DIP Homework 2

Qiuyi Zhang 12330402
joyeec9h3@gmail.com

November 4, 2014

Contents

1 Exercises **1**
1.1 Histogram Equalization 1
1.2 Spatial Filtering 2

2 Programming Tasks **2**
2.1 Histogram Equalization 2
2.1.1 Results 2
2.1.2 Analysis 3
2.1.3 Algorithm 4
2.2 Image Patch Extraction 4
2.2.1 Results 4
2.3 Spatial Filtering 5
2.3.1 Results 5
2.3.2 Applications of different filters 8
2.3.3 Algorithm 9

1 Exercises

1.1 Histogram Equalization

Answer: Let MN be the total number of pixels, n_{r_j} be the number of pixels in the original image with intensity value r_j, L be the intensity levels of the image. From eq. (3.3-8) in the textbook, the histogram equalization transformation would be:

$$s_k = \frac{(L-1)}{MN} \sum_{r_j=0}^{r_k} n_{r_j}$$

Hence s_k is monotonically increasing, therefore $n_{s_k} = n_{r_k}$. The second pass of histogram equalization would produce:

$$v_k = \frac{(L-1)}{MN} \sum_{s_j=0}^{s_k} n_{s_j}$$

Because $n_{s_k} = n_{r_k}$,

$$v_k = \frac{(L-1)}{MN} \sum_{r_j=0}^{r_k} n_{r_j} = s_k$$

This shows that a second pass of histogram equalization produces exactly the same result as the first pass.

1.2 Spatial Filtering

Answer: The merging of the bars in Fig. 1(c) is caused by the width of the filter matching the width of a bar plus the distance between the bars. When the filter is shifted between the bars, it will always cover exactly the same group of pixels that make up one bar, so the numerator for the averaging operator will always be the same within this area. Because the denominator is a constant for a given averaging filter, it will produce the same response across the area containing the bars, therefore the bars seem to be merged. But in Fig. 1(b) and Fig. 1(d), the width of the filter is smaller/larger than, or not a multiple of, the width of a bar plus the distance between the bars. Therefore there are no merging effects in these two figures.

