2. Manipularea imaginilor

Reprezentarea imaginilor
Accesarea pixelilor unei imagini
Conversia imaginilor şi spatii de culoare (Gri, RGB, HSV)
Filtrarea spatială a imaginilor (filtrul median, filtrul Gaussian)

În această aplicaţie se vor prezenta diferite forme de reprezentare a imaginilor, modalităţi de reprezentare a culorilor, cât şi două metode de filtrare spațială a imaginilor, mai exact filtrul median şi filtrul Gaussian.

2.1 Baze teoretice

2.1.1 Reprezentarea imaginilor

O imagine digitală poate fi reprezentată printr-o funcţie $f(x, y)$, unde parametrii funcţiei reprezintă coordonatele imaginii pe cele două direcţii ale lui $\Re^2$, aşa cum se poate vedea şi în Fig. 2.1. O formă compactă de scriere a unei imagini digitale monocrome este dată sub forma unei matrici ale cărei elemente sunt denumiți *pixeli* [**?**]:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}, \tag{2.1}$$

unde valorile pe care le poate lua $f(x, y)$ aparţin unui domeniu finit:

$$0 \leq f(x,y) \leq L-1. \tag{2.2}$$

Valoarea lui L este, în general, 256 pentru imagini gri. Acest proces de reprezentare a imaginilor, denumit eşantionare, este exemplificat în Fig. 2.2 pentru cazul unei imagini de nivel gri definită pe 8 biţi.

În cazul imaginilor color, $f(x, y)$ va conţine cele trei culori fundamentale (Roşu, Albastru şi Verde), ea putând fi reprezentată matematic în două forme:

- prin intercalarea pixelilor, unde fiecare linie este o matrice 2D, cu fiecare element reprezentând o listă cu trei valori;
- prin intercalarea culorilor, unde pe fiecare linie datele sunt separate în matrici 2D, câte una pentru fiecare canal de culoare:

$$f_{RGB}(x,y) = \left[f_r(x,y), \quad f_g(x,y), \quad f_b(x,y) \right]. \tag{2.3}$$

Pentru a se putea realiza o standardizare a culorilor a fost introdusă noţiunea de *model al culorilor*[**?**]. Un astfel de model poate fi orientat spre hardware (monitoare sau imprimate) sau spre aplicaţii software, unde scopul final este acela de manipulare a culorilor. Cel mai

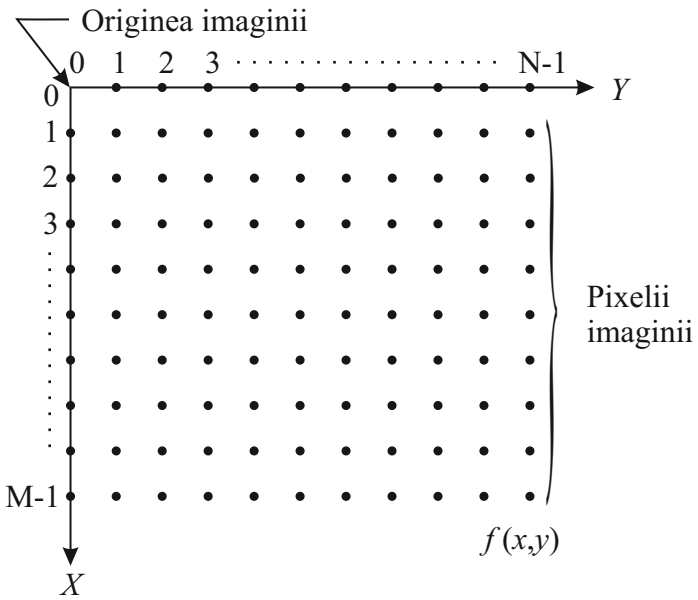

Fig. 2.1 Coordonatele unei imagini digitale $f(x, y)$.

