# BİL626 Assignment#2 Solution

# Histogram Matching

# Erhan Akagündüz

## MATLAB Script Codes

‘main.m’ source code is given below.

|  |
| --- |
| % author : erhan akagunduz    image1 = imread('can4\_25\_09.png');  image1 = double(image1);  % add 1 to 1-to-1 mapping, matlab has no 0 index  image1 = image1 + 1;  image2 = imread('can9\_2\_13.png');  image2 = double(image2);  image2 = image2 + 1;    % Calculate normalized histogram of image-1  normalizedHistIm1 = calculateNormalizedHistogram(image1);  % Calculate normalized histogram of image-2  normalizedHistIm2 = calculateNormalizedHistogram(image2);    % Uniform histogram of image-1  histEqualizedImage1 = equalizeHistogram(image1);  % Calculate histogram of uniformed histogram image-1  normHistOfHistEqualizedIm1 = calculateNormalizedHistogram(histEqualizedImage1);  % Save the uniformed image-1  histEqualizedImage1 = histEqualizedImage1 - 1;  histEqualizedImage1 = uint8(histEqualizedImage1);  imwrite(histEqualizedImage1, 'histEqualizedImage1.png');    %Uniform histogram of image-2  histEqualizedImage2 = equalizeHistogram(image2);  % Calculate histogram of uniformed histogram image-2  normHistOfHistEqualizedIm2 = calculateNormalizedHistogram(histEqualizedImage2);  % Save the uniformed image-2  histEqualizedImage2 = histEqualizedImage2 - 1;  histEqualizedImage2 = uint8(histEqualizedImage2);  imwrite(histEqualizedImage2, 'histEqualizedImage2.png');    % Calculate cdf map of image-1  Tr = calculateCumulativeDistributionFunctionMap(normalizedHistIm1);  % Calculate cdf map of image-2  Gz = calculateCumulativeDistributionFunctionMap(normalizedHistIm2);  % Calculate inverse map of image-2  inverseGz = calculateInverseFunctionMap(Gz);    % Calculate histogram matching map of new image  [m, n] = size(inverseGz);  Hr = zeros(m, n);  for i = 1 : length(inverseGz);         Hr(i) = inverseGz(Tr(i));  end    % Calculate pixel values of new image using  % histogram matching map  [m, n] = size(image1);  newImage = zeros(m,n);  for i = 1 : m      for j = 1 : n          newImage(i,j) = Hr(image1(i,j));      end  end    % Calculate normalized histogram of new image  normalizedHistNewIm = calculateNormalizedHistogram(newImage);  newImage = uint8(newImage - 1);  imwrite(image3, 'newImage.png');    figure(1);  title('Original Histogram of Image-1');  bar(normalizedHistIm1, 'b');    figure(2);  title('Original Histogram of Image-2');  bar(normalizedHistIm2, 'b');    figure(3);  title('Uniform Histogram of Image-1 by Using Tr');  bar(normHistOfHistEqualizedIm1, 'b');    figure(4);  title('Uniform Histogram of Image-2 by Using Gz');  bar(normHistOfHistEqualizedIm2, 'b');    figure(5);  title('Histogram of New Image by Using InverseGz[Tr]');  bar(normalizedHistNewIm, 'b');    % Plot all images together  figure(6);  subplot(3,1,1);  title('Image-1');  imshow(uint8(image1));    subplot(3,1,2);  title('New Image');  imshow(newImage);    subplot(3,1,3);  title('Image-2');  imshow(uint8(image2)); |

‘calculateNormalizedHistogram.m’ source code is given below.

|  |
| --- |
| function histogram = calculateNormalizedHistogram(I2D)      [m,n] = size(I2D);      length = m \* n;      I1D = reshape(I2D, length, 1);      histogram = double(zeros(256, 1));        for i = 1 : length          histogram(I1D(i)) = histogram(I1D(i)) + 1;      end        % Normalize the histogram      histogram = histogram / length;  end |

‘calculateCumulativeDistributionFunctionMap.m’ source code is given below.

|  |
| --- |
| function cdf = calculateCumulativeDistributionFunctionMap(histogram)      [m, n] = size(histogram);      sz = m \* n;      cdf = zeros(sz, 1);        sum = 0;      for i = 1 : sz          sum = sum + histogram(i);          cdf(i) = sum;      end        % Convert cdf to cdf map to use in calculations  % +1 to map, there is no 0 value index (1 to 256)      for i = 1 : sz         cdf(i) = round(cdf(i) \* 255) + 1;      end    end |

‘calculateInverseFunctionMap.m’ source code is given below.