2 Programming Tasks

2.1 Histogram Equalization

2.1.1 Results

Figure 1: The original image

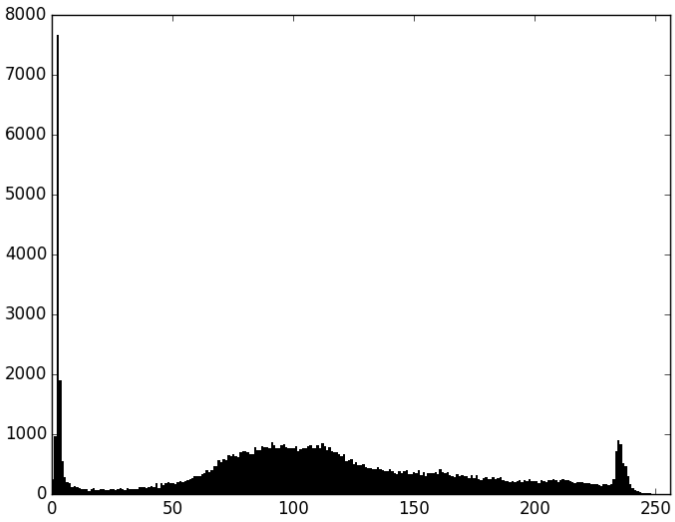

Figure 2: Histogram of the original image

Figure 3: Histogram-equalized result

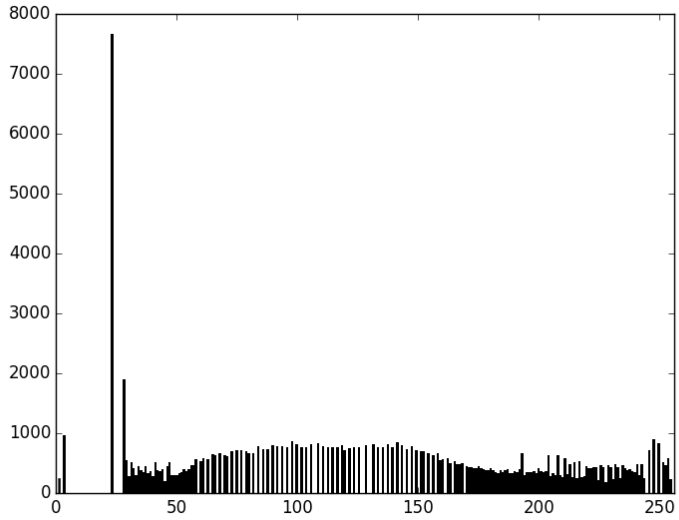

Figure 4: Histogram of the histogram-equalized result

2.1.2 Analysis

After histogram equalization, the image covers a wider range of intensity scale, therefore the result has a higher contrast. The snow on the mountain is whiter and the shades has more layers after being processed.

The equalized histogram clearly has a more uniformed plot, though inevitably there are some extreme values that still stand out after equalization, like the dark regions at the bottom of the image which results in the high bars in the left of the plot.

2.1.3 Algorithm

Algorithm 1 Histogram Equalization

function EQUALIZE_HIST(*input_img*)
  L = intensity levels of the *input_img*
  MN = number of pixels in *input_img*
  for $i = 0 \rightarrow L - 1$ **do**
    n_i = number of pixels in *input_img* with intensity level i
    $s_i = 0$
    for $j = 0 \rightarrow i$ **do**
      $s_i += n_j$
    end for
    $s_i = s_i \times \frac{(L-1)}{MN}$
  end for
  for each pixel p in *input_img* **do**
    i = the intensity of p
    Put a pixel with intensity s_i into *output_img*
  end for
  return *output_img*
end function

