10. Validarea detectiei fetelor si a componentelor faciale pe sevente de imagini

10.1. Introducere

Scopul acestei lucrari este de a integra metoda Viola&Jones de detectie a feţelor si a componentelor faciale (in principal ochii) cu o validare bazata pe analiza unei secvente de imagini. Genul de utilizare a unei astfel de abordari este in aplicatii de tip „liviness detection” in sistemele biometrice bazate pe recunoasterea de fete. Astfel de aplicatii inceraca sa valideze suplimentar o detectie + o recunoastere de fata printr-o analiza suplimentara bazata pe miscare pentru a preveni incercarile de fraudare a sistemului prin prezentarea unui imagini a personei in loc de prezenta efectiva a persoanei.

La modul concret se va face o validare bazata pe *detectia clipitului* persoanei respective. D.p.d.v. practic se va realiza o integrare a implementarilor din lucrarile L6 (Segmentarea obiectelor in miscare prin modelarea fundalului “Background Subtraction”) si L9 (Detectia feţelor si a componentelor faciale) + procesari de imagini uzuale.

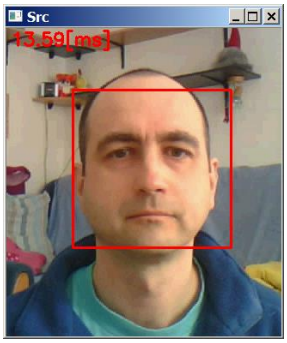
10.2. Mersul lucrarii

Se va deschide secventa video de test (`test_msv1_short.avi`). Procesarea se va face frame cu frame, avansul intre cadre facandu-se prin apasarea unei taste (setati la ‚0’ parametrul functiei `cvWaitKey(0)`)

Se va converti frame-ul curent (de analizat) din color in grayscale.

Pentru fiecare frame/cadru (incepand cu al doilea) se vor realiza urmatoarele operatii:

1. Se va apela metoda de detectie a fetelor si a componentelor faciale pe frame-ul curent. Se va memora pozitia (dreptunghiul care incadreaza) prima faţă gasita in frame-ul curent (`Rect faceROI=faces[0]`). Acest dreptunghi se va folosi ulterior ca si masca de procesare (Regions Of Interest – ROI).



2. Se va implementa operatia de „Background Subtraction” prin metoda de diferentiere simpla intre cadrul curent si cel precedent. In momentul in care persoana clipeste in dreptul ochilor vor aparea zone cu arie mai mare marcate cu alb:

Imaginea diferenta dintre cadrul curent si cadrul trecut

3. Pe imaginea diferenta se vor aplica operatii morfologice (erozoine + dilatare) pt. eliminarea zgomotelor datorate eventualelor miscarii ale capului sau zgomotului de achizitie / compresie al secventei video.

Imagine diferenta dupa aplicarea filterlor morfologice
(ex: eroziune 1x + dilatare 1x cu nucleu de 3x3)

4. Se va masca imaginea diferenta finala (*dst*) de la pasul 3 cu ROI-ul feței obtinut la pasul 1):

```
temp = dst(faceROI); // mat grayscale pt. procesarea ROI
```

5. Se realizeaza etichetarea obiectelor ramase in zona ROI aferenta fetei (din matricea *temp*). Se calculeaza proprietatile geometrice ale obiectelor unificate: aria, centrul de masa.

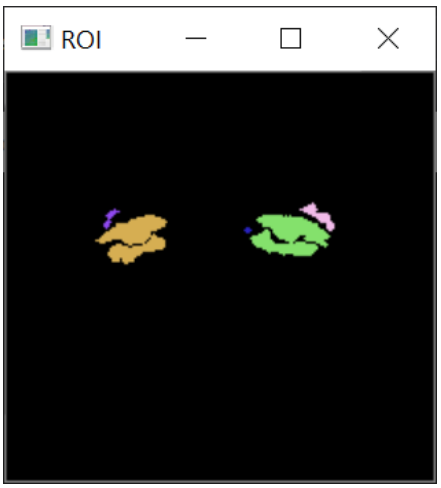
Pt. etichetarea obiectelor si calculul proprietatilor geometrice (arie si centre de masa) folositi codul dat ca si exemplu in functia `Labeling` din modulul *Functions*.

```
// Labeling -----------------------------------------
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
roi = Mat::zeros(temp.rows, temp.cols, CV_8UC3); // matrice (3 canale) folositea pentru
afisarea (color) a obiectelor detectate din regiunea de interes
findContours(temp, contours, hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);
Moments m;
if (contours.size() > 0)
{
	// iterate through all the top-level contours,
	// draw each connected component with its own random color
	int idx = 0;
	for (; idx >= 0; idx = hierarchy[idx][0])
	{
		const vector<Point>& c = contours[idx];

		m = moments(c); // calcul momente
		double arie = m.m00; // aria componentei conexe idx
		double xc = m.m10 / m.m00; // coordonata x a CM al componentei conexe idx
		double yc = m.m01 / m.m00; // coordonata y a CM al componentei conexe idx

		Scalar color(rand() & 255, rand() & 255, rand() & 255);
		drawContours(roi, contours, idx, color, CV_FILLED, 8, hierarchy);

	}
}
```

