

8. Detectia fețelor si a componentelor faciale

Scop: Scopul acestei lucrari este de a integra metoda Viola&Jones [1] de detectie a fețelor si a componentelor faciale. Metoda se bazeaza pe detectia de trasaturi de tip Haar calculate pe sub-regiuni din imagine folosind imaginea integrala (vezi cursul 7-8), si identificarea prezentei fețelor umane cu ajutorul unui clasificator de tip cascada. Abordarea propusa de autori ofera capabilitati de timp real si versatilitate pentru detectia oricaror tipuri de obiect pentru care s-a facut o antrenare prealabila a clasificatorului cascada.

8.1. Modele de clasificare de obiecte bazate pe trasaturi Haar

In OpenCV sunt disponibile modelele obtinute prin antrenarea clasificatorilor bazati pe trasaturi Haar in format *.xml. pentru diverse tipuri de obiecte, dar lista nu este exhaustiva putand fi extinsa de catre utilizatori. Modelele clasificatorilor cascada bazati pe trasaturi Haar sunt disponibile in instalarea OpenCV in locatia *OpenCV\sources\data\haarcascades* si sunt urmatoarele:

```
haarcascade_eye.xml
haarcascade_eye_tree_eyeglasses.xml
haarcascade_frontalface_alt.xml
haarcascade_frontalface_alt_tree.xml
haarcascade_frontalface_alt2.xml
haarcascade_frontalface_default.xml
haarcascade_fullbody.xml
haarcascade_lefteye_2splits.xml
haarcascade_lowerbody.xml
haarcascade_mcs_eyepair_big.xml
haarcascade_mcs_eyepair_small.xml
haarcascade_mcs_leftear.xml
haarcascade_mcs_lefteye.xml
haarcascade_mcs_mouth.xml
haarcascade_mcs_nose.xml
haarcascade_mcs_rightear.xml
haarcascade_mcs_righteye.xml
haarcascade_mcs_upperbody.xml
haarcascade_profileface.xml
haarcascade_righteye_2splits.xml
haarcascade_upperbody.xml
```

Exista si modele pt. clasificatori cascada care folosesc si alte tipuri de trasaturi:

- Local Binary Patterns (LBP): *lbpcascade_frontalface.xml* (*OpenCV\sources\data\lbpcascades*)
- Histogram of Oriented Gradients (HOG): *hogcascade_pedestrians.xml*,
(*OpenCV\sources\data\hogcascades*)

8.2. Detectia obiectelor folosind clasificatori cascada in OpenCV

Detectia obiectelor se poate realiza apaland functia:

```
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())
```

Unde:

- **CascadeClassifier** – clasa din care se instantiaza obiectul (modelul) pentru clasificatorul cascada. Modelul poate fi incarcat dintr-un fisier XML sau YAML folosind functia [Load\(\)](#).
- **image** – Matrice de tip CV_8U continand imaginea sursa.
- **objects** – Vector de iesire continand ROI (dreptunghiuri) care contin obiectele detectate.
- **scaleFactor** – Parametru care specifica factorul de scalare folosit in detectia multirezolutive.
- **minNeighbors** – Parametru care specifica cati vecini ai fiecarui candidat (zone rectangulare) se retin.
- **flags** – Nu se foloseste in modelele cascada de tip nou (doar in cele vechi – cvHaarDetectObjects).
- **minSize** – Dimensiunea rectangulara minima a obiectului detectabil. Obiecte mai mici sunt ignorate.
- **maxSize** – Dimensiunea rectangulara maxima a obiectului detectabil. Obiecte mai mari sunt ignorate.

