3. The histogram of image intensity levels

3.1. Introduction

This laboratory work presents the concept of image histogram together with an algorithm for dividing the image histogram into multiple bins and reducing the number of image gray levels (gray levels quantization).

3.2. The histogram of intensity levels

Given a grayscale image with the highest intensity value L (for an image with 8 bits/pixel $L=255$), the intensity (gray) level histogram is defined as a function $h(g)$ that is equal to as value the number of pixels in the image (or in the region of interest) that have intensity equal to g, for each intensity level $g \in [0 \ldots L]$.

$$h(g) = N_g \tag{3.1}$$

N_g – the number of pixels in the image or in the region of interest that have the intensity equal to g.

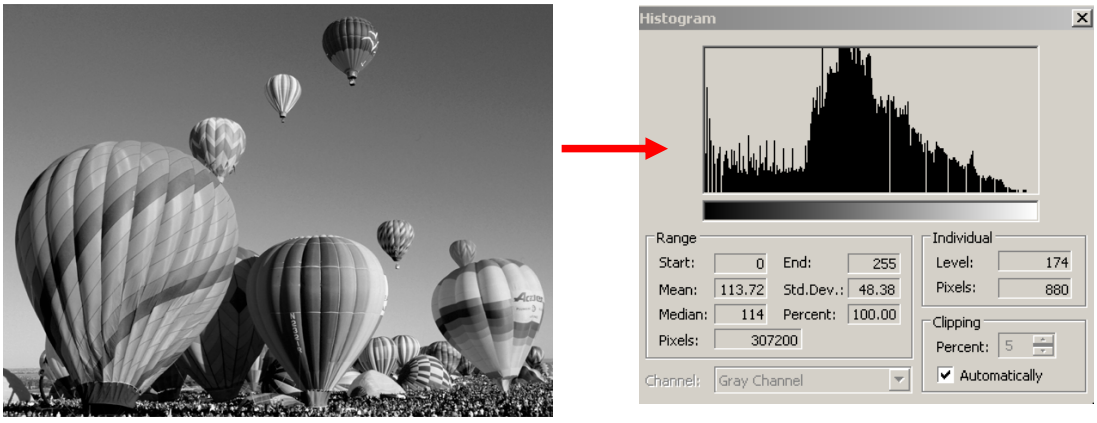


Fig. 3.1 Example: the histogram of a grayscale image

The function obtained by normalizing the histogram with the number of pixels in the image (in the ROI) is called the probability density function (PDF) of the intensity levels.

$$p(g) = \frac{h(g)}{M} \tag{3.2}$$

Where:

$M = image\_height \times image\_width$

PDF has the following properties:

$$\begin{cases} p(g) >= 0 \\ \int_{-\infty}^{\infty} p(g)dg = 1, \quad \sum_{g=0}^{L} \frac{h(g)}{M} = \frac{M}{M} = 1 \end{cases} \tag{3.3}$$

3.3. Application: Multilevel thresholding

In the following we describe an algorithm which determines multiple thresholds for reducing the number of image intensity (gray) levels.

Its first step is to determine the local maxima of the histogram. Then, each gray level is assigned to the closest maximum.

The following steps must be performed in order to determine the histogram maxima:

1. Normalize the histogram (transform it into a PDF)
2. Choose a window width *2*WH+1* (a good value for WH is 5)
3. Choose a threshold *TH* (a good value is 0.0003)
4. For each position (middle of the window) *k* from *0+WH* to *255-WH*
   - Compute the average *v* of normalized histogram values in the interval [*k-WH*, *k+WH*]. Remark: the value *v* is the average of *2*WH+1* values
   - If *PDF*[*k*]>*v+TH* and *PDF*[*k*] is greater or equal than all *PDF* values in the interval [*k-WH*, *k+WH*] then *k* corresponds to a histogram maximum. Store it and then continue from the next position.
5. Insert 0 at the beginning of the maxima position list and 255 at the end (this allows the colors black and white to be represented exactly).

The second step is thresholding. Thresholds are located at equal distances between the maxima. Therefore the algorithm for thresholding is simply to assign to each pixel the color value of the nearest histogram maximum.