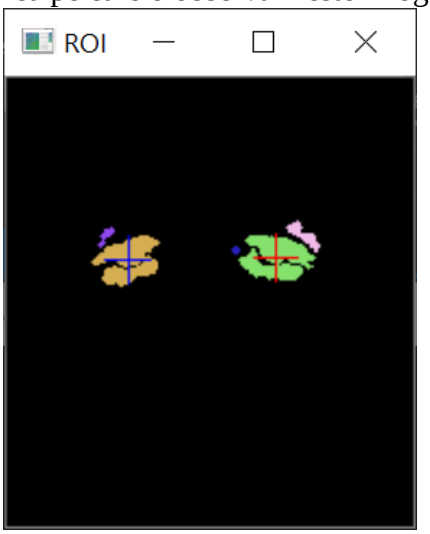


6. Se vor adauga ariile si centrele de masa ale obiectelor detectate intr-o lista (vector) de candidati Folositi o structura pentru a memora elementele vectorului:

```
typedef struct {
	double arie;
	double xc;
	double yc;
} mylist;
vector<mylist> candidates;
candidates.clear(); // se apeleaza pt. fiecare cadru
// Labeling code …
	// Pt. fiecare obiect
		// adauga PGS in lista candidates
			mylist elem;
			elem.arie = arie;
			elem.xc = xc;
			elem.yc = yc;
			candidates.push_back(elem);
```

7. Se va implementa un abore de decizie bazat pe cateva reguli simple:

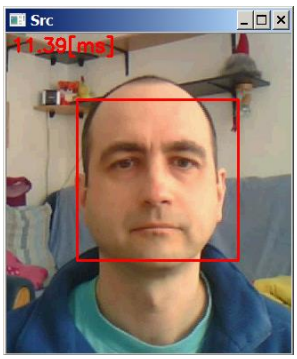
- Se retin pozitiile centrelor de masa ale celor doua obiecte (din lista de candidati) care au aria cea mai mare (ca sa eliminati eventualele zgomote ramase in imgine chiar si dupa aplicarea filtrelor morfologice). Se vor desena in fereastra aferenta ROI (matricea *roi*) centrele de masa folosind functia `DrawCross` (din modulul functions): cu rosu pt. ochiul stang si cu albastru pentru ochiul drept (imaginea pe care o observam este in oglinda).

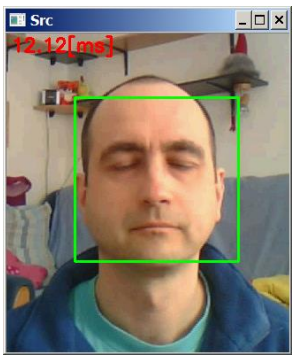


- Pozitiile centrelor de masa (CM) ale acestor obiecte trebuie sa fie pe o linie orizontala (ex: diferentele pe y ale CM sa fie mai mici decat o valoare ky*ROI.height (ex: ky ≈ 0.1) ȘI sa fie situate in jumatatea superioara a ROI)
- Diferentele pe orizontala (distanta pupilara DP) dintre centrele de masa (CM) sa fie un procent din latimea fetei: kx1*ROI.width < DP < kx2*ROI.width (ex: kx1 ≈ 0.3, kx2 ≈ 0.5)
- Pozitiile centrelor de masa apar in cele 2 jumatati verticale diferite ale feței: xc_ochi_ochi stang > ROI.width/2 (ocuiul stang il vedem in imagine in dreapta) si xc_ochi_ochi_drept < ROI.width/2 (ochiul drept il vedem in imagine in stanga)
-

Obs: Va trebui sa ajustati eventualele constante (kx, ky la valori optime)!.

8. Se va afisa in imaginea sursa faţa detectata cu culoarea verde daca sa trecut cu succes prin procesul de validare (s-a detectat clipitul – ochi inchisi) si cu rosu in caz contrar (ochii deschisi). Se va afisa si timpul de procesare.





Pentru desenarea unui dreptunghi aveti la dispozitie in OpenCV functia `rectangle(frame, p1, p2, col, 2, 8, 0)`, unde *p1* si *p2* sunt structuri de tip *Point* continand coordonatele coltului stanga sus, respective dreapta jos ale dreptunghiului de desenat sau OpenCV functia `rectangle(frame, rect, col, 2, 8, 0)`