9. Detectarea feţelor în imagini

Metode de clasificare
Detectarea feţelor utilizând clasificatori în cascadă

În acest laborator se va studia o metodă de detectare a feţelor în imagini gri utilizându-se algoritmi de clasificare a caracteristicilor vizuale.

9.1 Baze teoretice

În lucrare de faţă este utilizată metoda dezvoltată de Viola şi Jones [**?**], ulterior completată de Rainer şi Mazdt [**?**], în care au fost propuse implementări ale unor noi algoritmi pentru detecţia de obiecte în timp real. Avantajul acestei noi modalităţi de detecţie este acela că algoritmii folosiţi au un grad de detecţie foarte ridicat şi pot fi aplicaţi pentru orice tip de obiect. Una dintre aplicaţiile unde pot fi utilizate metodele propuse este aceea a detectării fizionomiei şi caracteristicilor faciale. Metoda pentru detectarea fizionomiei are la bază clasificatorul AdaBoost [**?**] şi caracteristici de tip Haar [**? ?**], fiind cunoscută şi sub denumirea de *clasificator Haar* [**?**].

Algoritmul pentru detecţia de obiecte propus de Viola şi Jones [**?**] are la bază caracteristici, motivaţia alegerii acestora în detrimentul alegerii directe a pixelilor fiind dată de faptul că, pe de o parte, caracteristicile pot fi codificate în aşa fel încât să conţină într-un singur loc mai multe informaţii iar, pe de altă parte, viteza de procesare este superioară utilizării directe a pixelilor. În Fig. 9.1 sunt prezentate caracteristicile utilizate de algoritm pentru detectarea fizionomiei.

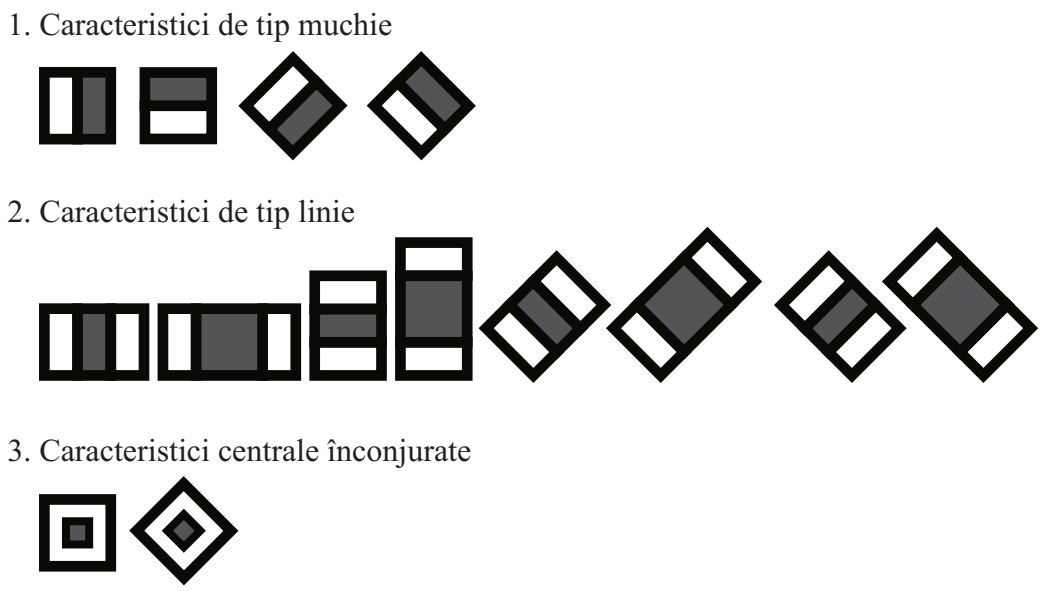

Fig. 9.1 Caracteristici de tip Haar utilizate la detectarea de obiecte [**? ?**].

Caracteristicile utilizate pentru prima dată în [**?**] şi care pot fi folosite la detectarea fizionomiei umane au suferit o serie de modificări. Acest lucru se datorează, în principal, faptului că la prima apariţie a metodei au fost introduse doar caracteristici rectangulare, având o reprezentare de la două până la patru module. Asupra acestei modalităţi de reprezentare iniţiale au apărut modificări în [**?**], lucrare în care s-a eliminat modelul rectangular cu patru module şi a fost definită o nouă modalitate de reprezentare, sub forma unor caracteristici Haar rotite. Introducerea acestei noi reprezentări conduce la o creştere a performanţelor sistemului (şi anume la reducerea numărului de detectări incorecte) cu până la 10%. Astfel, după cum se poate vedea şi în Fig. 9.1, se poate discuta despre 14 prototipuri de caracteristici.

9.2 Cerinţe

1. Să se încarce o imagine color;
2. Să se convertească imaginea color într-o imagine gri;
3. Să se instanțieze și să se încarce un clasificator Haar de detectare a fețelor;
4. Să se detecteze fețele în imaginea gri utilizând clasificatorul Haar;
5. Să se afișeze regiunile fețelor detectate.