Un tutorial care exemplifica folosirea clasificatorului cascada pentru detectia fetelor din imagini este prezentat in [4]. Se indica modul in care se poate realiza detectia tuturor fețelor dintr-o imagine. Pentru fiecare față detectata (in regiunea rectangulara care incadreaza fiecare față) se poate face detectia componentelor faciale (ochi, gura, nas). Un sablon derivat din acest tutorial este dat in forma functiei este dat mai jos:

```
/* -----
---
Detects all the faces and eyes in the input image
window_name - name of the destination window in which the detection results are
displayed
frame - source image
minFaceSize - minimum size of the ROI in which a Face is searched
minEyeSize - minimum size of the ROI in which an Eye is searched
    according to the antropomorphic features of a face, minEyeSize = minFaceSize / 5
Usage: FaceDetectandDisplay( "Dst", dst, minFaceSize, minEyeSize );
-----
- */
void FaceDetectandDisplay( const string& window_name, Mat frame,
                           int minFaceSize, int minEyeSize )
{
    std::vector<Rect> faces;
    Mat frame_gray;

    cvtColor( frame, frame_gray, CV_BGR2GRAY );
    equalizeHist( frame_gray, frame_gray );

    //-- Detect faces
    face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE,
                                   Size(minFaceSize, minFaceSize) );

    for( int i = 0; i < faces.size(); i++ )
    {
        // get the center of the face
        Point center( faces[i].x + faces[i].width*0.5, faces[i].y + faces[i].height*0.5 );
        // draw circle around the face
        ellipse( frame, center, Size( faces[i].width*0.5, faces[i].height*0.5), 0, 0,
                360, Scalar( 255, 0, 255 ), 4, 8, 0 );

        Mat faceROI = frame_gray( faces[i] );
```

```

std::vector<Rect> eyes;
//-- In each face (rectangular ROI), detect the eyes
eyes_cascade.detectMultiScale( faceROI, eyes, 1.1, 2, 0 |CV_HAAR_SCALE_IMAGE,
                               Size(minEyeSize, minEyeSize) );

for( int j = 0; j < eyes.size(); j++ )
{
    // get the center of the eye
    //atentie la modul in care se calculeaza pozitia absoluta a centrului ochiului
    // relativa la coltul stanga-sus al imaginii:
    Point center( faces[i].x + eyes[j].x + eyes[j].width*0.5,
                  faces[i].y + eyes[j].y + eyes[j].height*0.5 );
    int radius = cvRound( (eyes[j].width + eyes[j].height)*0.25 );
    // draw circle around the eye
    circle( frame, center, radius, Scalar( 255, 0, 0 ), 4, 8, 0 );
}
}
imshow( window_name, frame ); //-- Show what you got
}