|  |
| --- |
| function inverseFunctionMap = calculateInverseFunctionMap(functionMap)      l = length(functionMap);      inverseFunctionMap = zeros(l, 1);      % Calculate inverse map values      % There should be only one value in function map      % to calculate inverse map.      for i = 1 : l          v = find(functionMap(:) == (i));  % if length of v is 1 the there is inversion, v(1) is the inverse map          if 1 == length(v)              inverseFunctionMap(i) = v(1);          end      end      % Calculate the values which are zeros by      % approximating value.      inverseFunctionMap(1) = min(functionMap);      inverseFunctionMap(l) = max(functionMap);      i = 2;      while i < l            % find the first zero          a = i;          while (inverseFunctionMap(a) > 0)              a = a + 1;          end            %find number other than 0 or boundary          b = a;          while (inverseFunctionMap(b) == 0)              b = b + 1;          end          % Calculate the step value to assign new values          % to zero valued positions          step = (inverseFunctionMap(b) - inverseFunctionMap(a - 1)) / (b - (a - 1));          % First value to be used to calculate new value          value = inverseFunctionMap(a - 1);          % Assign new values to zero valued positions          k = 1;          for j = a : b - 1              inverseFunctionMap(j) = round(value + k \* step);              k = k+1;          end          i = b;      end  end |

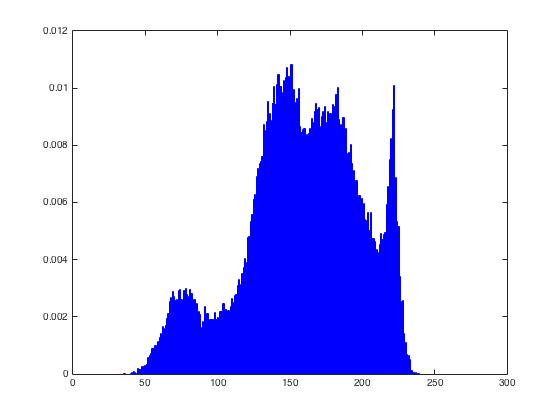
‘equalizeHistogram.m’ source code is given below.

|  |
| --- |
| function histEqIm = equalizeHistogram(I2D)      normalizedHistogram = calculateNormalizedHistogram(I2D);      cdfMap = calculateCumulativeDistributionFunctionMap(normalizedHistogram);        [m, n] = size(I2D);      histEqIm = zeros(m, n);      histEqIm = double(histEqIm);        for i = 1 : m          for j = 1 : n              histEqIm(i,j) = cdfMap(I2D(i,j));          end      end  end |

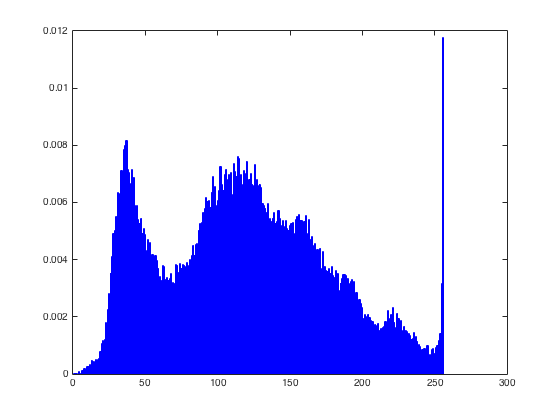
## Original Histograms

Original histogram plots of images are given here. All images are generated from normalized histogram values.

Original Histogram of Image-1



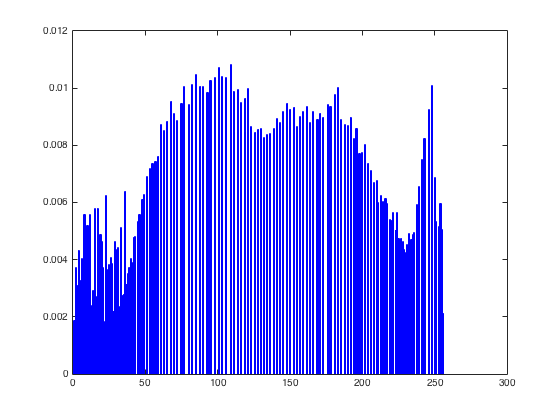
Original Histogram of Image-2



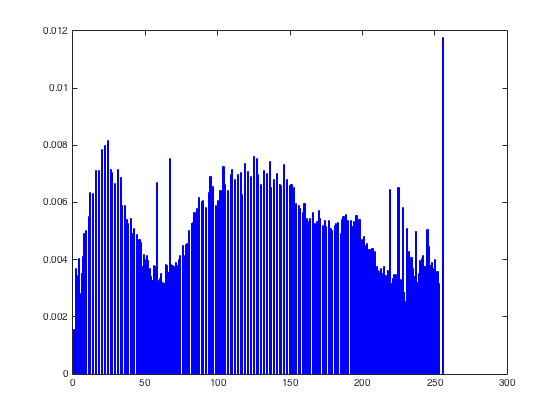
## Uniform Histograms

Uniform histogram plots of images are given here. T(r) and G(z) are used for calculations.

