Digital Image Processing HW1

Hejung Yang
School of Electrical and Electronic Engineering, Yonsei University.

Problem 1: Piecewise Linear Transformation

hw1_1.m loads cameraman.tif image file, applies two different piecewise linear transform to the image, and plot the images in order.

PiecewiseLinearTr.m defines piecewise linear transform of given input and output transformed image. Algorithm1 is the pseudocode for implementing PiecewiseLinearTr.m.

Algorithm 1: PiecewiseLinearTr.m

$output = zeros$;
$slope = zeros$;
$y\_inter = zeros$;
for $i = 1 : (len(a) - 1)$ **do**
  $slope_i = (b(i+1) - b(i))/(a(i+1) - a(i))$;
  $y\_inter_i = (a(i+1) * b(i) - a(i) * b(i+1))/(a(i+1) - a(i))$;
end
for $i = 1 : M$ **do**
  for $j = 1 : N$ **do**
    for $k = 1 : (len(a) - 1)$ **do**
      if $input(i, j)$ *between* $a(k), a(k+1)$ **then**
        $output(i, j) = slope_k * input(i, j) + y\_inter_k$;
      end
    end
  end
end

Result of hw1_1.m can be found in Figure2. The leftmost one is the original image, middle is the transformed version with segment coordinates of [0,1], [1,0]. The rightmost one is from the segment coordinates of [0 .25 .5 .75 1],[0 .75 .25 .5 1].

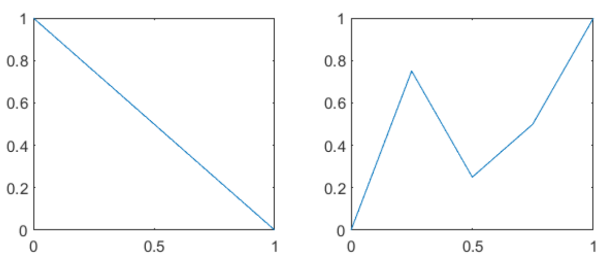

Figure 1: transformation functions. left: coordinates of [0,1], [1,0], right: coordinates of [0 .25 .5 .75 1],[0 .75 .25 .5 1].

Figure1 plots two transformation functions. It can be seen that left transformation inverses the input intensity, which is proved on middle version of Figure2. In the rightmost cameraman image, darkest part like coat is whitened and background, which has relatively high intensity, is darkened. This corresponds with right function of Figure1.

Figure 2: result of running hw1_1.m

Problem 2: Image Histogram

hw1_2.m loads input.jpg and plot histogram of the image. Hist.m implements gathering histogram statistics. Algorithm2 is the pseudocode for the file.

Algorithm 2: Hist.m

$hist = zeros$;
for $i = 1 : M$ **do**
  for $j = 1 : N$ **do**
    $hist(input(i, j)) + = 1$;
  end
end
plot *hist*;

Figure3 is the result of the hw1_2.m.

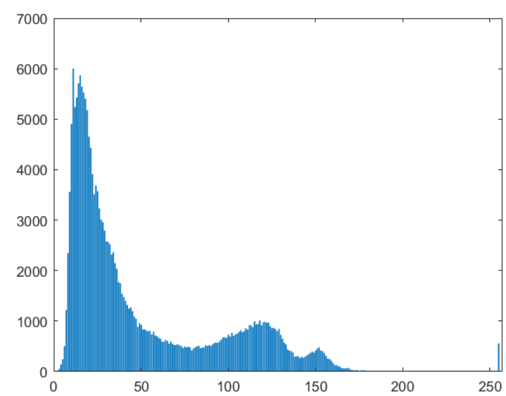

Figure 3: histogram visualizations.

Problem 3: Histogram Equalization

hw1_3.m loads input.jpg and applies histogram equalization. HistEq.m accepts input as image and output histogram-equalized version of the image. The pseudocode for HistEq.m can be found in Algorithm3.

Figure4 depicts the resulting image and its corresponding hsitogram. It can be seen that the resulting image has enhanced contrast, and the equalized histogram has more flattened distribution than the original histogram. Transformed histogram follows more to uniform than the original.

Problem 4: Histogram Equalization

Histogram equalized image(left of Figure4) has better overall contrast than the original, but some details on lighter part is lost due to over-brightness. To solve the issue, adaptive histogram equalization is implemented in HistEq_v2.m. In naive approach, each pixel is equalized base on the histogram of its neighbors. This requires too much computation complexity, thus each pixel is interpolated from the neighboring tile histograms. Detailed implementation is in Algorithm4. Detailed corner case exceptions are ignored.