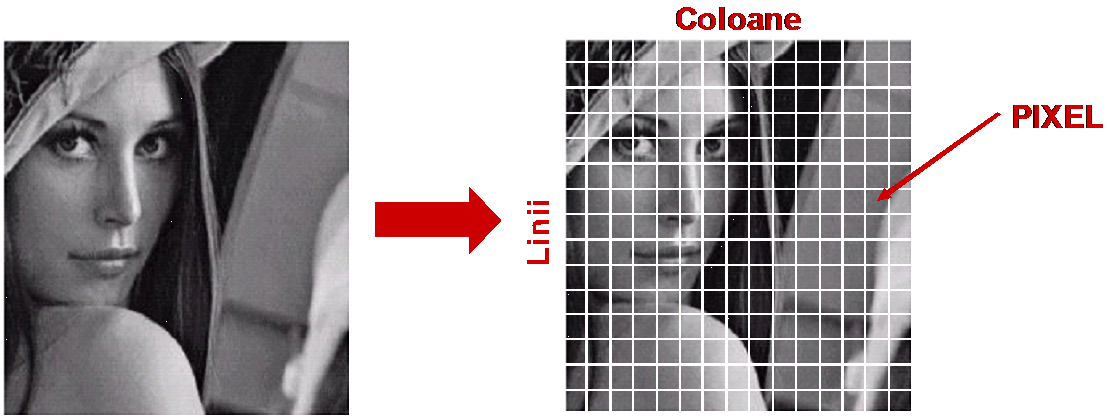

Fig. 2.2 Reprezentarea unei imagini de nivel gri prin eşantionare.

utilizat model al culorilor, din categoria orientată spre hardware, este modelul RGB (*Red, Green, Blue*). Acest model este utilizat în special la monitoarele color dar şi la o clasă largă de camere video. Reprezentarea imaginilor se poate realiza şi prin alte modele de culori, dintre care se pot aminti: HSV (*Hue, Saturation, Value*), CMY (*Cyan, Magenta, Yellow*), sau Lab. Dintre acestea, modelul HSV este un model de reprezentare ce se aseamănă cu modul de vedere uman [**?**], iar modelul CMY este utilizat manipularea datelor în imprimante color.

2.1.2 Filtrarea imaginilor

Filtrarea imaginilor este folosită pentru a se reduce zgomotul din imaginile de intrare, fiind definită în domeniul spaţial ca o operaţie de convoluţie:

$$g(x, y) = f(x, y) * w(x, y), \tag{2.4}$$

unde $f(x, y)$ este imaginea de intrare, $g(x, y)$ imaginea de ieşire, iar w o funcţie fereastră, sau mască, aplicată tuturor pixelilor din imaginea de intrare.