Uniformed Histogram of Image-1

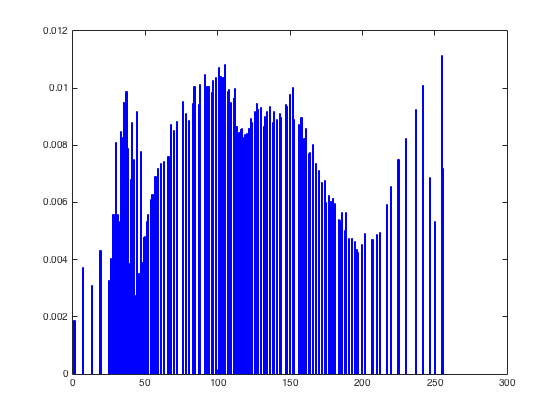


Uniform Histogram of Image-2



## Histogram of New Image

Histogram of new image is given below.



## Images

All images are given below. First one is image-1, in the middle newly generated image is seen and the last one is image-2

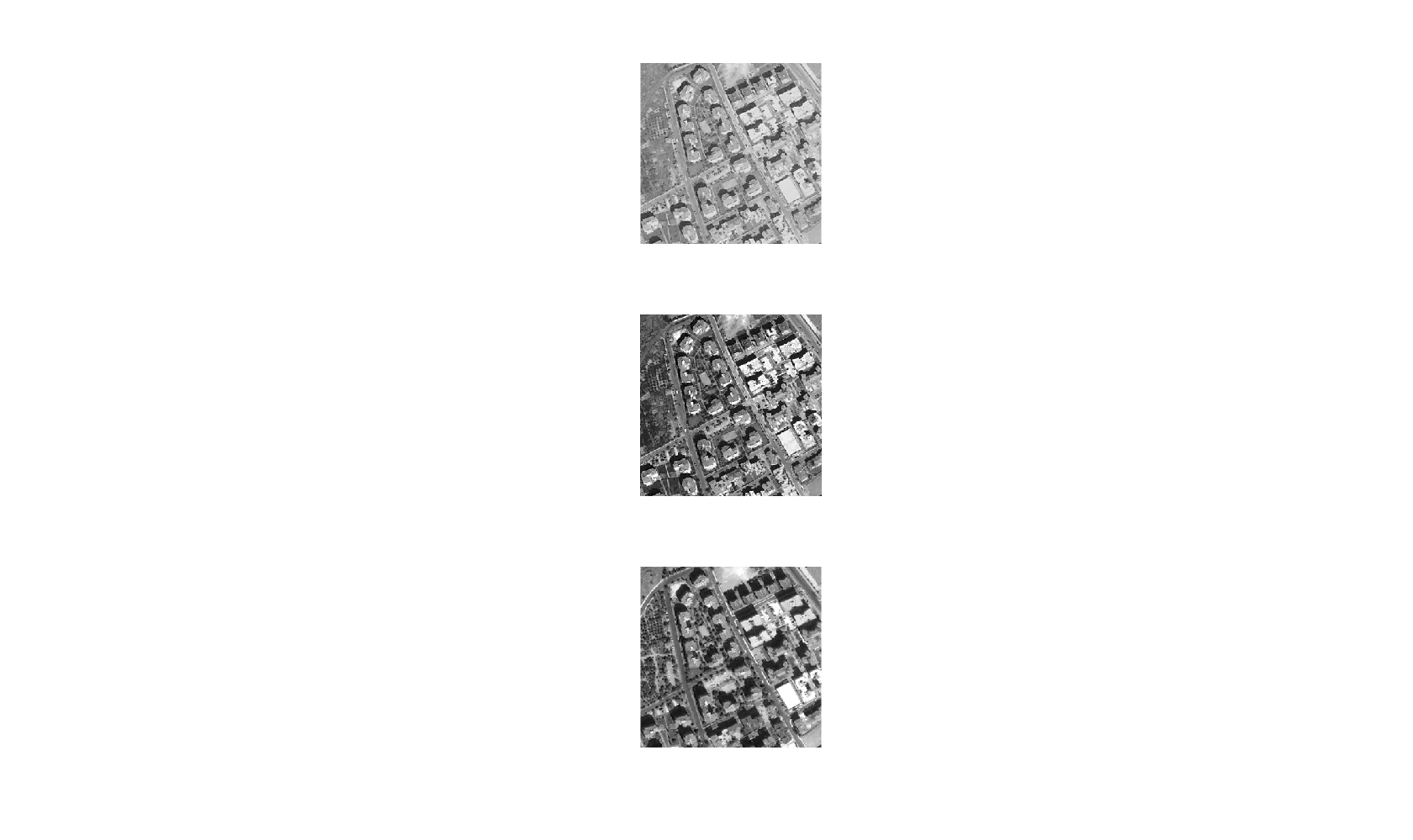


Image-1



New Image





Image-2

## Comments

By applying G-1[T(r)], I tried to find similar histogram results between image-2 and new image. As shown in the figures, histogram shape of new-image has similar characteristic with respect to histogram of image-2. As a result, new generated image has similar contrast perception with image-2. Image-1 has narrower histogram with respect to new image and image-2. For this reason, image-1 is seen flu.

By uniforming histogram of images, I have seen that histogram graphs of images are widened and more pixel values are used to generate new images.