This gives better visualization than naive histogram equalization in Figure5 (third image). Some details on sidewall of the building is restored, fence becomes sharper etc. However, some artifacts also arise, make it look

Algorithm 3: HistEq.m

$hist = zeros$;
$lut = zeros$;
for $i = 1 : M$ **do**
  for $j = 1 : N$ **do**
    $hist(input(i, j)) += 1$;
  end
end
for $i = 1 : L$ **do**
  $z = sum(hist(1 : i))$;
  $z = z * (L - 1)/(M * N)$;
  $lut(i) = z$;
end
$output = input$;
for $i = 1 : M$ **do**
  for $j = 1 : N$ **do**
    $output(i, j) = lut(input(i, j))$;
  end
end

Algorithm 4: HistEq_v2.m

$hist = zeros$;
$lut = zeros$;
for $tile\_i = 1 : tile\_x$ **do**
  for $tile\_j = 1 : tile\_y$ **do**
    build lut_i, j for $tile_i, j$;
  end
end
$output = input$;
for $i = 1 : input\_x$ **do**
  $tile\_i = floor(i/tile\_w - 0.5)$;
  $prob\_i = (i/tile\_w - 0.5) - floor(i/tile\_w - 0.5)$;
  for $j = 1 : input\_y$ **do**
    $tile\_j = floor(j/tile\_h - 0.5)$;
    $prob\_j = (j/tile\_h - 0.5) - floor(j/tile\_h - 0.5)$;
    $output(i, j) = ((lut_{i,j}(input(i, j)) * (1 - prob\_j)) +$
    $(lut_{i,j+1}(input(i, j)) * prob\_j)) * (1 - prob\_i) +$
    $((lut_{i+1,j}(input(i, j)) * (1 - prob\_j)) +$
    $(lut_{i+1,j+1}(input(i, j)) * prob\_j)) * prob\_i$;
  end
end

Algorithm 5: HistMatching.m

build lut_t for historgram equalization given $input$;
build lut_g for histogram equalization given $input\_match$;
$lut_{g'} = zeros$;
for $i = 1 : L$ **do**
  $lut_{g'}(lut_g(i)) = i$;
end
for $i = 1 : L$ **do**
  if $lut_{g'}(i)$ *is undefined* **then**
    assign interpolated value on $lut_{g'}(i)$;
  end
end
$output = input$;
for $i = 1 : M$ **do**
  for $j = 1 : N$ **do**
    $output(i, j) = lut_{g'}(lut_t(input(i, j)))$;
  end
end

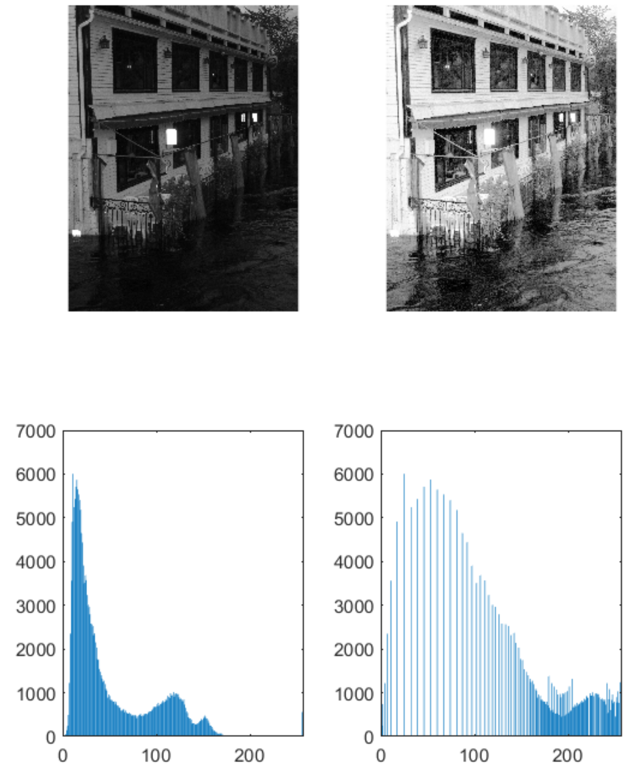

Figure 4: Result image of histogram equalization and the corresponding histogram. left: original image and its histogram, right: histogram-equalized image and its histogram.

like stained. For further improvement, additional contrast limitation, histogram clipping, is added on the adaptive histogram equalization(CLAHE, Contrast Limited AHE), implemented on HistEq_v3.m.

Histogram clipping is done as follows: For given histogram within a tiled segment, clip the histogram on certain limit and re-distribute the remaining intensities across the whole intensity range. This limits maximum contrast amplification, thus reducing the boost effect of noises.

From the fourth image in Figure5, it can be seen that histogram clipping limits constrast so that the overall image becomes more natural on one hand and details being boosted on the other.

Problem 5: Histogram Matching

hw1_5.m loads input.jpg and performs histogram matching given target histogram from input_match.jpg. Histogram matching is done by applying inverse histogram equalization of input_match.jpg at histogram-equalized image from input.jpg. Details can be found in Algorithm5.

When building inverse-LUT from targetting image, some of the intensities may be undefined because LUT is not always one-to-one. In that case, interpolation can be used to predict the missing value.

Result can be found in Figure6. The processed image seems to follow intensity distribution of target image(middle). Also, the historgam in Figure6 convinces the matching be done properly, as the output histogram resembles to the target histogram.