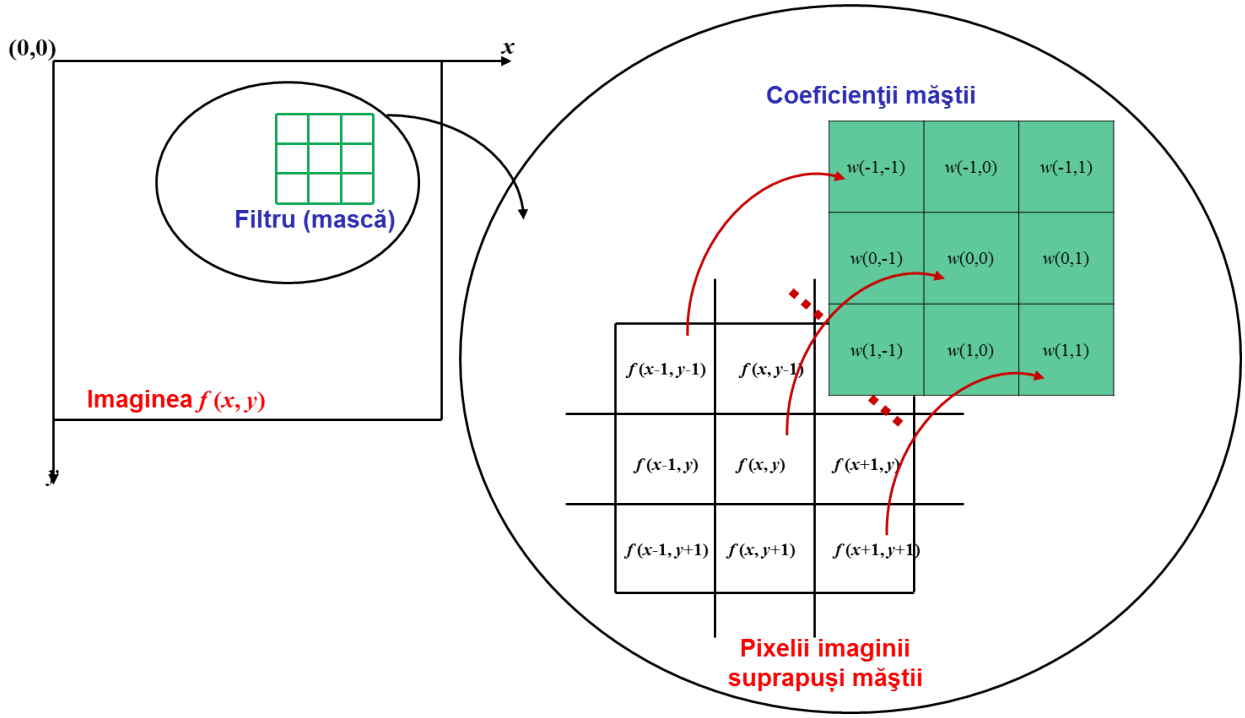

Fig. 2.3 Reprezentarea operației de filtrare spațială a unei imagini.

Filtrul median este reprezentat de acea mască în care toţi coeficienţii sunt egali:

$$w(x,y) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}. \tag{2.5}$$

2.2 Cerinţe

1. Să se încarce şi să se afişeze o imagine color;
2. Să se convertească imaginea color într-o imagine gri;
3. Să se convertească imaginea color în spaţiul de culoare HSV;

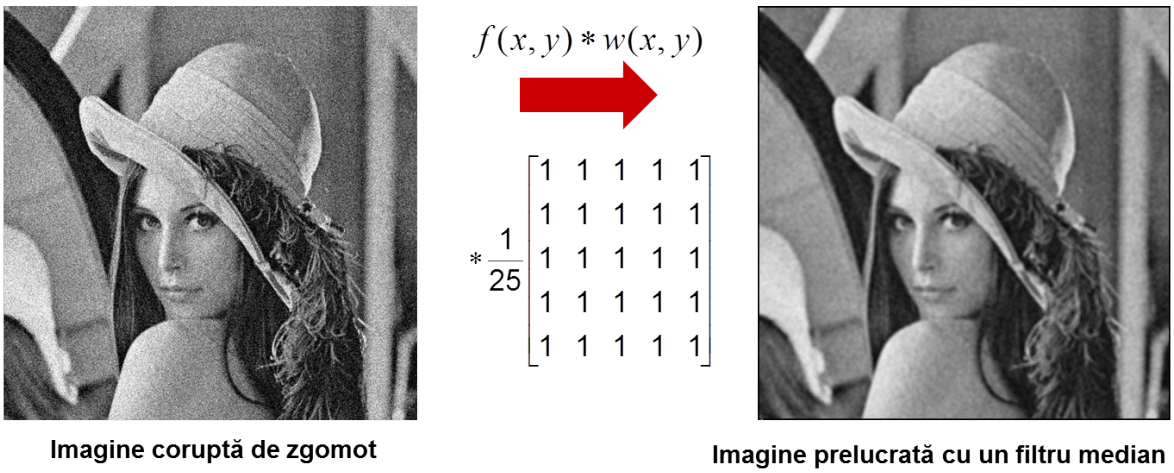

Fig. 2.4 Filtrarea unei imagini utilizând o mască de dimensiune $5x5$.