```

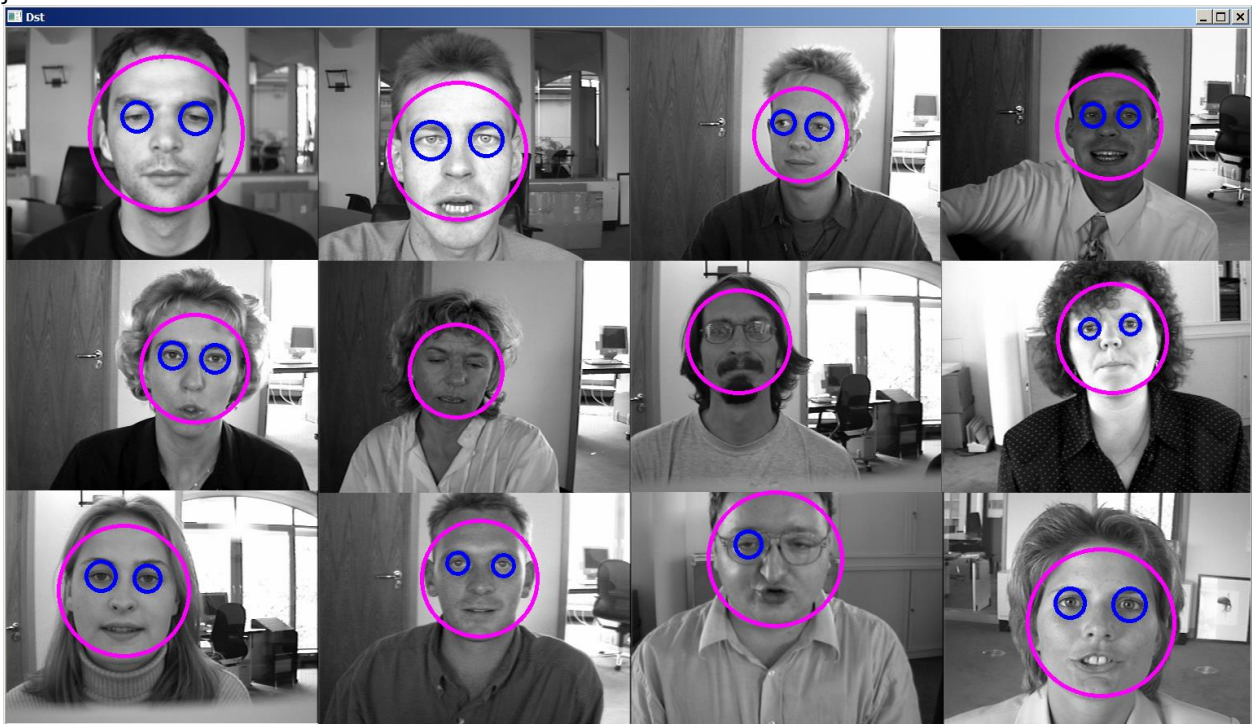


Fig. 8.1. Rezultatele apelului functiei FaceDetectandDisplay.

8.3. Detalii de implementare

In modulul *OpenCVApplication.cpp* trebuie sa mai faceti urmatoarele:

- sa declarati urmatoarele variabile globale:


```

CascadeClassifier face_cascade; // cascade classifier object for face
CascadeClassifier eyes_cascade; // cascade classifier object for eyes
CascadeClassifier mouth_cascade; // cascade classifier object for face
CascadeClassifier nose_cascade; // cascade classifier object for eyes
      
```
- sa va asigurati ca modelele folosite ale clasificatorilor cascada (fisierle *.xml)


```

haarcascade_eye_tree_eyeglasses.xml
haarcascade_frontalface_alt.xml
haarcascade_mcs_mouth.xml
haarcascade_mcs_nose.xml
      
```

exista in directorul de lucru current (directorul radacina) al aplicatiei OpenCVApplication.

În funcția de procesare aferentă laboratorului curent, înainte de apelul funcției `FaceDetectandDisplay`, este necesară încărcarea modelelor clasificatorilor, ca și în exemplul de mai jos.

```
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
String mouth_cascade_name = "haarcascade_mcs_mouth.xml";
String nose_cascade_name = "haarcascade_mcs_nose.xml";

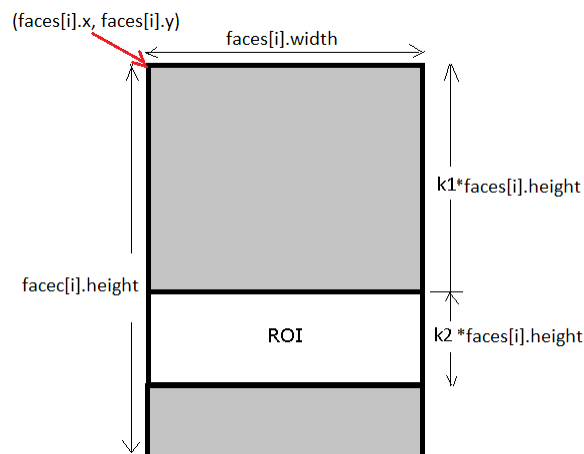
// Load the cascades
if (!face_cascade.load(face_cascade_name))
{
    printf("Error loading face cascades !\n");
    return;
}
if (!eyes_cascade.load(eyes_cascade_name))
{
    printf("Error loading eyes cascades !\n");
    return;
}
.....
```