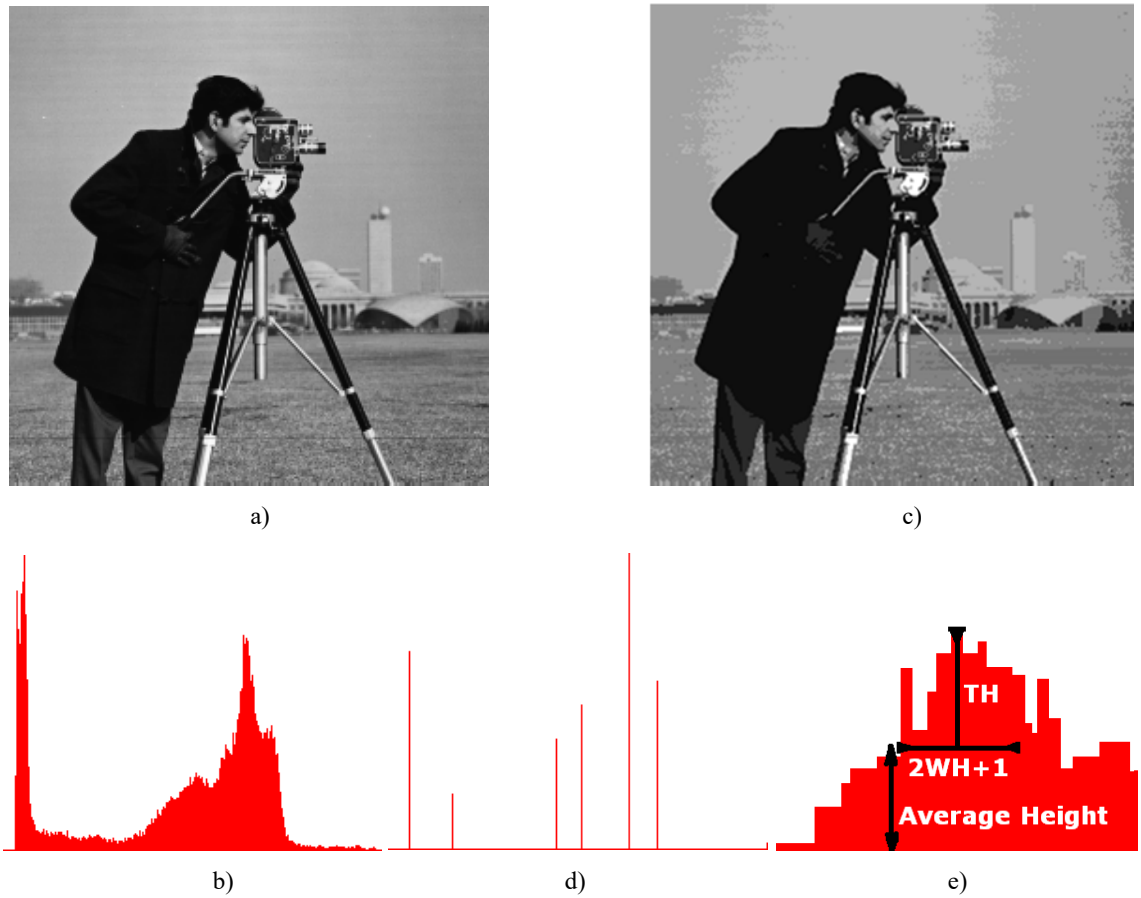


Fig. 3.2 a) The initial image; b) The histogram of the initial image; c) The obtained multilevel thresholded image; d) The histogram of the multilevel thresholded image; e) The histogram maxima computation algorithm

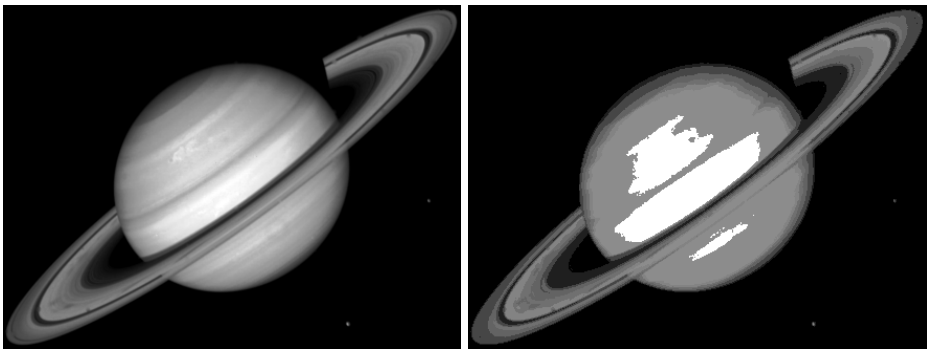
3.4. Floyd-Steinberg dithering

As seen in Fig. 3.3b, the results are visually unacceptable when the number of gray levels is small. To correct this, a dithering algorithm can be applied. Such an algorithm spreads the quantization error to multiple pixels. An example of a dithering algorithm is the Floyd-Steinberg algorithm:

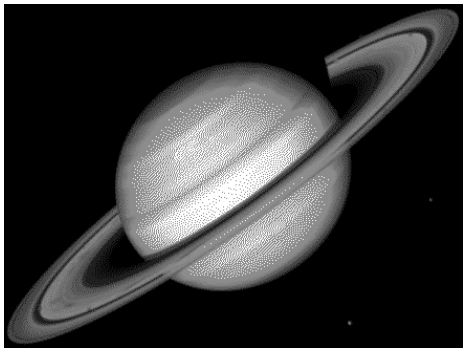
```
for i from [0, rows-1]
   for j from [0, cols-1]
      oldpixel := img(i,j)
      newpixel := find_closest_histogram_maximum(oldpixel)
      img(i,j) := newpixel
      error := oldpixel - newpixel
      if (i,j+1) inside img then
         img(i,j+1)   := img(i,j+1)   + 7*error/16
      if (i+1,j-1) inside img then
         img(i+1,j-1) := img(i+1,j-1) + 3*error/16
      if (i+1,j) inside img then
         img(i+1,j)   := img(i+1,j)   + 5*error/16
      if (i+1,j+1) inside img then
         img(i+1,j+1) := img(i+1,j+1) + error/16
```

This algorithm computes the quantization error and spreads it to the neighboring pixels according to the following fraction matrix (**X** = current pixel's location):

0	0	0
0	**X**	7/16
3/16	5/16	1/16



a) b)



c)

Fig. 3.3 a) The initial image; b) The obtained multilevel thresholded image; c) Dithering on the initial image using the Floyd-Steinberg algorithm

3.5. Implementation details

3.5.1. Displaying the histogram as an image

The histogram can be viewed as an image by making a bar plot. For each gray level draw a bar with height proportional to the number of appearances. The function below **`showHistogram`** plots a histogram (available in the **`OpenCVApplication`** framework). You need to provide the computed histogram, the number of bins, and the height of the desired output image. Bars/lines are automatically rescaled to fit the image but they remain proportional to the histogram values.

```
void showHistogram(const string& name, int* hist, const int  hist_cols,
                   const int hist_height) {
     Mat imgHist(hist_height, hist_cols, CV_8UC3, CV_RGB(255, 255, 255));
                                                 // constructs a white image
     //computes histogram maximum
     int max_hist = 0;
     for (int i = 0; i<hist_cols; i++)
     if (hist[i] > max_hist)
          max_hist = hist[i];
     double scale = 1.0;
     scale = (double)hist_height / max_hist;
     int baseline = hist_height - 1;
     for (int x = 0; x < hist_cols; x++) {
          Point p1 = Point(x, baseline);
          Point p2 = Point(x, baseline - cvRound(hist[x] * scale));
          line(imgHist, p1, p2, CV_RGB(255, 0, 255)); // histogram bins
                                                      // colored in magenta
     }
     imshow(name, imgHist);
}
```

3.5.2. Histogram with custom number of bins

The image histogram can be computed using a custom number of bins $m \leq 256$. This entails dividing the range 0-255 into m equal parts, then counting all the gray levels falling into each of the m bins or buckets. Such a representation is useful since it is lower dimensional.

3.6. Practical work

1. Compute the histogram for a given grayscale image (in an array of integers having dimension 256).
2. Compute the PDF (in an array of floats of dimension 256).
3. Display the computed histogram using the provided function.
4. Compute the histogram for a given number of bins $m \leq 256$.
5. Implement the multilevel thresholding algorithm from section 3.3.
6. Enhance the multilevel thresholding algorithm using the Floyd-Steinberg dithering from section 3.4.
7. Perform multilevel thresholding on a color image by applying the procedure from 3.3 on the Hue channel from the HSV color-space representation of the image. Modify only the Hue values, keeping the S and V channels unchanged or setting them to their maximum possible value. Transform the result back to RGB color-space for viewing.
8. **Save your work. Use the same application in the next laboratories. At the end of the image processing laboratory you should present your own application with the implemented algorithms.**

3.7. Bibliography

[1] R.C.Gonzales, R.E.Woods, *Digital Image Processing. 2-nd Edition*, Prentice Hall, 2002.
[2] Floyd-Steinberg algorithm, http://en.wikipedia.org/wiki/Floyd-Steinberg_dithering