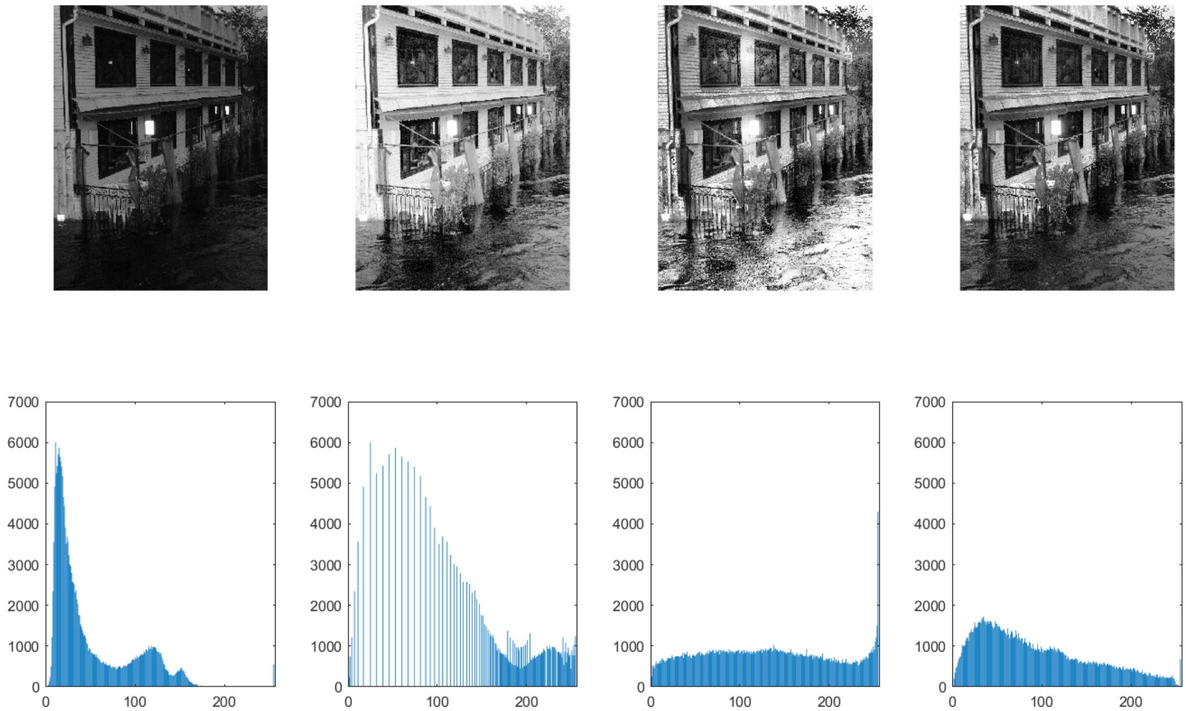

Figure 5: Results of several histogram equalizations and their histograms. first: original image. second: vanilla histogram equalization. third: adaptive histogram equalization. fourth: contrast limited adaptive histogram equalization.

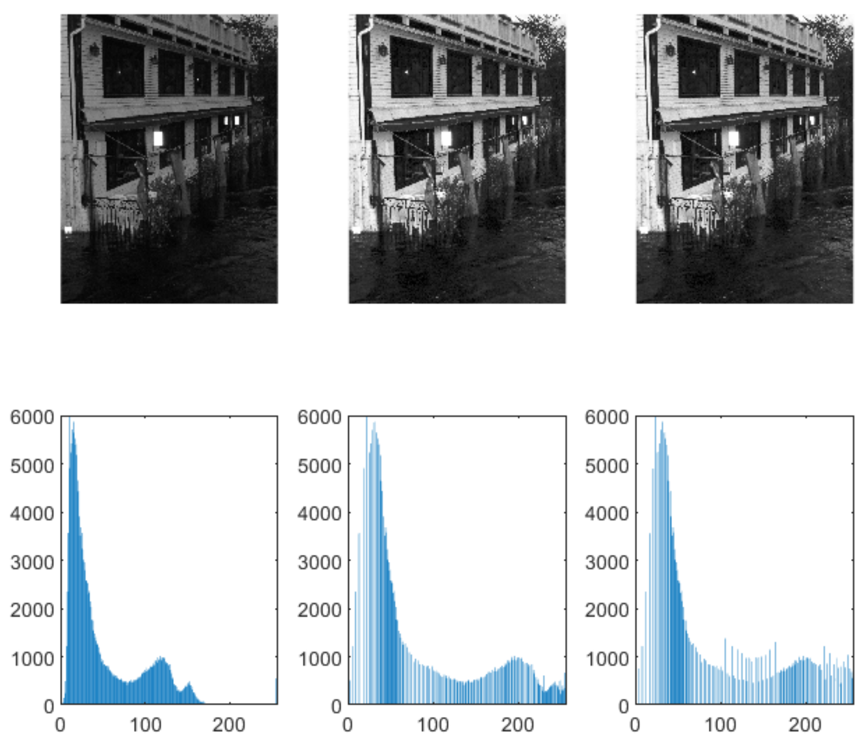

Figure 6: Resulting image of histogram matching and its corresponding histogram. left: original image and its histogram. middle: targetting image and its histogram (target histogram). third: result of applying histogram matching on the image and its histogram.