Exemplu de apel al funcției `FaceDetectandDisplay` (în funcția de procesare aferentă laboratorului curent):

```
...
src = imread("Images/Facesx12.bmp", CV_LOAD_IMAGE_COLOR);
dst=src.clone();
int minFaceSize = 30;
int minEyeSize = minFaceSize / 5; // conform proprietatilor antropomorfe ale
fetei (idem pt. gura si nas)
FaceDetectandDisplay( WIN_DST, dst, minFaceSize, minEyeSize );
...
```

8.4. Restrictionarea detectiei in regiuni de interes (ROI)

Pt. detectia altor componente faciale (ex. gura, nas), modelul cascada `haarcascade_mcs_mouth.xml` nu este foarte precis! În consecință se poate restricționa regiunea de căutare (ROI) într-o subregiune a feței pt. a evita detectii fals-pozitive. Pt. căutarea componentelor faciale într-o ROI care să fie o subregiune a feței se va proceda astfel:



Se definește o nouă ROI rectangulară pentru o subregiune a feței:

```
Rect comp_rect;
comp_rect.x=faces[i].x;
comp_rect.y=faces[i].y + k1*faces[i].height;
comp_rect.width=faces[i].width;
comp_rect.height= k2*faces[i].height;
```

unde: $k1$, $k2$ sunt valori subunitare si $k1 + k2 < 1$ (vedeti valorile specificate la punctul 2 din mersul lucrarii !!!)

Se extrage din imaginea sursa matricea de intensitati corespunzatoare ROI definite:

```
Mat comp_ROI = frame_gray(comp_rect);
```

Se apeleaza detectorul de obiecte. De exemplu pt. gura procedati astfel:

```
Rect mouth_ROI;
std::vector<Rect> mouth;
//-- In each face (rectangular ROI), detect the mouth
mouth_cascade.detectMultiScale(mouth_ROI, mouth, 1.1, 1, 0 | CV_HAAR_SCALE_IMAGE,
                               Size(minMouthSize, minMouthSize) );
```

8.5. Aplicatii practice

1. Se va integra intr-o functie de procesare noua metoda de detectie a fetelor pentru detectia tuturor fetelor dintr-o imagine si a ochilor din fiecare fata (functia FaceDetectandDisplay). Imagini de test puteti descarca de la urmatoarea adresa: <http://users.utcluj.ro/~tmarita/HCI/Media/Images/Faces.zip>
2. Se va extinde procesarea de la pasul 1 pentru detectia si a altor componente faciale (gura, nas). Optimizati cautarea componentelor faciale doar in anumite zone din ROI-ul corespondent fetei ca si in exemplul de mai jos:

```
Rect eyes_rect; //eyes are in the 20% ... 55% height of the face
eyes_rect.x = faces[i].x;
eyes_rect.y = faces[i].y + 0.2*faces[i].height;
eyes_rect.width = faces[i].width;
eyes_rect.height = 0.35*faces[i].height;
Mat eyes_ROI = frame_gray(eyes_rect);
Rect nose_rect; //nose is the 40% ... 75% height of the face
...
Rect mouth_rect; //mouth is in the 70% ... 99% height of the face
...
```

Observatii:

- Cel mai simplu este sa afisati componentele faciale ca si dreptunghiuri (folositi functia rectangle: rectangle(frame, rect, culoare, grosime, 8, 0);
- atentie la modul in care se calculeaza pozitia absoluta a componentei faciale relative la coltul stanga-sus al imaginii: componentele detectate sunt detectate relativ la regiunea de interes (ROI) in care le-ati cautat \Rightarrow mai departe trebuie sa le translatati relativ la coltul stanga-sus al imaginii ca sa le afisati corect in imaginea destinatie (vezi exemplul dat in FaceDetectandDisplay)