9.3 Codul sursă al aplicației

```
1  #include <iostream>
2  #include <stdio.h>
3  #include "opencv2/objdetect/objdetect.hpp"
4  #include "opencv2/highgui/highgui.hpp"
5  #include "opencv2/imgproc/imgproc.hpp"
6
7  using namespace cv;
8  using namespace std;
9
10 int main( void )
11 {
12   cout << "SVA Laborator 09: Detectarea fetelor in imagini" << endl;
13
14   // Incarcare imagine
15   Mat img_in = imread("imagine.jpg");
16
17   if ( img_in.empty() )
18   {
19     cout << "Imaginea nu a putut fi incarcata" << endl;
20     return false;
21   }
22
23   // Definitia si initializarea clasificatorului
24   CascadeClassifier clasificator_haar;
25
26   if ( !clasificator_haar.load("haarcascade_frontalface_alt.xml") )
27   {
28     cout << "Clasificatorul nu a putut fi incarcat...!" << endl;
29     return false;
30   }
31
```

```
32    // Conversia imaginii color de intrare in imagine gri
33    Mat img_gri;
34    cvtColor( img_in, img_gri, CV_BGR2GRAY );
35
36    // Imbunatatirea imaginii prin egalizarea histogramei
37    equalizeHist( img_gri, img_gri );
38
39    // Definitia unui vector de stocare a regiunilor fetei
40    vector<Rect> regiuni_fete;
41
42    // Detectarea fetelor utilizand clasificatorul Haar
43    clasificator_haar.detectMultiScale( img_gri, regiuni_fete, 1.1, 2, ↩
          CV_HAAR_SCALE_IMAGE, Size(40, 40) );
44
45    // Desenarea regiunilor fetei
46    for ( unsigned int i = 0; i < regiuni_fete.size(); i++ )
47      rectangle(img_in, regiuni_fete.at(i), Scalar(100, 255, 150), 2);
48
49    // Afisare rezultate
50    imshow("Detectia fetelor", img_in);
51    waitKey();
52
53    return true;
54  }
```

9.4 Descriere codului sursă

```
24  CascadeClassifier
```

Definirea unui obiect de tipul *CascadeClassifier*.

```
26  bool CascadeClassifier::load(const string& filename)
```

Acest apel este utilizat pentru a se încărca clasificatorul din fişierul *.xml.

```
43  void CascadeClassifier::detectMultiScale(const Mat& image, vector<↩
        Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int ↩
        flags=0, Size minSize=Size(), Size maxSize=Size())
```

Realizează detectarea în imaginea de intrare a tuturor obiectelor (având dimensiuni variate) pentru care a fost clasificatorul antrenat. Obiectele detectate sunt returnate sub forma unui vector cu dreptunghiuri (Rect).

- `image`: imaginea de intrare;
- `objects`: vectorul unde vor fi salvate obiectele (dreptunghiurile) detectate;
- `scaleFactor`: acest parametru specifică cât de mult este redusă dimensiunea imagini, pentru fiecare etapă de **scalare a imagini**. Alegerea unei valori mai mari are drept rezultat reducerea timpului de calcul şi scăderea numărului de ipoteze detectate (dacă factorul de scalare nu se încadrează cu anumite obiecte având o dimensiune precisă);
- `minNeighbors`: indică numărul de vecini pe care trebuie să îi aibă fiecare ipoteză pentru a fi considerată obiectul căutat. Acest parametru este util pentru că pe parcursul procesului de căutare pot apărea mai multe ipoteze într-o aceeaşi regiune (pentru factori de scalare diferiţi). Setarea acestui parametru reprezintă numărul de suprapuneri posibile pentru ca o ipoteză să fie considerată obiect.
- `flags`: poate avea patru combinaţii posibile, acestea putând fi combinate utilizând operaţia de SAU logic:

  - `CV_HAAR_DO_CANNY_PRUNING` – alegerea acestei valori pentru parametru considerat conduce la rejectarea de către clasificator a regiunilor plate (fără linii);
  - `CV_HAAR_SCALE_IMAGE` – pentru această valoare algoritmul de detecţie v-a realiza scalarea imagini, ci nu detectorul propriu-zis;
  - `CV_HAAR_FIND_BIGGEST_OBJECT` – setarea la această valoare conduce la returnarea de către algoritm a obiectelor de dimensiuni mari (numărul de obiecte returnate va fi unu sau nici unu);
  - `CV_HAAR_DO_ROUGH_SEARCH` alegerea acestei valori face ca algoritmul de căutare să finalizeze procesul de căutare, indiferent de factorul de scalare la care este detectat obiectul.
- `minSize`: dimensiunea minimă pe care o poate avea un obiect ce urmează a fi căutat. Obiectele cu o dimensiune mai mica decât aceasta sunt ignorate;
- `maxSize`: dimensiunea maximă pe care o poate avea un obiect ce urmează a fi căutat. Obiectele cu o dimensiune mai mare decât aceasta sunt ignorate;