)/60);
imshow(img_A1);
```

Code to calculate 2D DFT of an input image, in this case *test1.bmp*

Answer 2. The following spectrum is obtained after shifting the Fourier spectrum obtained in *Answer 1* to the center with "fftshift".



Shifting fourier spectrum obtained in *Answer 1* to the center with "fftshift"

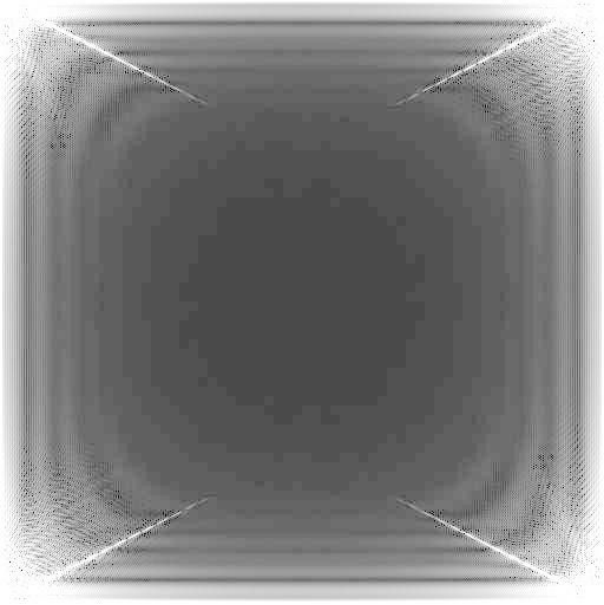


Same Fourier spectrum displayed using the 'false color' color map for better visualization

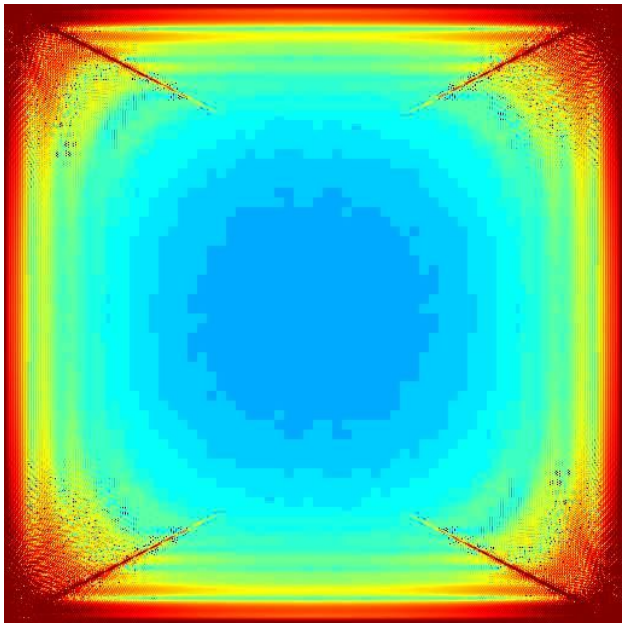
```
%% Question A2

% Shift fourier spectrum obtained in A1 using fftshift and display it
img_A2 = fftshift(img_A1);
imshow(img_A2);
```

Code to perform fftshift on the spectrum obtained in *Answer 1*

Answer 3. The log function, specifically “log10”, is applied to enhance the Fourier spectrum obtained in *Answer 2* and the following spectrum is obtained.



Applying log (“log10”) function to Fourier spectrum obtained in *Answer 2*



Fourier spectrum displayed using the ‘false color’ color map for better visualization

```
%% Question A3
% Use log function to enhance the fourier spectrum and display using 'false
% color' color map
img_A3 = uint8(255 / log10(double(1 + max(img_A2(:)))) * log10(double(1 + img_A2)));
falseColorImage_A3 = ind2rgb(img_A3, jet(256));
imshow(falseColorImage_A3);
```

Code to enhance the spectrum using the log (“log10”) function

Answer 4. As seen in the previous answer, the “gray scale” provides better visualization than “false color” as the color differences between the different spectrum components is not overlapping. In case of the false color image, the cyan color region becomes more overlapping, making those spectrum components less vivid and separate from each other. Hence, “gray scale” would be the recommended color map to use in this case.

B. Contrast Stretching

Answer 1.

(a) The code for contrast stretching of *test2.bmp* is implemented as below:

```
%% Question B1 and B2

% Applying contrast stretching to test2.bmp with best value of alpha
img_B2 = hstretch(double(imread('images\test2.bmp')), 1.6087);

% Implement Contrast stretching for input image with adjustable stretch
% alpha
function Iout = hstretch(Iin, alpha)
  meanVal = mean(Iin(:));
  Iout = uint8(alpha * Iin - alpha * meanVal + meanVal);
end
```

(b) Alpha should be chosen such that the contrast is maximised without clipping at gray levels 0 and 255. Hence best value of alpha is given by the value that stretches the input range of image values to as close to occupying the full range of 0 to 255 as possible, using the same formula as given in Tutorial D, Question 2.

The range of prominent values of the input image, as seen from its histogram, is 35 to 184 (the actual minimum is 28 and maximum is 255, but the values before 35 and after 184 are not very prominent). Hence, we need to stretch the range 35-184 to 0-255 as closely as possible. The mean value of the given image is 71.0325. Hence, the best value of alpha for given input image and mean value is 1.6087. This gives us the following image and its histogram as a result (see next page):

***test2.bmp* after Contrast Stretching**

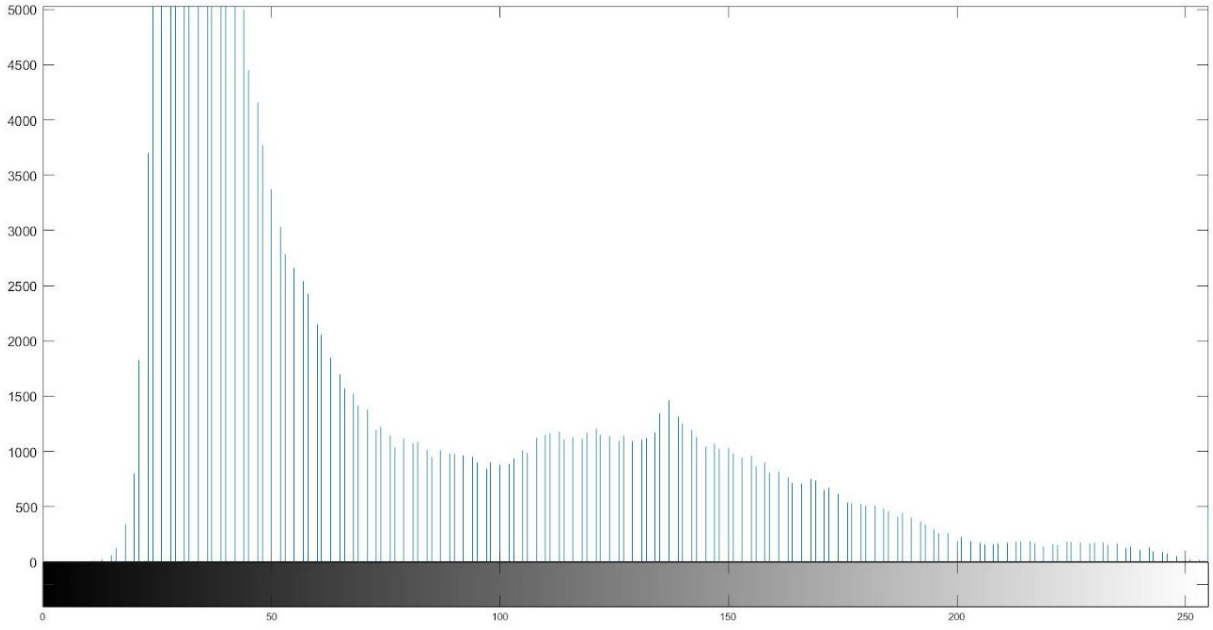


Image Histogram for *test2.bmp* after performing the contrast stretching

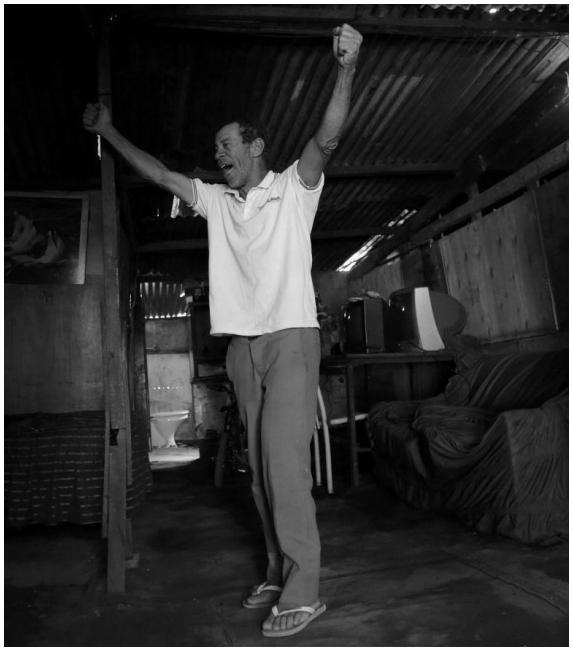
Although not clearly visible in the histogram, the maximum gray level found in this enhanced image is 255, and the minimum is 2. The input image has been stretched to occupy the full range of grayscale values without any clipping.

C. Histogram Transformation

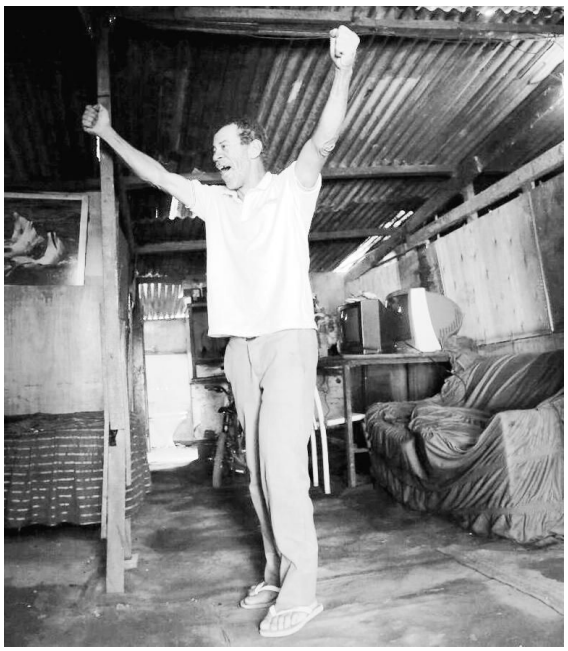
Answer 1. Code for histogram equalization is as below, using the Image Processing Toolbox:

```
img = imread('images\test3.bmp');
img_enhanced = histeq(img, 256);
```

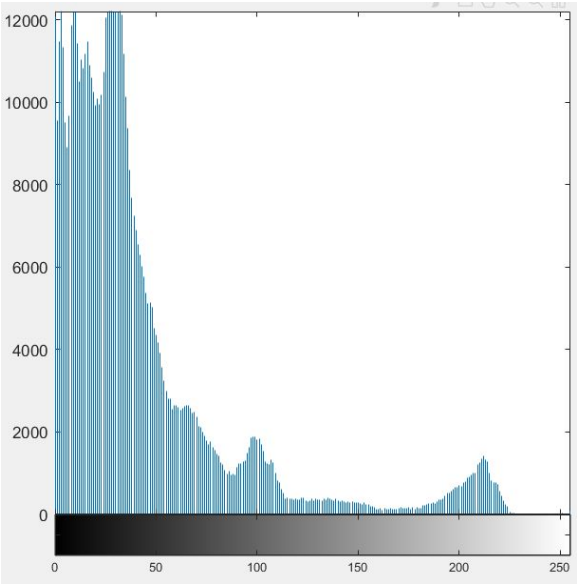
The following compares the old image and its histogram against that of the new image. As seen clearly, histogram equalization works well as it increases the global contrast and evenly distributes the gray level intensities. However, the image becomes a bit too bright.

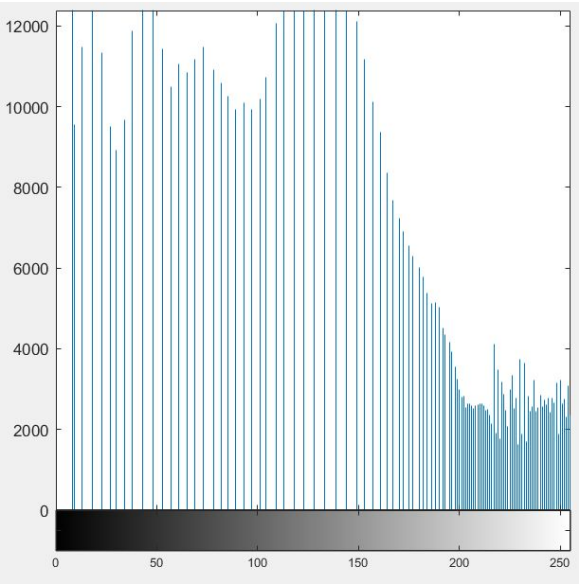


Before histogram equalization



After histogram equalization





Answer 2.

(a) The complete implementation in MATLAB is provided below. This is also provided in the taskC.m file submitted along with this report.