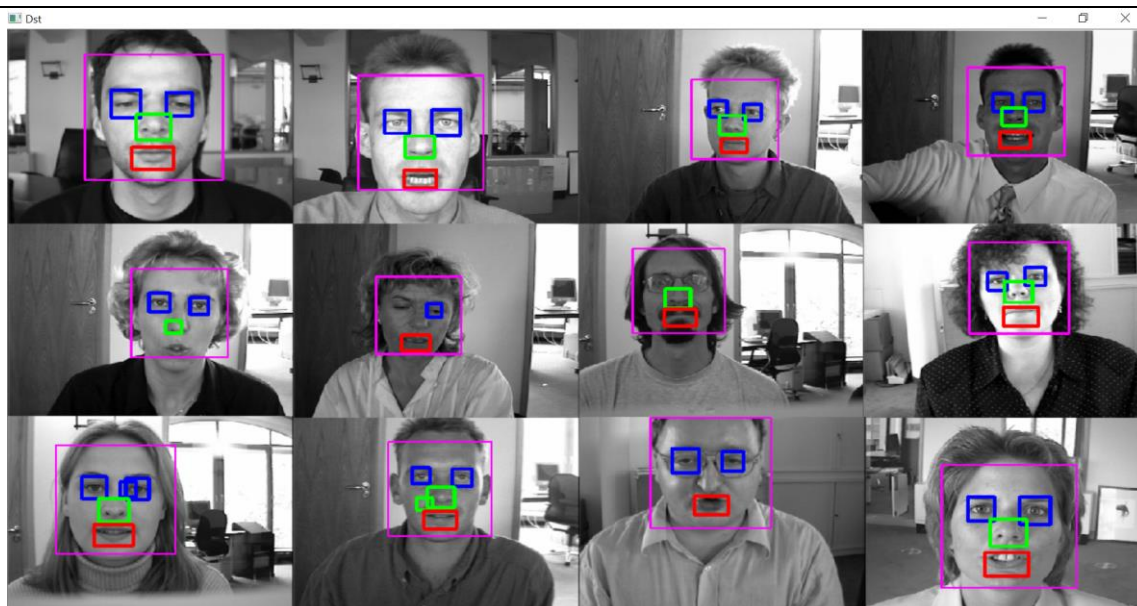


Fig. 8.2. Rezultatele apelului funcției FaceDetectandDisplay modificate pentru detectia fețelor și componentelor faciale (variante optimizată prin restricționarea zonelor în care se caută componentele faciale)

3. Se va crea o nouă funcție de procesare pentru detectia fețelor din secvențe video (off-line / on-line). Pentru acest task creați o versiune simplificată a funcției FaceDetectandDisplay care să detecteze și afișeze doar fețele (renunțați la căutarea celorlalte componente faciale). Se va măsura și afișa (la linia de comandă) timpul de procesare pentru apelul funcției FaceDetectandDisplay.
4. Se va testa detectia fețelor (doar pt. fete, fără componente faciale) folosind modelul de clasificator bazat pe trasături LBP (Local Binary Pattern) în locul celui bazat pe trasături Haar. Pt. aceasta se va încărca modelul clasificatorului LBP *lbpcascade_frontalface.xml* în locul *haarcascade_frontalface_alt.xml*

8.6. Bibliografie

- [1] Paul Viola, Michael Jones, Robust Real-time Object Detection, SECOND INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF VISION—MODELING, LEARNING, COMPUTING, AND SAMPLING VANCOUVER, CANADA, JULY 13, 2001, <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf> <http://cs.nyu.edu/~eugene/publications/viola-facedet04-talk.pdf>
- [2] T. Marita, Interacțiune Om-Calculator, note de curs, <http://users.utcluj.ro/~tmarita/HCI/C7-8.pdf>.
- [3] OpenCV Face Detection: Visualized, <http://vimeo.com/12774628>
- [4] OpenCV Tutorials, *objdetect* module. Object Detection, http://docs.opencv.org/2.4.11/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html#cascade-classifier