2.2 Image Patch Extraction

2.2.1 Results

Figure 5: The original image

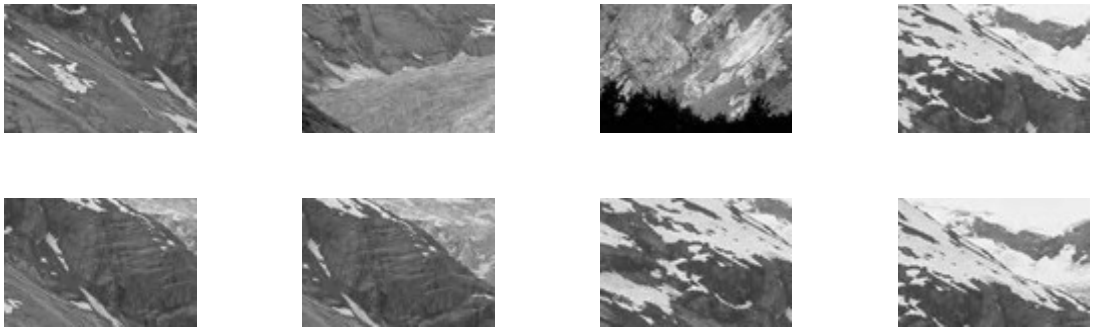

Figure 6: 96 × 64 patches

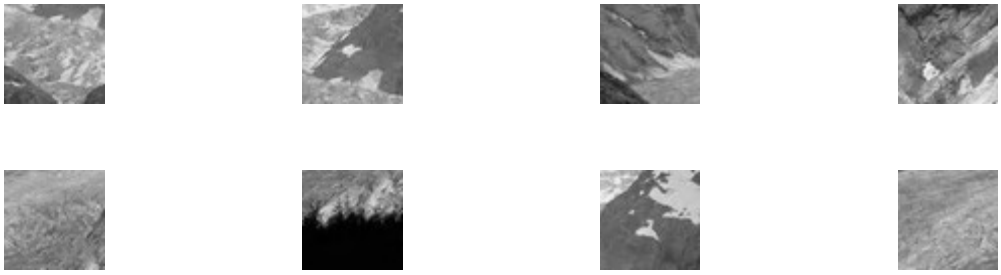

Figure 7: 50×50 patches

2.3 Spatial Filtering

2.3.1 Results

Figure 8: The original image

Figure 9: Smoothed image with 3×3 averaging filter

Figure 10: Smoothed image with 7×7 averaging filter

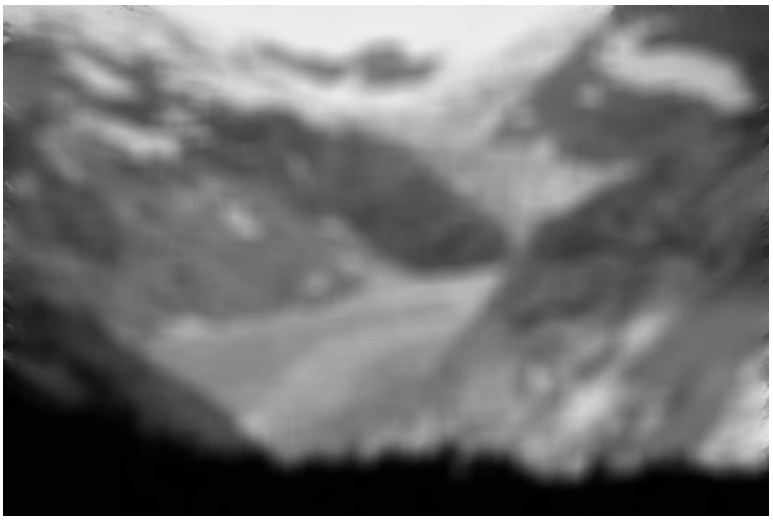

Figure 11: Smoothed image with 11×11 averaging filter

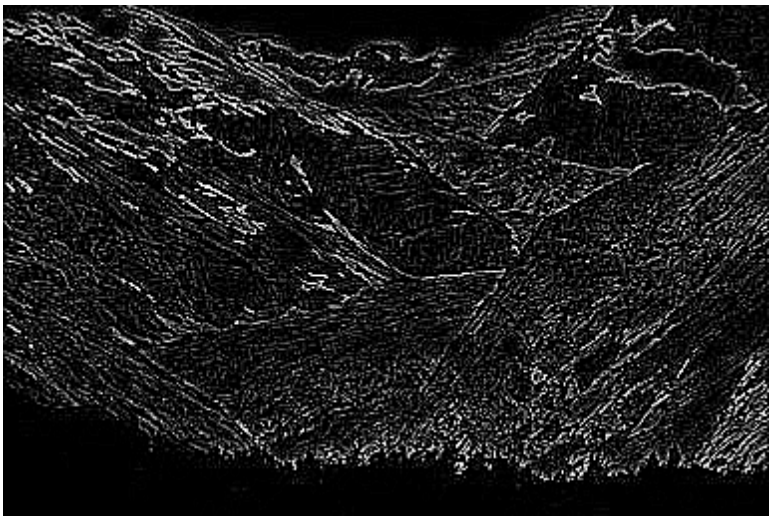

Figure 12: Image filtered with 3×3 Laplacian filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 13: Laplacian filter

Note

To obtain a sharpened image with the result of a laplacian filter, we can combine the original image with the filtered result. Because the laplacian filter used here has a positive center, the sharpened image can be produced by:

$$g(x, y) = f(x, y) + laplacian(x, y)$$

where $g(x, y)$, $f(x, y)$, and $laplacian(x, y)$ are intensity values at the pixels of coordinate (x, y) in the sharpened image, original image, and the laplacian filtered image, respectively.