```
%% Question C2

keypoints = [11 51; 23 70; 34 120; 59 200];
img_C = htrans(img, keypoints);

function Iout = htrans(Iin, keypoints)
  % Determine number of keypoints in input
  N_K = size(keypoints, 1);
  % Initialise an array to store all N_K-1 slope values
  slopes = zeros(1, N_K-1);
  % Calculate all N_K-1 slope values
  for i = 1:(N_K-1)
      dx = keypoints(i+1, 1) - keypoints(i, 1);
      dy = keypoints(i+1, 2) - keypoints(i, 2);
      slopes(1,i) = double(dy)/double(dx);
  end
  % Get a list of all the original image keypoints/gray levels
  r = keypoints(:, 1)';
  % Get a list of all the targeted image keypoints/gray levels
  s = keypoints(:, 2)';
  % If 0 in original image is not mapped, add 0,0 keypoint
  if r(1) ~= 0
      r = [0, r];
      s = [0, s];
      slopes = [double(s(2))/double(r(2)), slopes];
  end
  % If 255 in original image is not mapped, add 255,255 keypoint
  if r(end) ~= 255
      r = [r, 255];
      s = [s, 255];
      slopes = [slopes, double(s(end) - s(end-1))/double(r(end) - r(end-1))];
  end
  % Safety check to prevent out of range
  slopes = [slopes, 0];
  % Get size of input image
  [M, N] = size(Iin);
  % Initialize output image to all zeros
  Iout = zeros(M, N);
```

```
  % For every value in input image, determine which range it belongs to and
  % accordingly, apply the corresponding mapping
  for i = 1:M
      for j = 1:N
          rangeIndex = 1;
          for k = 2:size(r, 2)
              if Iin(i, j) >= r(k)
                  rangeIndex = k;
              end
          end
          Iout(i, j) = slopes(rangeIndex) * (Iin(i, j) - r(rangeIndex)) +
                       s(rangeIndex);
      end
  end
  % Plot transformation function
  plot(r,s);
end
```

(b) As seen in the implemented code, the argument `keypoints` is implemented exactly as required.

(c) As seen in the implemented code, the function is capable of accepting any number of keypoint pairs and is not restricted to any limit.

(d) To determine the "best" keypoints to use, I implemented a further script (percentile.m) that calculates the nth percentile of a given list of values. This is used to determine the different grayscale values of the input image at the 20th percentile, 40th, 60th, and 80th percentile. The idea is that these percentile values of the input image should be ideally mapped to the 20th/40th/60th/80th percentage values of the full range, i.e., 0 to 255, to give the "best" contrast and visual clarity. Hence, my keypoints is a list of 4 keypoint pairs where target points are 51, 102, 153 and 204 (20th/40th/60th/80th percentage values of 0 to 255) and the original points are thus calculated to be 11, 23, 34 and 59 (the 20th/40th/60th/80th percentile values of the input image). Hence, my keypoints array to give the "best" result is:

```
[  11  51;
   23 102;
   34 153;
   59 204  ]
```

The histogram, plot transformation function and the resulting image are shown in the next two pages.

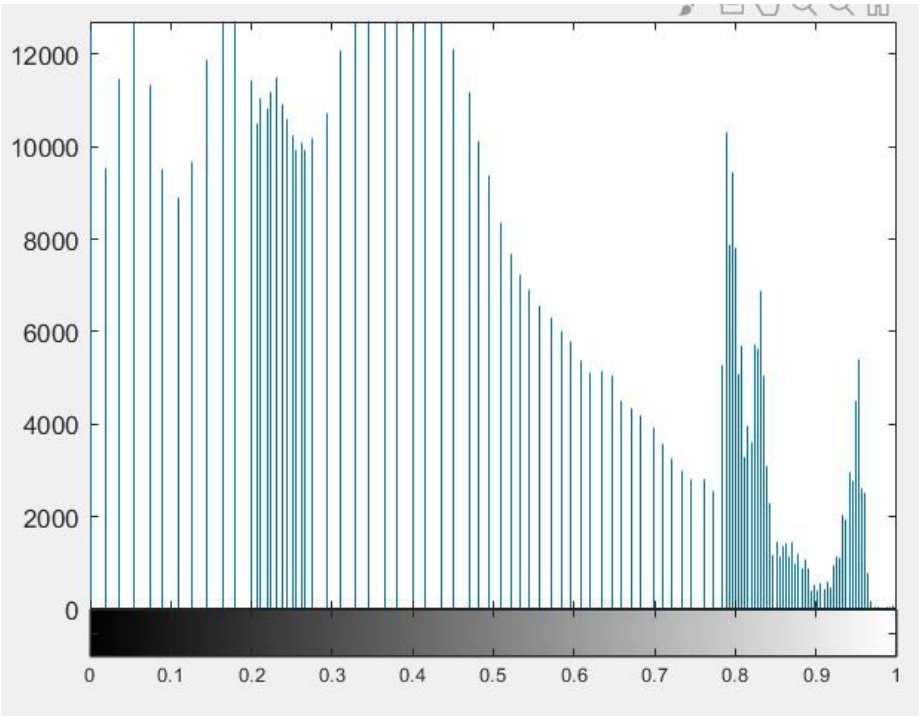
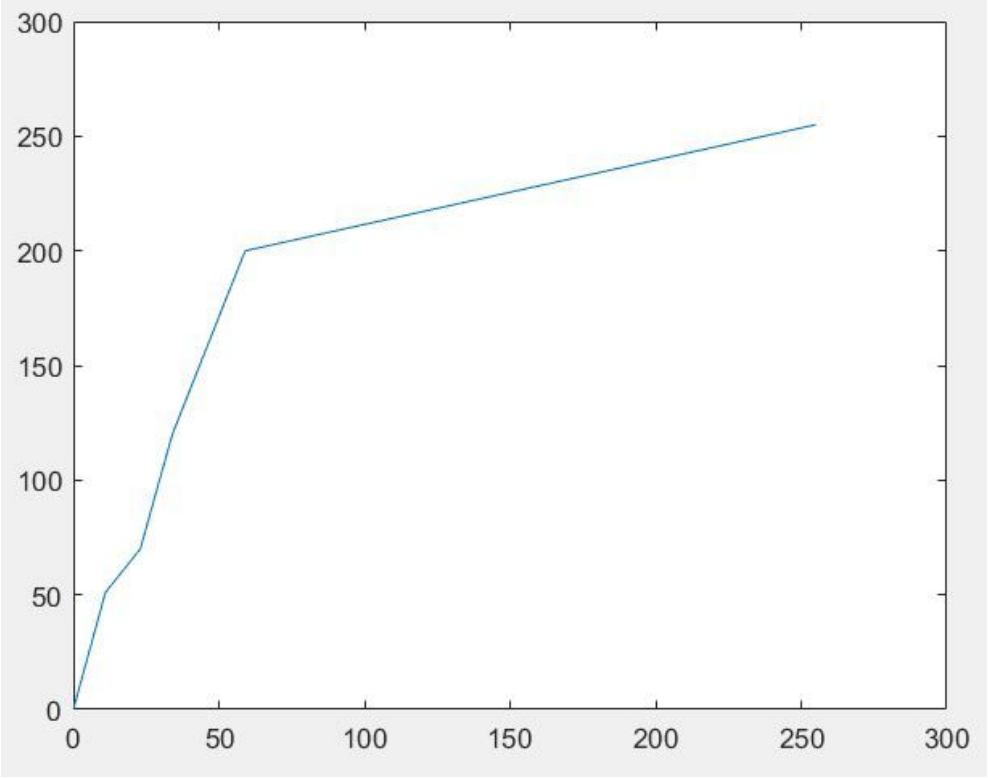
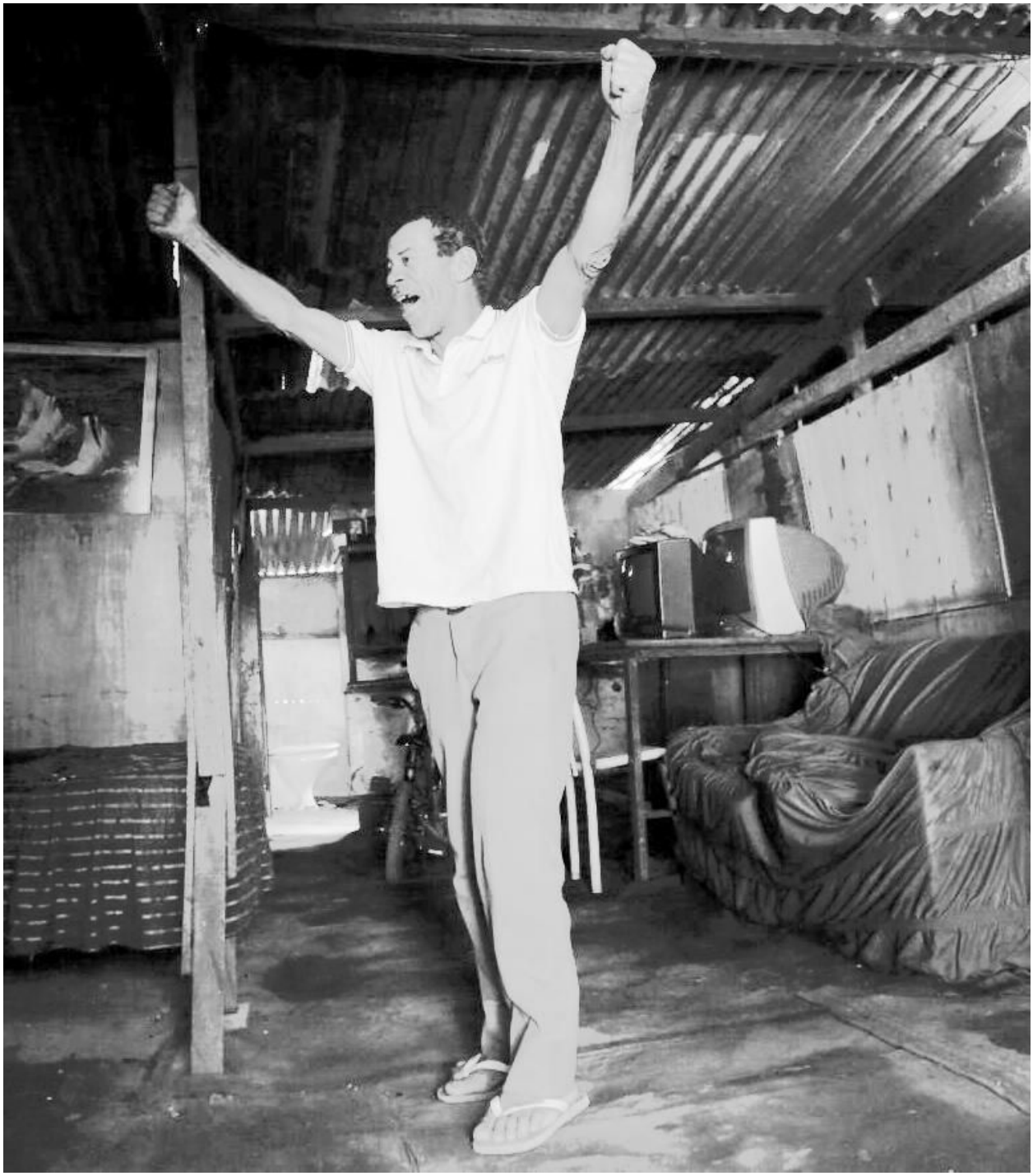


Image histogram of the resulting image after histogram transformation using the keypoints mentioned earlier



Plot of the transformation function used



Resulting image after histogram transformation using the keypoints mentioned earlier

x

THE END