# 3. Histograma nivelurilor de intensitate

## 3.1. Introducere

În această lucrare se vor prezenta conceptul de histogramă a nivelurilor de gri și un algoritm pentru a diviza această histogramă în mai multe zone în scopul reducerii numărului de niveluri de gri (cuantizare în spațiul culorilor).

## 3.2. Histograma nivelurilor de intensitate

Având o imagine în niveluri de gri cu nivelul maxim de intensitate $L$ (pentru o imagine cu 8 biți/pixel $L=255$), histograma nivelurilor de intensitate (gri) se definește ca o funcție $h(g)$ care are ca valoare, numărul de pixeli din imagine (sau dintr-o regiune) care au intensitatea $g \in [0 \ldots L]$ :

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

$N_g$ - numărul de pixeli din imagine sau din regiunea de interes (ROI) care au intensitatea $g$.

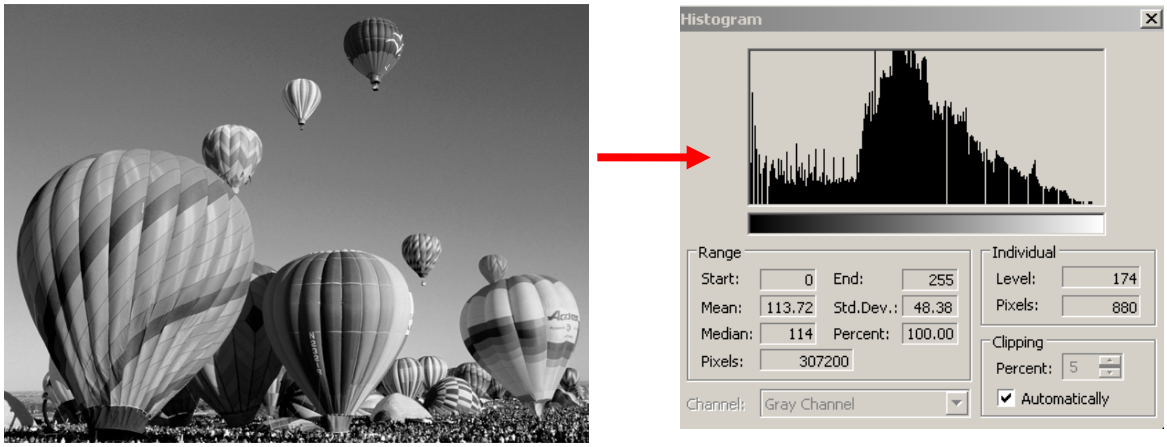



**Fig. 3.1** Exemplu: Histograma unei imagini grayscale

Histograma normalizată cu numărul de pixeli din imagine (din ROI) se numește funcția densității de probabilitate[1] (FDP) a nivelurilor de intensitate:

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

unde:

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

FDP are următoarele proprietăți:

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

[1]Având domeniul de definiție discret în statistică se numește funcție masă de probabilitate. Totuși se folosește termenul de funcție densitate de probabilitate corespunzător domeniului continuu datorită utilizării frecvente în literatura de specialitate.

## 3.3. Aplicație: Determinarea pragurilor multiple

Acest algoritm determină praguri multiple în scopul reducerii numărului de niveluri de intensitate (gri) a unei imagini.

Primul pas constă în determinarea maximelor din histogramă. Apoi fiecare nivel de gri este asignat la cel mai apropiat maxim.

Pentru determinarea maximelor din histogramă se urmăresc pașii:

1. Se normalizează histograma (se transformă într-o FDP)
2. Se alege o fereastră de lățime *2*WH+1* (o valoare bună pentru WH este 5)
3. Se alege un prag *TH* (o valoare bună este 0.0003)
4. Se „suprapune" fereastra peste histogramă. Pentru fiecare poziție *k* (a mijlocului ferestrei) de la 0+*WH* până la *255-WH*
   - Se calculează media *v* a valorilor din histograma normalizată în intervalul [*k-WH*, *k+WH*]. Obs: valoarea *v* este media a *2*WH+1* valori
   - Dacă *FDP*[*k*]>*v+TH* şi *FDP*[*k*] este mai mare sau egal cu toate valorile *FDP* din intervalul [*k-WH*, *k+WH*] atunci *k* corespunde la un maxim din histogramă. Se memorează și se continuă din poziția următoare.
5. Se inserează 0 la începutul listei de poziții de maxime și 255 la sfârșit. (aceasta permite culorilor negru respectiv alb să fie reprezentate exact).

Al doilea pas constă în determinarea efectivă a pragurilor. Acestea sunt situate la distanțe egale între maximele determinate anterior. Algoritmul de reducere a nivelurilor de gri este unul simplu: se asignează fiecare pixel la valoarea culorii celui mai apropiat maxim din histogramă.