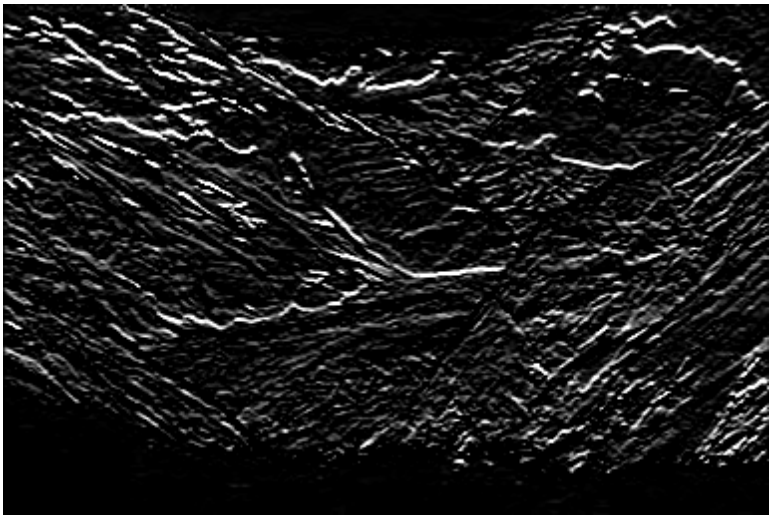

Figure 14: Image filtered with 3×3 horizontal Sobel filter

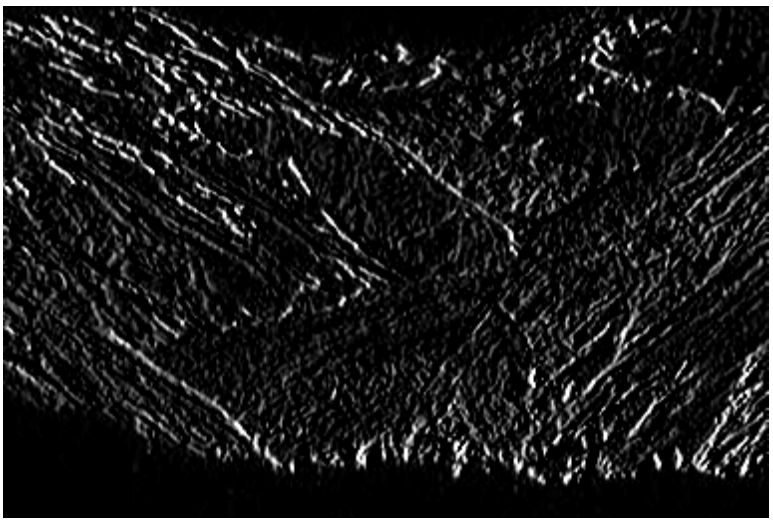

Figure 15: Image filtered with 3×3 vertical Sobel filter

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(a) 3×3 horizontal Sobel filter (b) 3×3 vertical Sobel filter

Figure 16: Sobel filters

Note

The two filtered result contain the vertical and horizontal derivatives of the original image respectively. To obtain the magnitude of gradient from these two partial derivatives, we can calculate the square root of the sum of the squares:

$$M(x, y) = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}$$

or approximate the value by calculating:

$$M(x, y) \approx |g_x(x, y)| + |g_y(x, y)|$$

where $M(x, y)$, $g_x(x, y)$, and $g_y(x, y)$ are intensity values at the pixels of coordinate (x, y) in the gradient image, horizontally filtered image, and the vertically filtered image, respectively.

To obtain a sharpened image with the gradient, we can calculate:

$$g(x, y) = f(x, y) + M(x, y)$$

where $g(x, y)$, $f(x, y)$, and $M(x, y)$ are intensity values at the pixels of coordinate (x, y) in the sharpened image, original image, and the gradient image, respectively.

2.3.2 Applications of different filters

Averaging filter

- Smooth or blur the image.
- Reduce noise.
- Combined with thresholding function to obtain a gross representation of an image.

Median filter

- Smooth or blur the image.
- Reduce noise, especially *salt-and-pepper* noise.

Max filter

- Find the brightest points in an image.

Min filter

- Find the darkest points in an image.

Laplacian filter

- Detect edges, corners, features, blobs in an image.
- Estimate motions in images.

Sobel filter

- Detect edges, corners, features, blobs in an image.

2.3.3 Algorithm

Note

The textbook uses zero padding for the example of correlation, but it also mentions that duplicates or mirror values can be used for padding (see side notes on P169). Here I use the duplicate of the neighborhood center for padding when dealing with borders because it produces a more natural result.

Algorithm 2 Filter

function FILTER2D($input\_img, filter$)
    N = width of $input\_img$, M = height of $input\_img$
    n = width of $filter$, m = height of $filter$
    $a = \lfloor \frac{m}{2} \rfloor$, $b = \lfloor \frac{n}{2} \rfloor$
    w = the $filter$ flattened to 1-d
    for each pixel p in the $input\_img$ **do**
        z = a $m \times n$ matrix filled with intensity of p
        Fill z with the $m \times n$ neighborhood of p. If there are pixels out of borders, leave it
        $i = w^T z$
        Put a pixel with intensity i in the corresponding position of $output\_img$
    end for
    return $output\_img$
end function