4. Să se acceseze un pixel, respectiv o regiune, din imagine;
5. Să se aplice un filtru median asupra imaginii.

2.3 Codul sursă al aplicaţiei

```
1  #include <iostream>
2  #include <opencv2/core/core.hpp>
3  #include <opencv2/highgui/highgui.hpp>
4  #include <opencv2/imgproc/imgproc.hpp>
5
6  using namespace cv;
7  using namespace std;
8
9  int main(int argc, char ** argv)
10 {
11   cout << "SVA Laborator 02: Manipularea imaginilor" << endl;
12
13   Mat img_in, img_out;
14   string strImgPath;
15
16   // Incarcarea imaginii de pe disk
17   if (argc == 2)
18     strImgPath = argv[1];
19   else
20     strImgPath = "../pcrai.jpg";
21
22   img_in = imread(strImgPath.c_str(), CV_LOAD_IMAGE_UNCHANGED);
23
24   // Verificarea incarcarii corecte a imaginii
25   if(!img_in.data)
26   {
27     cout << "Imaginea " << strImgPath << " nu a putut fi incarcata" ↩
         << endl;
28     return -1;
29   }
30
31   // Conversia din imagine in culori in imagine gri
32   cvtColor(img_in, img_out, CV_RGB2GRAY);
33   imshow("Culoare -> Gri", img_out);
34   waitKey();
35   destroyAllWindows();
36
37   // Conversia din spatiul de culoare RGB in spatiul HSV
38   cvtColor(img_in, img_out, CV_RGB2HSV);
39   imshow("RGB -> HSV", img_out);
40   waitKey();
41   destroyAllWindows();
42
43   // Accesarea pixelului avand coordonatele (10, 10)
44   Vec3b pxValue = img_in.at<Vec3b>(10,10);
45   cout << "(10,10): R = " <<
46       (int)pxValue[0] << " G = " <<
```

```
47        (int)pxValue[1] << " B = " <<
48        (int)pxValue[2] << endl;
49
50    // Accesarea unei regiuni de interes din imagine
51    for (int i = 100; i < 200; i++)
52    {
53      for (int j = 50; j < 200; j++)
54      {
55        img_in.at<Vec3b>(j,i)[0] = 255;
56        img_in.at<Vec3b>(j,i)[1] = 255;
57        img_in.at<Vec3b>(j,i)[2] = 255;
58      }
59    }
60
61    imshow("Regiune de interes", img_in);
62    waitKey();
63    destroyAllWindows();
64
65    // Aplicarea unui filtru median
66    medianBlur(img_in, img_out, 5);
67    imshow("Imagine originala", img_in);
68    imshow("Imagine filtrata", img_out);
69    waitKey();
70    destroyAllWindows();
71
72    return 0;
73  }
```

2.4 Descrierea funcţiilor principale

```
32  void cvtColor(InputArray src, OutputArray dst, int code)
```

Converteşte o imagine dintr-un spaţiu de culoare în altul.

- `src`: imaginea de intrare;
- `dst`: imaginea de ieşire;
- `code`: codul conversiei de culoare:
  - `CV_BGR2HSV`, `CV_RGB2HSV`, `CV_HSV2BGR`, `CV_HSV2RGB`;
  - `CV_BGR2HLS`, `CV_RGB2HLS`, `CV_HLS2BGR`, `CV_HLS2RGB`;
  - `CV_BGR2Lab`, `CV_RGB2Lab`, `CV_Lab2BGR`, `CV_Lab2RGB`.

```
66  void medianBlur(InputArray src, OutputArray dst, int ksize)
```

Filtrarea unui imagini utilizându-se un filtru median.

- `src`: imaginea de intrare;
- `dst`: imaginea de ieşire;
- `ksize`: dimensiunea ferestrei mască (trebuie sa fie un număr impar mai mare ca 1: 3, 5, 7, etc.).