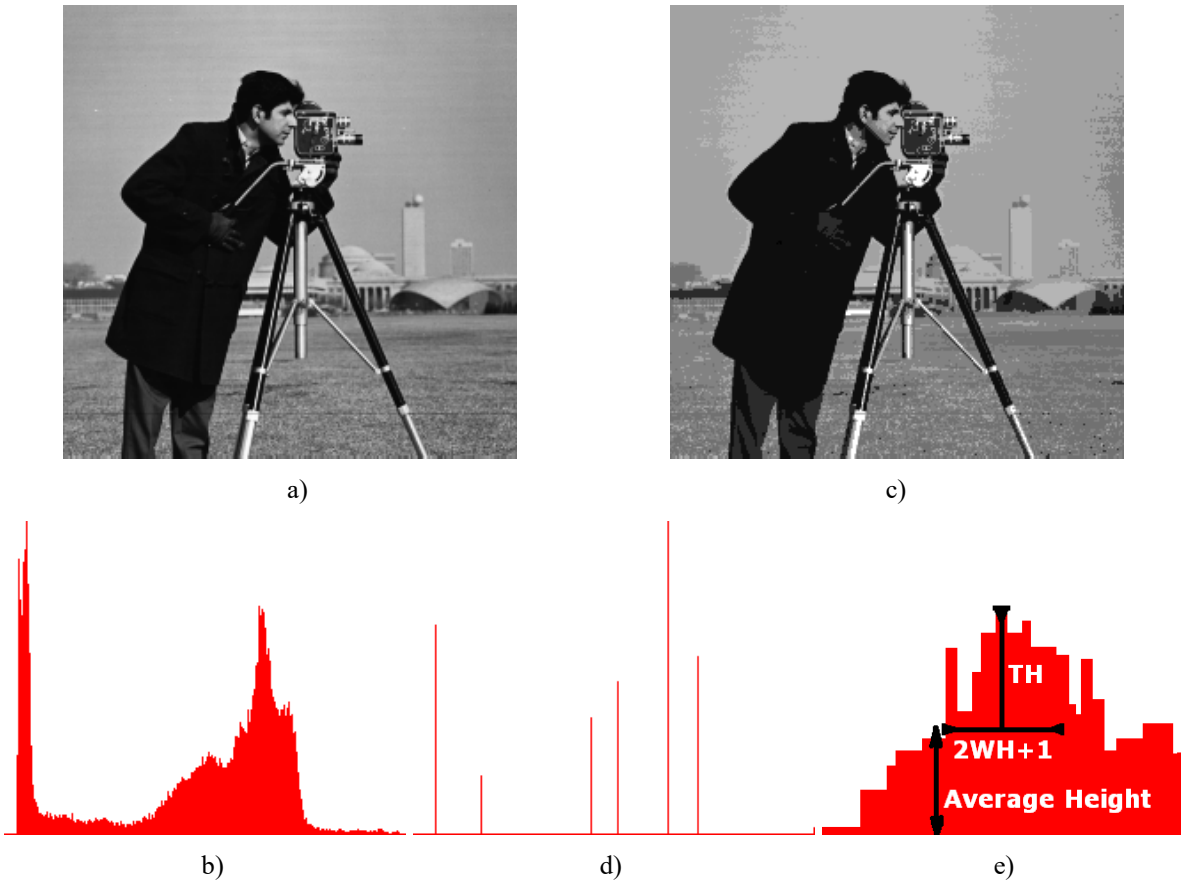



**Fig. 3.2** a) Imaginea inițială; b) Histograma imaginii inițiale; c) Imaginea cuantizată obținută (praguri multiple); d) Histograma imaginii cuantizate; e) Algoritmul de calcul al maximelor din histogramă

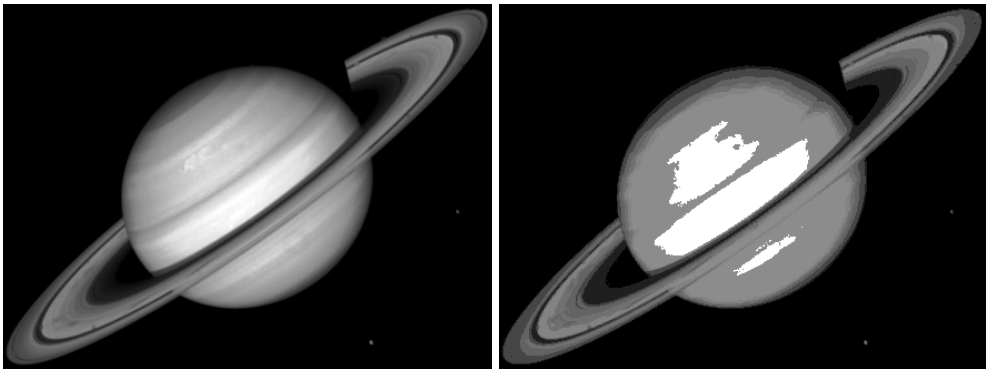
## 3.4. Algoritmul de corecție Floyd-Steinberg

După cum se observă în Fig. 3.3b, rezultatele sunt vizual inacceptabile când numărul de niveluri de gri este scăzut. Pentru a obține rezultate vizuale acceptabile se poate aplica un algoritm de distribuire a erorii de cuantizare pe mai mulți pixeli (*dithering*). Un exemplu de astfel de algoritm este Floyd-Steinberg:

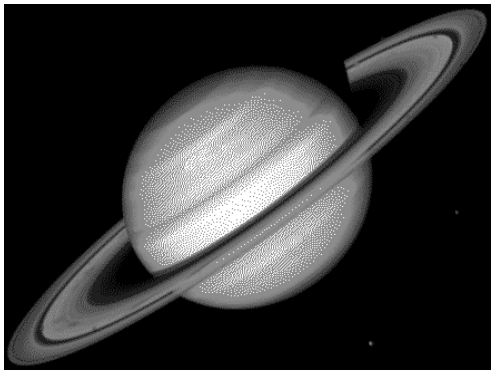
```
for i from [0, rows-1]
   for j from [0, cols-1]
      oldpixel := img(i,j)
      newpixel := find_closest_histogram_maximum(oldpixel)
      img(i,j) := newpixel
      error := oldpixel - newpixel
      if (i,j+1) inside img then
         img(i,j+1)   := img(i,j+1)   + 7*error/16
      if (i+1,j-1)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

Acest algoritm calculează eroarea de cuantificare și o distribuie la pixelii vecini. Fracțiunile de distribuție a erorii sunt prezentate în matricea următoare (**X** = locația pixelului curent):

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



a) b)



c)

**Fig. 3.3** a) Imaginea inițială; b) Imaginea cuantizată obținută (praguri multiple); c) Împrăștierea erorii pragurilor multiple utilizând algoritmul Floyd-Steinberg

## 3.5. Detalii de implementare

### 3.5.1. Afișarea histogramei ca o imagine

Histograma poate fi afișată sub forma unui grafic cu bare verticale. Pentru fiecare nivel de gri se desenează o bară verticală proporțională cu numărul de apariții al acestuia în imagine. Funcția **showHistogram** de mai jos realizează această operație (disponibilă în aplicația **OpenCVApplication**). Trebuie să furnizați titlul ferestrei de afișare, histograma calculată, numărul de acumulatoare, dimensiunea dorită pentru înălțimea imaginii de ieșire. Barele sunt redimensionate automat astfel încât să încapă în imagine și totodată rămân proporționale cu valorile din histogramă.

```
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. Histograma cu un număr redus de acumulatoare (*bins*)

Histograma se poate calcula și dacă folosim un număr redus de acumulatoare $m \leq 256$. Pentru aceasta trebuie să împărțim intervalul [0,255] în $m$ părți egale și apoi să numărăm pixelii care aparțin fiecărei părți. Această reprezentare este utilă fiindcă are dimensionalitate redusă.

## 3.6. Activități practice

1. Calculați histograma (într-un vector de întregi de dimensiune 256) pentru o imagine grayscale (cu 8 biți/pixel).
2. Calculați FDP (într-un vector de tip float de dimensiune 256).
3. Afișați histograma calculată utilizând funcția din laborator.
4. Calculați histograma folosind un număr redus de $m \leq 256$ acumulatoare.
5. Implementați algoritmul de reducere a nivelurilor de gri (praguri multiple) de la 3.3.
6. Îmbunătățiți algoritmul de reducere a nivelurilor de gri (praguri multiple) utilizând distribuirea erorii cu algoritmul Floyd-Steinberg de la 3.4.
7. Realizați algoritmul de reducere a nivelurilor de gri pe canalul Hue din spațiul de culoare HSV al unei imagini color. Modificați doar valorile din canalul H, păstrând canalele S și V neschimbate. O altă opțiune este setarea lor (S și V) la valoarea maximă permisă. Pentru vizualizare transformați înapoi în spațiul RGB.
8. **Salvați-vă ceea ce ați lucrat. Utilizați aceeași aplicație în laboratoarele viitoare. La sfârșitul laboratorului de procesare a imaginilor va trebui să prezentați propria aplicație cu algoritmii implementați!!!**

## 3.7. **Referințe**

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